LEMBAR LAPORAN PRAKTIKUM PAPB

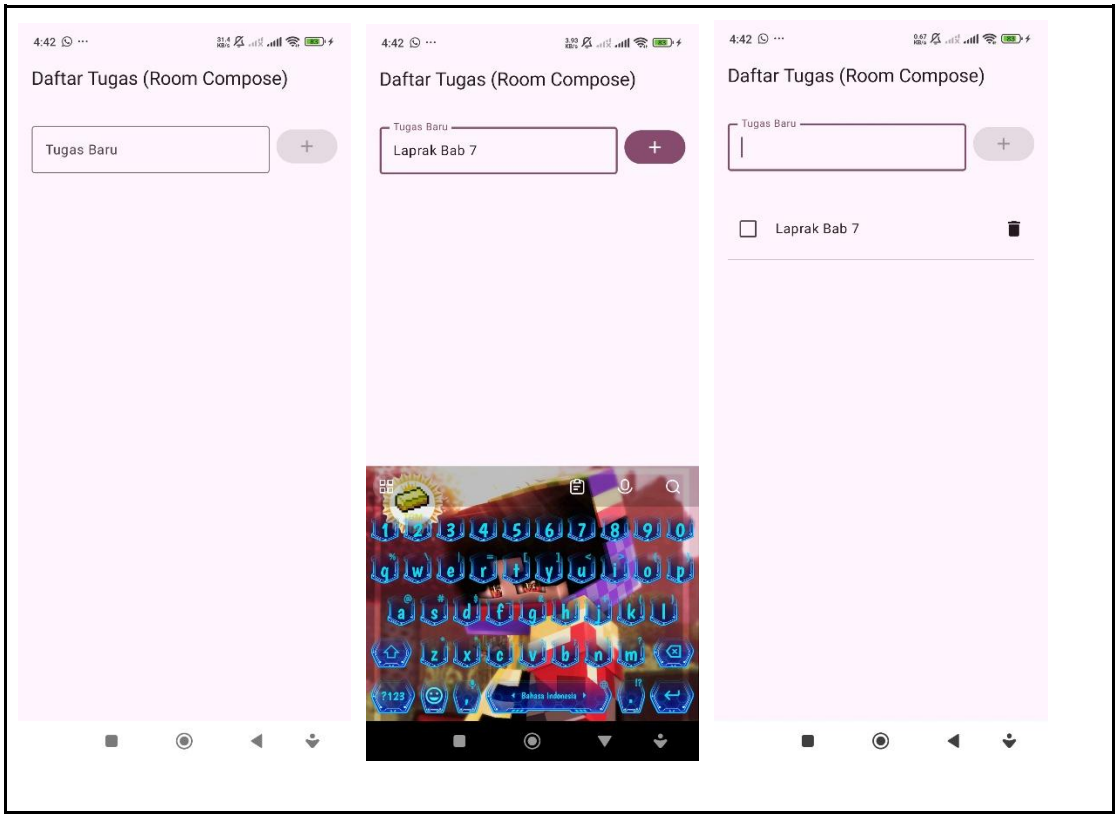
BAB: 7
NAMA: NADHIF RIF'AT RASENDRIYA
NIM: 235150201111074
ASISTEN: 1. RIFKY AKHSANUL HADI
TGL PRAKTIKUM: 04 NOVEMBER 2025

1. TUGAS DAN REFLEKSI

1. Cobalah fungsi CRUD pada aplikasi yang telah berhasil berjalan:
a. Create: Menambahkan tugas baru
b. Read: Menampilkan tugas yang telah ditambahkan pada to-do list
c. Update: Mengubah status tugas (centang)
d. Delete: Menghapus tugas
Lampirkan screenshot aplikasi sebagai jawaban tugas ini.

Jawaban

Screenshot



Link kode program dan README.md

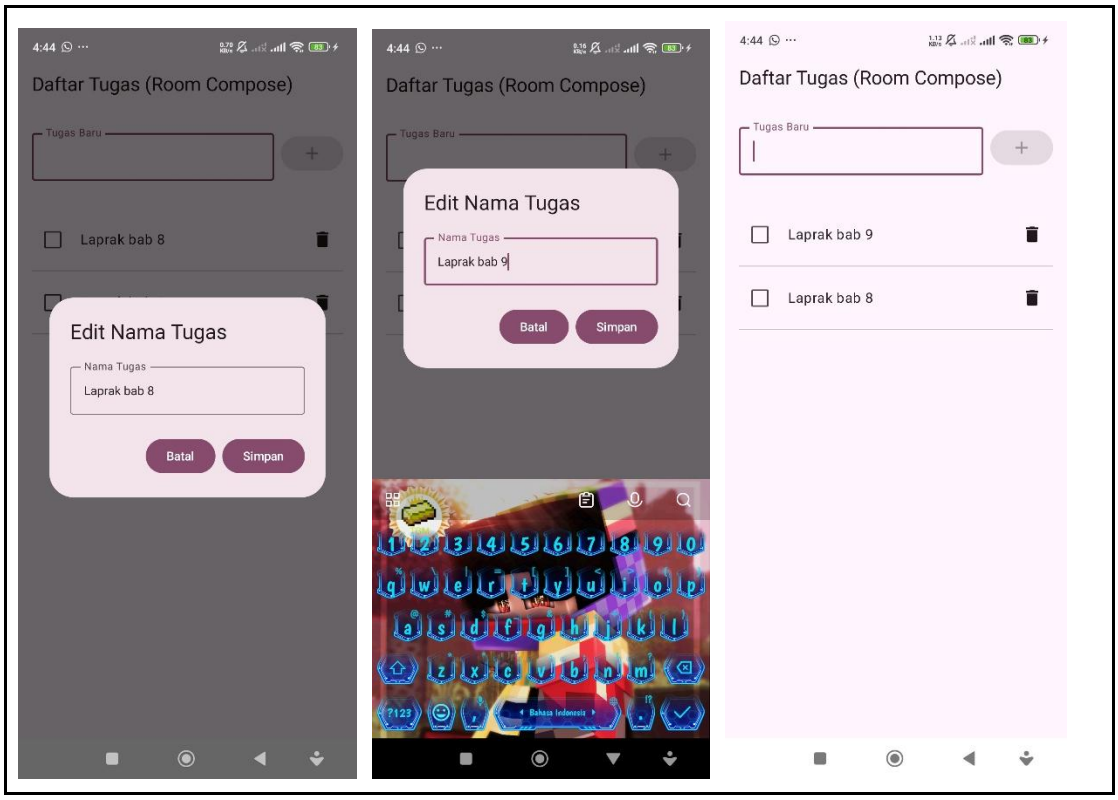
Link : drive.google.com / github.com (pastikan folder atau repo public)

https://github.com/nadhif-royal/PraktikumRoomToDoList

2. Tambahkan fungsionalitas aplikasi sehingga dapat mengubah nama tugas pada to-do list! Lampirkan screenshot aplikasi beserta source code sebagai jawaban tugas ini.

Jawaban

Screenshot



Link kode program dan README.md

Link : drive.google.com / github.com (pastikan folder atau repo public)

https://github.com/nadhif-royal/PraktikumRoomToDoList

Kesimpulan

Praktikum ini berhasil membuktikan secara nyata konsep-konsep inti dari *library* Room yang dijelaskan dalam modul. Kita telah berhasil mengimplementasikan ketiga komponen utamanya: Task.kt sebagai **Entity** yang mendefinisikan struktur tabel, TaskDao.kt sebagai **DAO** yang menyediakan metode abstraksi untuk CRUD, dan TaskDatabase.kt sebagai **Database Class** yang menginisialisasi Room. Implementasi ini juga secara ketat mengikuti arsitektur **MVVM** yang dianjurkan , di mana TaskRepository bertindak sebagai *single source of truth* yang memisahkan logika data (DAO) dari TaskViewModel

Praktikum ini juga menerapkan aturan kritis modul mengenai operasi asinkron untuk mencegah *ANR* (Application Not Responding). Sesuai teori, operasi *write* (create, update, delete) di TaskDao diimplementasikan sebagai suspend function dan dieksekusi dengan aman di *background thread* menggunakan **Kotlin Coroutines** (via viewModelScope). Untuk operasi *read*, kita memanfaatkan dukungan Room terhadap **Flow** melalui fungsi getAllTasks(). Hal ini memungkinkan MainActivity untuk mengobservasi data secara *real-time*, sehingga UI dapat diperbarui secara otomatis setiap kali ada perubahan data di database, persis seperti yang dijelaskan dalam dasar teori.