BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan maka diperoleh kesimpulannya sebagai berikut:

1. Penelitian menghasilkan aplikasi permohonan cuti karyawan berbasis sistem operasi Android menggunakan bahasa Kotlin dengan alur pengguna yang sepenuhnya berada di dalam aplikasi. Proses pengelolaan akun dan permohonan cuti bisa dilakukan tanpa harus keluar dari aplikasi.
2. Penerapan pola arsitektur MVVM membantu untuk membuat aplikasi yang jelas pemisahannya. MVVM aplikasi menjadi 3 bagian berupa *model* yang mewakili data dan logika bisnis aplikasi, *view* sebagai antarmuka pengguna yang menangani presentasi data dan interaksi dengan pengguna sedangkan *viewmodel* mengubah data dari *model* ke format yang dapat ditampilkan oleh *view*.
3. Implementasi *Reactive Programming* memanfaatkan aliran data dalam bentuk Kotlin Flow untuk mengirim data dari *model* ke *viewmodel*, dan LiveData untuk mengirim data dari *viewmodel* ke *view*. *Flow* dijalankan di *thread* I/O (Dispatchers.IO) untuk operasi pada aliran data yang melibatkan akses ke basis data atau sumber daya lain yang

dapat menyebabkan kelambatan jika dilakukan di *thread* utama. LiveData, di sisi lain, dijalankan di *thread* utama (Dispatchers.Main) yang melakukan *render* pada antarmuka.

4. Penggunaan *state* antarmuka memungkinkan untuk menghasilkan aplikasi yang responsif terhadap input pengguna dengan memberikan indikator *loading* dan memberikan data apabila *success*, namun akan memberikan pesan apabila terjadi *error* secara langsung.
5. Berdasarkan pengujian yang telah dilakukan, aplikasi mampu untuk mengelola permohonan cuti, termasuk pengajuan, persetujuan, dan pengecekan status cuti sesuai dengan *Use Case* yang ditentukan sebelum sampai ke tangan *End User*. Aplikasi juga bisa berjalan lancar mulai dari versi Android 7.0 ke atas.

5.2 Saran

Adapun beberapa hal yang disarankan untuk pengembangan penelitian ini adalah sebagai berikut.

1. Menambah fitur aplikasi seperti absensi karyawan dan kuota cuti.
2. Menambahkan peran pengguna selain karyawan dan manajer.
3. Menggunakan Android Jetpack Compose dan Kotlin Multiplatform
4. Merancang REST API sendiri menggunakan PostgreSQL dan Spring Boot *Framework* dengan bahasa Kotlin sehingga lebih memiliki kontrol pada bagian *back-end* aplikasi.

5. Mencoba menggunakan pola arsitektur lain yang terbaru seperti MVI (*Model-View-Intent*).
6. Mengeksplorasi teknologi *reactive programming* lainnya seperti RxJava
7. Mengaplikasikan paradigma *reactive programming* dengan aplikasi yang memiliki spesifikasi kebutuhan menerima data secara terus menerus, seperti aplikasi dengan sensor atau aplikasi yang memanfaatkan lokasi.
8. Menganalisis masalah-masalah konkurensi yang bisa terjadi pada aplikasi Android.