

BAB 3

PELAKSANAAN MBKM PROYEK KEMANUSIAAN

3.1 Kedudukan dan Koordinasi

Pelaksanaan program MBKM Proyek Kemanusiaan dengan kedudukan sebagai *Software Engineer* di Gugus Mitigasi Lebak Selatan (GMLS). GMLS merupakan sebuah organisasi yang bergerak di bidang kemanusiaan, dengan fokus utama pada upaya mitigasi bencana di wilayah Lebak Selatan, Provinsi Banten. Mengingat lingkup kerja GMLS yang mencakup pemanfaatan teknologi untuk pengurangan risiko bencana, peran seorang *software engineer* menjadi krusial.

Secara struktural dalam organisasi GMLS, posisi *Software Engineer* berada di bawah naungan divisi "*Data & Technology*". Divisi ini dikepalai oleh Dayah Fata Fadillah. Tanggung jawab utama divisi ini meliputi berbagai aspek teknis yang vital bagi operasional GMLS, seperti pengembangan peta rawan bencana berbasis Sistem Informasi Geografis (SIG), pengelolaan *database* kebencanaan yang komprehensif, manajemen sistem peringatan dini (*EWS*) yang mencakup sensor, aplikasi, dan sirene, serta integrasi teknologi modern seperti *drone* untuk pemantauan dan analisis pasca-bencana.

GMLS sendiri merupakan organisasi yang relatif ramping, dengan tim inti yang terdiri dari lima orang pengurus utama. Kondisi ini menciptakan lingkungan kerja yang dinamis dan kolaboratif. Keterbatasan jumlah personel inti mengindikasikan bahwa mahasiswa MBKM Proyek Kemanusiaan, termasuk pada posisi *Software Engineer*, memiliki peluang besar untuk terlibat secara langsung dalam berbagai tahapan proyek teknologi. Interaksi yang lebih intensif dan jalur komunikasi yang pendek dengan kepala divisi, bahkan dengan Direktur GMLS, Bapak Anis Faisal Reza, menjadi sebuah keniscayaan, terutama untuk proyek-proyek yang memiliki nilai strategis bagi organisasi. Fleksibilitas dalam alur kerja dan koordinasi menjadi ciri khas, namun di sisi lain menuntut tingkat kemandirian dan inisiatif yang tinggi dari mahasiswa MBKM Proyek Kemanusiaan.

3.2 Tugas yang Dilakukan

Selama periode MBKM Proyek Kemanusiaan, fokus utama adalah pada pembuatan aplikasi *Android* untuk pelaporan, *monitoring*, dan analisis bencana pada Gugus Mitigasi Lebak Selatan. Tugas-tugas spesifik yang dilakukan meliputi:

1. Pengembangan sistem autentikasi pengguna menggunakan *Firebase Authentication* untuk keamanan akses aplikasi.
2. Implementasi *splash screen* dan alur *onboarding* untuk pengenalan aplikasi kepada pengguna baru.
3. Membangun *dashboard* utama yang menampilkan statistik bencana, peta sebaran bencana, dan akses cepat ke fitur pelaporan.
4. Pengembangan modul pelaporan bencana dengan *form input* lokasi (*GPS/manual*), jenis bencana, tingkat kerusakan, deskripsi kejadian, dan *upload* foto dokumentasi.
5. Membuat sistem *monitoring* bencana dengan daftar laporan *real-time*, status verifikasi, dan *tracking* perkembangan penanganan.
6. Implementasi peta interaktif menggunakan *Google Maps SDK* dengan *marker* lokasi bencana, *clustering*, dan *layer* informasi geografis.
7. Membangun fitur analisis bencana dengan visualisasi data statistik, grafik *trend* kejadian, dan laporan berkala.
8. Pengembangan sistem notifikasi *push* untuk *alert* bencana darurat dan *update* status laporan.
9. Membuat modul profil pengguna untuk *management* data *personal* dan preferensi notifikasi.
10. Implementasi *panel admin* untuk verifikasi laporan, *management* pengguna, dan *monitoring system*.
11. Menerapkan arsitektur *MVVM* dengan *dependency injection* menggunakan *Hilt* untuk struktur kode yang *maintainable*.
12. Integrasi *Firebase Firestore* untuk penyimpanan data *real-time* dan sinkronisasi *multi-device*.
13. Implementasi sistem *offline-first* dengan *Room Database* untuk akses data tanpa koneksi internet.
14. Menerapkan *security measures* dan validasi data untuk memastikan integritas *information* bencana.

3.3 Uraian Pelaksanaan MBKM Proyek Kemanusiaan

Minggu	Pekerjaan yang dilakukan
1	Pengenalan organisasi GMLS dan orientasi lapangan untuk memahami kebutuhan sistem pelaporan bencana.
2	Wawancara <i>stakeholder</i> dan pembelajaran <i>Android Development</i> untuk persiapan pengembangan aplikasi.
3	<i>Setup project Android</i> dengan arsitektur <i>MVVM</i> dan konfigurasi <i>Firebase</i> untuk <i>backend services</i> .
4	Implementasi sistem autentikasi <i>Firebase</i> dan pengembangan <i>splash screen</i> dengan <i>onboarding flow</i> .
5	Pembuatan <i>dashboard</i> utama dengan <i>card-based layout</i> dan implementasi <i>location tracking</i> .
6	Pengembangan <i>form</i> pelaporan bencana dengan validasi <i>input</i> dan integrasi <i>GPS</i> serta <i>image capture</i> .
7	Implementasi sistem <i>monitoring</i> laporan <i>real-time</i> dengan <i>RecyclerView</i> dan fitur <i>search filtering</i> .
8	Pengembangan <i>push notification FCM</i> dan visualisasi data dengan <i>charting</i> dan <i>clustering</i> peta.
9	Pembuatan panel <i>admin</i> dengan <i>dashboard analytics</i> dan fitur <i>approve/reject</i> laporan bencana.
10	Implementasi sistem notifikasi <i>push</i> dan <i>alert emergency</i> menggunakan <i>Firebase Cloud Messaging</i> .
11	Pengembangan fitur <i>settings</i> dan konfigurasi aplikasi untuk personalisasi pengguna.
12	Implementasi fitur <i>resources</i> dan <i>emergency information</i> untuk panduan kesiapsiagaan bencana.
13	Pengembangan fitur <i>analytics</i> dan visualisasi data dengan <i>charting</i> untuk <i>dashboard admin</i> .
14	Implementasi sistem <i>offline-first</i> dengan <i>Room Database</i> dan optimasi performa aplikasi.
15	Finalisasi dokumentasi teknis dan optimasi <i>security measures</i> untuk integritas data.

Tabel 3.1. Jadwal Kegiatan MBKM Proyek Kemanusiaan

Pelaksanaan MBKM Proyek Kemanusiaan dilakukan selama 15 minggu dari tanggal 17 Februari hingga 28 Mei 2025, dengan 3 kali trip ke lokasi GMLS di Lebak Selatan untuk koordinasi langsung dan implementasi sistem. Rincian kegiatan yang dilakukan tiap minggu ditunjukkan pada Tabel 3.1.

3.3.1 Penjelasan Pelaksanaan per Minggu

Minggu pertama dan kedua difokuskan pada orientasi mendalam dan analisis kebutuhan sistem. Kegiatan dimulai dengan pengenalan organisasi GMLS dan orientasi lapangan untuk memahami kondisi geografis serta tantangan kebencanaan di wilayah tersebut. Pembelajaran intensif *Android Development* dan bahasa pemrograman *Kotlin* dilakukan melalui dokumentasi resmi dan *hands-on practice*, diikuti dengan pembuatan *wireframe* dan *UI/UX design* aplikasi menggunakan *Figma*.

Minggu ketiga dan keempat fokus pada *setup project Android* dengan arsitektur *MVVM*, konfigurasi *dependency injection* dengan *Hilt*, dan *setup Firebase project* untuk *backend services*. Implementasi *Firebase Authentication* untuk sistem *login/register* pengguna dilakukan dengan validasi yang *robust*, diikuti dengan pengembangan *interface* utama aplikasi dan *dashboard* yang menampilkan statistik bencana terkini.

Minggu kelima dan keenam difokuskan pada implementasi modul pelaporan bencana sebagai fitur inti aplikasi. Pengembangan *form input* data kejadian dengan *field* lengkap dan integrasi *GPS* untuk lokasi otomatis dilakukan, diikuti dengan pengembangan fitur *upload* foto dokumentasi bencana dan integrasi *Google Maps SDK* untuk visualisasi lokasi.

Minggu ketujuh dan kedelapan difokuskan pada pembuatan sistem *monitoring* bencana dengan daftar laporan *real-time* dan implementasi *push notification* menggunakan *Firebase Cloud Messaging*. Pengembangan fitur analisis bencana menggunakan visualisasi data statistik, implementasi *clustering* pada peta interaktif, dan sistem *filtering* berdasarkan periode waktu dan jenis bencana juga dilakukan pada periode ini.

Minggu kesembilan dan kesepuluh difokuskan pada implementasi panel *admin* dan sistem notifikasi *push*. Pengembangan panel *admin* yang komprehensif untuk verifikasi laporan dan *dashboard admin* dengan *analytics real-time* dilakukan, diikuti dengan implementasi sistem notifikasi *push* dan *alert emergency* menggunakan *Firebase Cloud Messaging* untuk memastikan penyebaran informasi darurat yang efektif.

Minggu kesebelas dan duabelas difokuskan pada pengembangan fitur pendukung aplikasi. Implementasi fitur *settings* dan konfigurasi aplikasi untuk personalisasi pengguna dilakukan, termasuk pengaturan tema, bahasa, notifikasi, dan privasi. Pengembangan fitur *resources* dan *emergency information* juga

dilakukan untuk menyediakan panduan kesiapsiagaan bencana yang komprehensif kepada pengguna.

Minggu ketigabelas difokuskan pada pengembangan fitur *analytics* dan visualisasi data dengan *charting* untuk *dashboard admin*. Implementasi berbagai jenis *chart* dan grafik untuk menampilkan statistik bencana, *trend* kejadian, dan analisis data yang komprehensif dilakukan untuk memberikan *insight* yang *valuable* bagi administrator dalam pengambilan keputusan.

Minggu keempatbelas difokuskan pada implementasi sistem *offline-first* dengan *Room Database* dan optimasi performa aplikasi. Pengembangan *database* lokal untuk menyimpan data sementara ketika tidak ada koneksi internet dilakukan, diikuti dengan implementasi sinkronisasi otomatis dan optimasi performa untuk memastikan aplikasi berjalan dengan *smooth*.

Minggu kelimabelas dialokasikan untuk finalisasi dokumentasi teknis dan optimasi *security measures* untuk integritas data. Penyelesaian dokumentasi komprehensif tentang arsitektur aplikasi, implementasi *security validation*, dan penerapan *best practices* untuk memastikan keamanan dan integritas data bencana dilakukan pada periode ini.

3.3.2 Detail Pekerjaan yang Dikerjakan

Selama periode MBKM Proyek Kemanusiaan, dilakukan berbagai aktivitas pengembangan aplikasi Android yang mencakup aspek teknis dan non-teknis. Pekerjaan yang dikerjakan dapat dikelompokkan menjadi lima tahap utama yang dilaksanakan secara bertahap dan terstruktur, mulai dari analisis kebutuhan hingga pengembangan fitur lanjutan aplikasi.

A Tahap 1: Analisis dan Perancangan Sistem

A.1 Analisis Kebutuhan Sistem

Tahap pertama dimulai dengan analisis mendalam terhadap kebutuhan sistem pelaporan bencana di GMLS. Proses ini diawali dengan studi literatur tentang sistem penanggulangan bencana dan *best practices* dalam pengembangan aplikasi *emergency response* untuk memahami standar industri yang berlaku. Selanjutnya dilakukan wawancara intensif dengan *stakeholder* GMLS untuk memahami *workflow* pelaporan bencana yang sudah ada, mengidentifikasi *pain points* dalam sistem manual yang digunakan sebelumnya, serta menganalisis

kebutuhan fungsional dan non-fungsional aplikasi. Tahap ini juga mencakup penentuan target pengguna dan persona pengguna aplikasi untuk memastikan desain yang sesuai dengan karakteristik pengguna akhir di wilayah Lebak Selatan.

A.2 Perancangan Arsitektur Sistem

Setelah kebutuhan teridentifikasi, dilakukan perancangan arsitektur sistem yang komprehensif. Pemilihan arsitektur MVVM (Model-View-ViewModel) menjadi fondasi utama untuk memisahkan logika bisnis dari tampilan, memungkinkan *maintainability* dan *testability* yang lebih baik. Implementasi arsitektur ini terlihat jelas dari pemisahan antara *ViewModel* yang memegang *state*, *Repository* sebagai abstraksi untuk *data access*, dan struktur paket ui, domain, dan data yang terorganisir dengan baik, sebagaimana ditunjukkan pada Kode 3.1.

```
1 // src/main/java/com/example/gmls/ui/viewmodels/AuthViewModel.kt
2 @HiltViewModel
3 class AuthViewModel @Inject constructor(
4     private val authRepository: AuthRepository, // Interaksi
5     private val locationService: LocationService
6 ) : ViewModel() {
7
8     private val _authState = MutableStateFlow<AuthState>(AuthState
9     .Initial)
10    val authState: StateFlow<AuthState> = _authState
11
12    fun login(email: String, password: String) {
13        viewModelScope.launch(viewModelExceptionHandler) {
14            _authState.value = AuthState.Loading
15            try {
16                val user = withTimeoutOrNull(OPERATION_TIMEOUT) {
17                    // Memanggil repository, bukan langsung ke
18                    // Firebase
19                    authRepository.login(email.trim(), password)
20                }
21            }
22            // ... handling response
23        }
24    }
25 }
```

Kode 3.1: Implementasi *MVVM* dengan *Clean Architecture*

Perancangan *database schema* untuk *Firebase Firestore* dilakukan dengan membuat koleksi *users* untuk data pengguna, *disasters* untuk laporan bencana, dan *admin_logs* untuk *audit trail* aktivitas administrator. Skema *database* ini diimplementasikan melalui *mapper classes* yang mendefinisikan struktur data secara eksplisit, seperti yang terlihat pada Kode 3.2.

```
1 // src/main/java/com/example/gmls/data/mapper/UserFirestoreMapper.  
  kt  
2 fun mapToFirestore(user: User): Map<String, Any?> {  
3     return mapOf(  
4         "email" to user.email,  
5         "fullName" to user.fullName,  
6         "phoneNumber" to user.phoneNumber,  
7         "role" to user.role,  
8         "isVerified" to user.isVerified,  
9         "isActive" to user.isActive,  
10        "nik" to user.nik,  
11        "address" to user.address,  
12        "createdAt" to user.createdAt,  
13        "updatedAt" to user.updatedAt  
14    )  
15 }
```

Kode 3.2: Database Schema Mapping untuk Firestore

Desain API endpoints dan data flow antara client dan server dirancang menggunakan *Repository pattern* yang mengabstraksi sumber data dan memungkinkan switching antara local dan remote data sources secara transparent. Interface repository mendefinisikan kontrak yang jelas untuk operasi data, sebagaimana diimplementasikan pada Kode 3.3.

```
1 // src/main/java/com/example/gmls/domain/repository/UserRepository  
  .kt  
2 interface UserRepository {  
3     suspend fun register(registrationData: RegistrationData):  
  Result<String>  
4  
5     suspend fun login(email: String, password: String): Result<  
  String>  
6  
7     suspend fun getUserProfile(userId: String): User  
8  
9     suspend fun updateUserProfile(userId: String, updates: Map<  
  String, Any>): Result<Unit>
```

```

10
11 suspend fun getAllUsers(): List<User>
12
13 suspend fun verifyUser(userId: String): Result<Unit>
14 }

```

Kode 3.3: *Repository Pattern* untuk *Data Abstraction*

Sistem autentikasi dan otorisasi menggunakan *Firebase Authentication* diintegrasikan untuk memberikan keamanan akses yang *robust*. Implementasi autentikasi dilakukan melalui *FirebaseAuthRepository* yang menangani proses *login* dan registrasi dengan validasi yang ketat, seperti yang ditunjukkan pada Kode 3.4.

```

1 // src/main/java/com/example/gmls/data/repository/
  FirebaseAuthRepository.kt
2 override suspend fun login(email: String, password: String): User
  {
3   val result = auth.signInWithEmailAndPassword(email, password).
    await()
4   val userDoc = firestore.collection("users").document(result.
    user!!.uid).get().await()
5   if (!userDoc.exists()) throw Exception("User data not found")
6   val data = userDoc.data ?: throw Exception("User data is null"
  )
7   return userMapper.mapToUser(result.user!!.uid, data)
8 }

```

Kode 3.4: Implementasi *Firebase Authentication*

Seluruh struktur *project Android* direncanakan mengikuti *clean architecture principles* untuk memastikan *scalability* dan *maintainability* jangka panjang, dengan pemisahan yang jelas antara *presentation layer*, *domain layer*, dan *data layer*.

A.3 Perancangan Antarmuka Pengguna (UI/UX)

Perancangan UI/UX dilakukan dengan fokus pada kemudahan penggunaan dalam situasi darurat, mengingat aplikasi ini akan digunakan dalam kondisi yang mungkin stressful dan membutuhkan akses cepat. Pembuatan *wireframe* dan *mockup* menggunakan *Figma* dilakukan untuk semua layar aplikasi, memastikan konsistensi visual dan *user experience* yang optimal.

Pemilihan color scheme disesuaikan dengan branding GMLS dan dioptimalkan untuk mudah dibaca dalam berbagai kondisi pencahayaan, termasuk kondisi outdoor dan low-light. Implementasi skema warna dilakukan melalui sistem tema yang komprehensif dengan dukungan dark dan light mode, sebagaimana terlihat pada Kode 3.5.

```
1 // src/main/java/com/example/gmls/ui/theme/Theme.kt
2 private val DarkColorScheme = darkColorScheme (
3     primary = AccentRed,
4     onPrimary = OilBlack,
5     background = OilBlack,
6     onBackground = Gray100,
7     surface = OilBlackSurface,
8     onSurface = Gray100,
9     error = Color(0xFFFF6B6B)
10    // ... pemetaan warna lainnya untuk tema gelap
11 )
12
13 @Composable
14 fun GMLSTheme (
15     appTheme: AppTheme = AppTheme.SYSTEM,
16     content: @Composable () -> Unit
17 ) {
18     val colorScheme = when (appTheme) {
19         AppTheme.LIGHT -> LightColorScheme
20         AppTheme.DARK -> DarkColorScheme
21         AppTheme.SYSTEM -> if (isSystemInDarkTheme ())
22             DarkColorScheme else LightColorScheme
23     }
24
25     MaterialTheme (
26         colorScheme = colorScheme,
27         typography = Typography,
28         content = content
29     )
30 }
```

Kode 3.5: Implementasi Color Scheme dan Theme System

Desain user flow untuk proses pelaporan bencana dirancang agar intuitif dan cepat, meminimalkan jumlah langkah yang diperlukan untuk melaporkan kejadian darurat. Implementasi user flow ini terlihat pada struktur ReportDisasterScreen yang menggunakan pendekatan step-by-step yang terorganisir, seperti yang

ditunjukkan pada Kode 3.6.

```
1 // src/main/java/com/example/gmls/ui/screens/disaster/
  ReportDisasterScreen.kt
2 @Composable
3 fun ReportDisasterScreen(...) {
4     Scaffold(...) { paddingValues ->
5         Column(
6             modifier = Modifier.fillMaxSize().padding(
paddingValues).verticalScroll(scrollState),
7             verticalArrangement = Arrangement.spacedBy(16.dp)
8         ) {
9             // Tahap 1: Memilih Tipe Bencana
10            Text("Disaster Type", style = MaterialTheme.typography
.titleMedium)
11            LazyRow(...) {
12                items(disasterTypes) { type ->
13                    DisasterTypeChip(
14                        type = type,
15                        isSelected = selectedType == type,
16                        onClick = { selectedType = type }
17                    )
18                }
19            }
20
21            // Tahap 2: Judul dan Deskripsi
22            DisasterTextField(value = title, onValueChange = {
title = it })
23            OutlinedTextField(
24                value = description,
25                onValueChange = { description = it },
26                label = { Text("Describe the disaster") }
27            )
28
29            // Tahap 3: Lokasi
30            Text("Location", style = MaterialTheme.typography.
titleMedium)
31            Row(...) {
32                Checkbox(
33                    checked = useCurrentLocation,
34                    onCheckedChange = { useCurrentLocation = it }
35                )
36                Text("Use my current location")
37            }
38        }
39    }
40 }
```

```

38     }
39 }
40 }

```

Kode 3.6: User Flow Pelaporan Bencana yang Intuitif

Perancangan responsive design memastikan aplikasi dapat beradaptasi dengan berbagai ukuran layar smartphone yang digunakan masyarakat. Implementasi accessibility guidelines diterapkan untuk memastikan aplikasi dapat digunakan oleh pengguna dengan keterbatasan fisik atau visual, termasuk dukungan untuk screen reader dan deskripsi konten yang komprehensif, sebagaimana diimplementasikan pada Kode 3.7.

```

1 // src/main/java/com/example/gmls/ui/components/
  AccessibilityComponents.kt
2 @Composable
3 fun AccessibleButton(
4     onClick: () -> Unit,
5     text: String,
6     modifier: Modifier = Modifier,
7     contentDescription: String? = null
8 ) {
9     Button(
10        onClick = onClick,
11        modifier = modifier.semantics {
12            // Menambahkan deskripsi konten untuk dibaca oleh
13            screen reader
14            contentDescription?.let { this.contentDescription = it
15        }
16        role = Role.Button
17    }
18    ) {
19        Text(text = text)
20    }
21 }
22 @Composable
23 fun AccessibleCard(
24     onClick: () -> Unit,
25     title: String,
26     subtitle: String? = null,
27     modifier: Modifier = Modifier
28 ) {
29     Card(

```

```

29     onClick = onClick,
30     modifier = modifier.semantics(mergeDescendants = true) {
31         contentDescription = buildString {
32             append(title)
33             subtitle?.let { append(", $it") }
34         }
35     }
36 ) {
37     // Card content
38 }
39 }

```

Kode 3.7: Implementasi Accessibility Guidelines

B Tahap 2: Setup dan Konfigurasi Project

B.1 Inisialisasi Project Android

Setup project Android dilakukan dengan konfigurasi yang optimal untuk *development* modern. Pembuatan *project Android* baru menggunakan *Kotlin* sebagai bahasa pemrograman utama, mengingat *Kotlin* telah menjadi bahasa resmi untuk pengembangan *Android* dan menawarkan sintaks yang lebih *concise* dan *safe* dibandingkan *Java*. Konfigurasi *Gradle build system* dilakukan dengan *dependency management* yang efisien, menggunakan *version catalogs* untuk mengelola versi *library* secara terpusat. *Setup version control* menggunakan *Git* dengan *branching strategy* yang terstruktur, menerapkan *GitFlow* untuk memisahkan *development*, *staging*, dan *production branches*.

Konfigurasi *Android manifest* mencakup *permissions* yang diperlukan seperti *location* untuk *GPS tracking*, *camera* untuk foto dokumentasi, dan *storage* untuk menyimpan data *offline*. Deklarasi *permissions* ini sangat penting untuk memastikan aplikasi dapat mengakses fitur-fitur *essential* yang dibutuhkan untuk *emergency response*, seperti yang terlihat pada Kode 3.8.

```

1 <!-- src/main/AndroidManifest.xml -->
2 <manifest xmlns:android="http://schemas.android.com/apk/res/
   android"
3     xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-permission android:name="android.permission.INTERNET" />
6     <uses-permission android:name="android.permission.
   POST_NOTIFICATIONS" />

```

```

7     <uses-permission android:name="android.permission.
ACCESS_NETWORK_STATE" />
8     <uses-permission android:name="android.permission.
ACCESS_FINE_LOCATION" />
9     <uses-permission android:name="android.permission.
ACCESS_COARSE_LOCATION" />
10    <uses-permission android:name="android.permission.CAMERA" />
11    <uses-permission android:name="android.permission.
READ_EXTERNAL_STORAGE" />
12    <uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE" />
13
14    <application
15        android:name=".GMLSApplication"
16        android:allowBackup="true"
17        android:icon="@mipmap/ic_launcher"
18        android:label="@string/app_name"
19        android:theme="@style/Theme.GMLS">
20
21        <activity
22            android:name=".MainActivity"
23            android:exported="true"
24            android:theme="@style/Theme.GMLS">
25            <intent-filter>
26                <action android:name="android.intent.action.MAIN"
/>
27                <category android:name="android.intent.category.
LAUNCHER" />
28            </intent-filter>
29        </activity>
30
31        <service
32            android:name=".data.service.DisasterMessagingService"
33            android:exported="false">
34            <intent-filter>
35                <action android:name="com.google.firebase.
MESSAGING_EVENT" />
36            </intent-filter>
37        </service>
38
39    </application>
40

```

```
41 </manifest>
```

Kode 3.8: Konfigurasi Android Manifest dengan Permissions

Setup proguard rules dilakukan untuk *code obfuscation* dan optimasi *APK size*, memastikan aplikasi final memiliki ukuran yang optimal dan kode yang terlindungi. Konfigurasi *build variants* juga diatur untuk mendukung *development*, *staging*, dan *production environments* dengan parameter yang berbeda.

B.2 Integrasi Firebase Services

Firebase dipilih sebagai *backend-as-a-service* untuk mempercepat *development* dan mengurangi kompleksitas infrastruktur *server*. *Setup Firebase project* dimulai dengan konfigurasi *google-services.json* yang berisi kredensial dan konfigurasi *project*. Integrasi beberapa layanan *Firebase* dilakukan secara terpadu dalam satu *service class* untuk memastikan konsistensi dan efisiensi operasi, sebagaimana diimplementasikan pada Kode 3.9.

```
1 // src/main/java/com/example/gmls/data/remote/FirebaseService.kt
2 suspend fun register(userData: RegistrationData): Result<
  FirebaseUser> {
3     return executeWithRetry {
4         // 1. Menggunakan Firebase Authentication
5         val authResult = auth.createUserWithEmailAndPassword(
6             userData.email,
7             userData.password
8         ).await()
9         val user = authResult.user ?: throw Exception("User
  creation failed")
10
11        // 2. Menggunakan Firebase Firestore
12        val userMap = mapOf(
13            "fullName" to userData.fullName,
14            "email" to userData.email,
15            "phoneNumber" to userData.phoneNumber,
16            "role" to "USER",
17            "isVerified" to false,
18            "isActive" to true,
19            "createdAt" to FieldValue.serverTimestamp()
20        )
21        firestore.collection("users").document(user.uid).set(
  userMap).await()
22
```

```

23     user
24     }
25 }
26
27 private suspend fun uploadImages(images: List<Uri>): List<String>
28 {
29     return images.map { imageUrl ->
30         executeWithRetry {
31             val fileName = "${UUID.randomUUID()}.jpg"
32             // 3. Menggunakan Firebase Storage
33             val storageRef = storage.reference.child("
34                 disaster_images/$fileName")
35             storageRef.putFile(imageUri).await()
36             storageRef.downloadUrl.await().toString()
37         }.getOrThrow()
38     }
39 }

```

Kode 3.9: Integrasi Komprehensif Firebase Services

Integrasi *Firebase Authentication* dilakukan untuk menyediakan sistem *login/register* yang aman dengan berbagai metode autentikasi. Konfigurasi *Firebase Firestore* sebagai *database real-time* memungkinkan sinkronisasi data secara otomatis antar *device* dan menyediakan *offline support* yang *essential* untuk aplikasi *emergency response*.

Setup Firebase Cloud Messaging diimplementasikan untuk *push notifications*, memungkinkan pengiriman *alert* darurat kepada pengguna secara *real-time*. Implementasi *FCM service* dilakukan dengan *custom message handling* untuk berbagai jenis notifikasi *emergency*, seperti yang ditunjukkan pada Kode 3.10.

```

1 // src/main/java/com/example/gmls/data/service/
2 // DisasterMessagingService.kt
3 @AndroidEntryPoint
4 class DisasterMessagingService : FirebaseMessagingService() {
5     @Inject
6     lateinit var notificationManager: NotificationManager
7
8     override fun onMessageReceived(remoteMessage: RemoteMessage) {
9         super.onMessageReceived(remoteMessage)
10
11         remoteMessage.notification?.let { notification ->

```

```

12         sendNotification(
13             title = notification.title ?: getString(R.string.
app_name),
14             message = notification.body ?: "",
15             data = remoteMessage.data
16         )
17     }
18
19     // Handle data payload untuk custom actions
20     if (remoteMessage.data.isNotEmpty()) {
21         handleDataPayload(remoteMessage.data)
22     }
23 }
24
25 private fun sendNotification(title: String, message: String,
data: Map<String, String>) {
26     val intent = Intent(this, MainActivity::class.java).apply
{
27         addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
28         // Add data untuk deep linking
29         data.forEach { (key, value) ->
30             putExtra(key, value)
31         }
32     }
33
34     val pendingIntent = PendingIntent.getActivity(
35         this, 0, intent, PendingIntent.FLAG_IMMUTABLE
36     )
37
38     val notification = NotificationCompat.Builder(this,
CHANNEL_ID)
39         .setContentTitle(title)
40         .setContentText(message)
41         .setSmallIcon(R.drawable.ic_notification)
42         .setAutoCancel(true)
43         .setContentIntent(pendingIntent)
44         .setPriority(NotificationCompat.PRIORITY_HIGH)
45         .build()
46
47     notificationManager.notify(System.currentTimeMillis().
toInt(), notification)
48 }

```

49 }

Kode 3.10: Firebase Cloud Messaging Implementation

Integrasi *Firebase Storage* menyediakan solusi penyimpanan foto dokumentasi bencana dengan *automatic scaling* dan *CDN global*. Konfigurasi *Firebase Analytics* ditambahkan untuk *monitoring* penggunaan aplikasi dan memahami *behavior* pengguna untuk *improvement* berkelanjutan.

B.3 Implementasi Dependency Injection

Hilt dipilih sebagai *dependency injection framework* untuk *Android* karena integrasinya yang *seamless* dengan *Android Architecture Components*. *Setup Hilt* dimulai dengan konfigurasi *modules* dan *components* yang mendefinisikan bagaimana *dependencies* disediakan di seluruh aplikasi. Implementasi *Hilt modules* dilakukan untuk menyediakan *instances* dari berbagai komponen aplikasi dengan *scope* yang tepat, sebagaimana terlihat pada Kode 3.11.

```
1 // src/main/java/com/example/gmls/di/RepositoryModule.kt
2 @Module
3 @InstallIn (SingletonComponent::class)
4 object RepositoryModule {
5
6     @Provides
7     @Singleton
8     fun provideUserRepository(
9         firebaseService: FirebaseService,
10        userMapper: UserFirebaseMapper
11    ): UserRepository {
12        return UserRepositoryImpl(firebaseService, userMapper)
13    }
14
15    @Provides
16    @Singleton
17    fun provideDisasterRepository(
18        firebaseService: FirebaseService,
19        disasterMapper: DisasterFirebaseMapper
20    ): DisasterRepository {
21        return DisasterRepositoryImpl(firebaseService,
22        disasterMapper)
23    }
24
25    @Provides
```

```

25     @Singleton
26     fun provideAdminRepository(
27         firebaseService: FirebaseService,
28         userRepository: UserRepository,
29         disasterRepository: DisasterRepository
30     ): AdminRepository {
31         return AdminRepositoryImpl(firebaseService, userRepository
32     , disasterRepository)
33     }
34 }
35 @Module
36 @InstallIn(SingletonComponent::class)
37 object ServiceModule {
38
39     @Provides
40     @Singleton
41     fun provideLocationService(
42         @ApplicationContext context: Context
43     ): LocationService {
44         return LocationServiceImpl(context)
45     }
46
47     @Provides
48     @Singleton
49     fun provideNotificationManager(
50         @ApplicationContext context: Context
51     ): NotificationManager {
52         return context.getSystemService(Context.
53     NOTIFICATION_SERVICE) as NotificationManager
54 }

```

Kode 3.11: Setup Hilt Modules untuk Dependency Injection

Pembuatan repository pattern dilakukan untuk abstraksi data layer, memisahkan sumber data dari business logic dan memudahkan testing dengan mock objects. Implementasi ViewModelFactory dengan Hilt injection memastikan ViewModels mendapatkan dependencies yang diperlukan secara otomatis tanpa manual factory creation, seperti yang diimplementasikan pada Kode 3.12.

```

1 // src/main/java/com/example/gmls/ui/viewmodels/AdminViewModel.kt
2 @HiltViewModel
3 class AdminViewModel @Inject constructor(

```

```

4     private val userRepository: UserRepository,
5     private val disasterRepository: DisasterRepository,
6     private val adminRepository: AdminRepository
7 ) : ViewModel() {
8
9     private val _adminState = MutableStateFlow(AdminState())
10    val adminState: StateFlow<AdminState> = _adminState.
11    asStateFlow()
12
13    fun loadAllData() {
14        viewModelScope.launch {
15            _adminState.value = _adminState.value.copy(isLoading =
16            true, error = null)
17            try {
18                val users = userRepository.getAllUsers()
19                val disasters = disasterRepository.getAllDisasters
20                ()
21                val userAnalytics = calculateUserAnalytics(users)
22                val disasterAnalytics = calculateDisasterAnalytics
23                (disasters)
24
25                _adminState.value = _adminState.value.copy(
26                users = users,
27                disasters = disasters,
28                userAnalytics = userAnalytics,
29                disasterAnalytics = disasterAnalytics,
30                isLoading = false
31            )
32            } catch (e: Exception) {
33                _adminState.value = _adminState.value.copy(
34                error = e.message,
35                isLoading = false
36            )
37        }
38    }
39 }

```

Kode 3.12: Injeksi Dependencies ke ViewModel dengan Hilt

Konfigurasi singleton instances untuk services yang shared seperti LocationService dan NotificationService memastikan efisiensi resource dan konsistensi state. Setup testing configuration dengan Hilt testing framework

memungkinkan injection of test doubles untuk unit testing dan integration testing yang reliable, sangat penting untuk memastikan quality aplikasi emergency response.

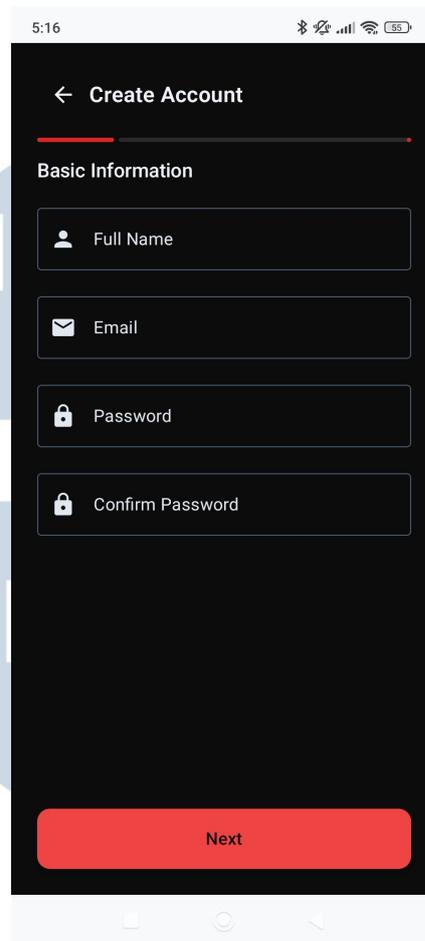
C Tahap 3: Pengembangan Fitur Autentikasi dan Onboarding

C.1 Implementasi Splash Screen dan Onboarding

Pengembangan layar awal aplikasi dirancang untuk memberikan first impression yang baik dan memperkenalkan aplikasi kepada pengguna baru. Implementasi SplashScreen dilakukan dengan animasi logo GMLS dan loading indicator yang memberikan feedback visual selama proses inialisasi aplikasi. Pengembangan OnboardingScreen menggunakan multiple pages untuk menjelaskan fitur-fitur utama aplikasi secara bertahap, membantu pengguna memahami cara menggunakan aplikasi dengan efektif. Implementasi OnboardingPreferenceManager memastikan onboarding hanya ditampilkan sekali untuk setiap pengguna, meningkatkan user experience untuk pengguna yang sudah familiar. Desain *smooth transitions* antar layar *onboarding* menggunakan *ViewPager2* memberikan navigasi yang *fluid* dan modern. Implementasi *skip functionality* memungkinkan pengguna yang sudah *familiar* untuk langsung mengakses aplikasi tanpa melalui seluruh proses onboarding.

C.2 Sistem Autentikasi Pengguna

Pengembangan sistem login dan registrasi dirancang dengan fokus pada keamanan dan user-friendliness, mengingat pentingnya verifikasi identitas dalam aplikasi emergency response. Implementasi LoginScreen dilengkapi dengan validasi input email dan password yang comprehensive, memastikan format data yang benar sebelum dikirim ke server. Pengembangan multi-step RegistrationScreen menggunakan progressive disclosure untuk mengumpulkan informasi pengguna secara bertahap, mengurangi cognitive load dan meningkatkan completion rate. Implementasi ValidationUtils menyediakan validasi format email, kekuatan password, dan NIK sesuai standar Indonesia untuk memastikan data yang akurat dan valid.

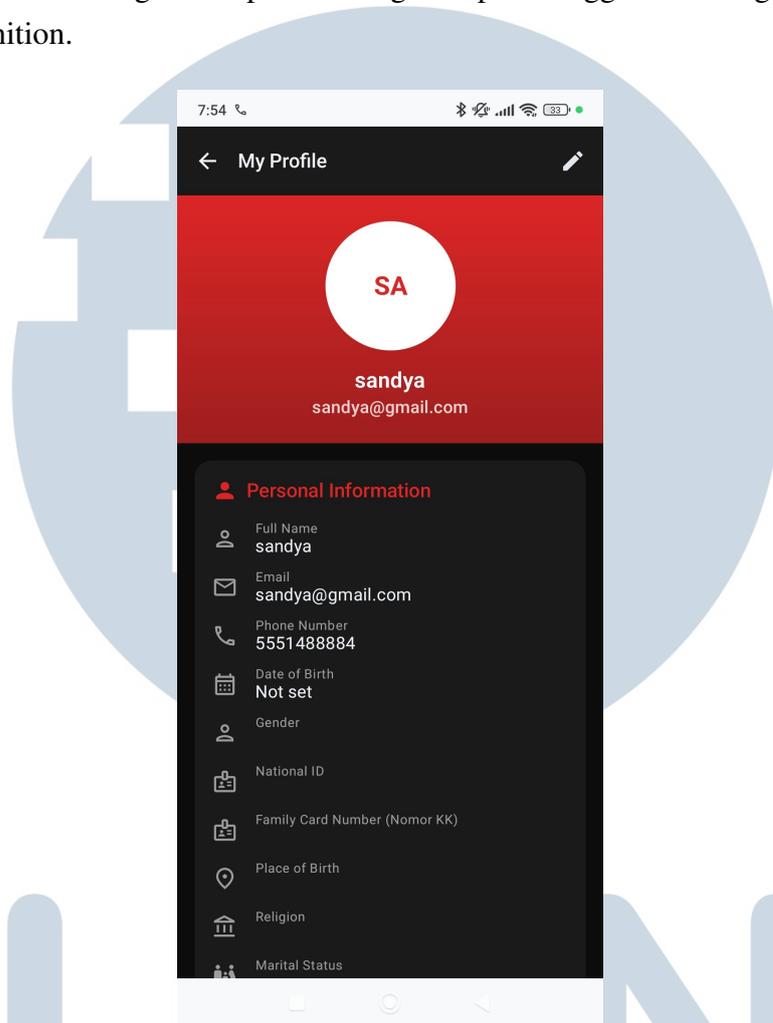


Gambar 3.1. Layar Registrasi Pengguna dengan Progressive Disclosure

Gambar 3.1 menunjukkan implementasi layar registrasi yang menggunakan pendekatan progressive disclosure untuk mengumpulkan informasi pengguna secara bertahap. Progress indicator di bagian atas memberikan feedback visual tentang tahapan registrasi yang sedang berlangsung. Formulir registrasi menampilkan *field-field essential* dengan ikon yang intuitif untuk memudahkan identifikasi. Validasi *real-time* diterapkan untuk memastikan *format* data yang benar sebelum pengguna dapat melanjutkan ke *tahap* berikutnya. Desain menggunakan color scheme yang konsisten dengan branding GMLS dan mengikuti prinsip accessibility untuk memastikan keterbacaan yang optimal.

Integrasi dengan Firebase Authentication dilakukan untuk proses autentikasi yang aman dan reliable, memanfaatkan infrastruktur Google yang telah teruji untuk menangani jutaan pengguna. Pengembangan forgot password functionality dengan email reset memberikan opsi pemulihan akun yang mudah diakses bagi pengguna yang lupa kredensial mereka. Implementasi biometric authentication sebagai

opsi login alternatif meningkatkan convenience dan security, memungkinkan pengguna untuk mengakses aplikasi dengan cepat menggunakan fingerprint atau face recognition.



Gambar 3.2. Layar Pengaturan Profil Pengguna dengan *Interface* Komprehensif

Gambar 3.2 mendemonstrasikan implementasi layar pengaturan profil pengguna dengan interface yang komprehensif dan user-friendly. Header aplikasi menampilkan logo GMLS dengan elemen visual yang distinctive dan banner organisasi yang memberikan identitas visual yang jelas. Formulir login menggunakan desain minimalis dengan field-field essential yang dilengkapi ikon yang intuitif untuk memudahkan identifikasi. Link pemulihan *password* ditempatkan strategis dengan *styling* yang kontras untuk *visibility* yang optimal. Tombol *login* menggunakan *styling* yang *prominent* untuk memberikan *emphasis* pada aksi utama. Keseluruhan desain menggunakan dark theme yang memberikan kesan profesional dan modern, sesuai dengan konteks aplikasi emergency response.

C.3 Manajemen Data Pengguna

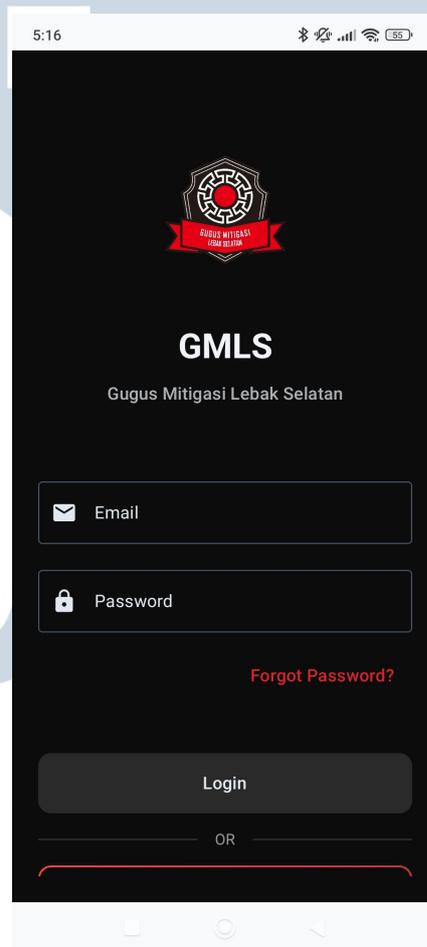
Sistem untuk mengelola data pengguna dirancang secara komprehensif sesuai dengan standar dan kebutuhan Indonesia, mengingat aplikasi ini akan digunakan untuk keperluan emergency response yang memerlukan data akurat. Implementasi data model User mencakup field lengkap seperti NIK, Nomor KK, dan alamat detail yang sesuai dengan format administrasi Indonesia.

Logika dan data spesifik domain untuk alamat Indonesia diimplementasikan melalui `AccurateIndonesianAddressData` yang berisi data akurat mengenai wilayah administratif Indonesia termasuk provinsi, kabupaten, dan kecamatan. File ini menyediakan data alamat Indonesia dalam bentuk Map dan menyediakan fungsi lookup seperti `getRegencies` dan `getDistricts` untuk mengisi dropdown secara dinamis, menunjukkan kemampuan untuk mengelola dan menyajikan data hierarkis yang kompleks, sebagaimana terlihat pada Kode 3.13.

```
1 // src/main/java/com/example/gmls/ui/components/  
  AccurateIndonesianAddressData.kt  
2 object AccurateIndonesianAddressData {  
3  
4     val provinces = listOf(  
5         "Aceh", "Sumatera Utara", "Jawa Barat", "Banten", // ...  
6         38 provinsi  
7     )  
8  
9     val regencies = mapOf(  
10        "Banten" to listOf(  
11            "", "Kota Serang", "Kota Tangerang",  
12            "Kabupaten Lebak", "Kabupaten Pandeglang",  
13        ),  
14        "Jawa Barat" to listOf(  
15            "", "Kota Bandung", "Kota Bekasi",  
16            // ... data kabupaten/kota lainnya  
17        )  
18  
19     fun getRegencies(province: String): List<String> {  
20         return regencies[province] ?: listOf("")  
21     }  
22  
23     // ... fungsi lainnya untuk kecamatan, kelurahan, dll.
```

Kode 3.13: Data Alamat Indonesia yang Akurat

Integrasi `AccurateIndonesianAddressData` menyediakan dropdown alamat terstruktur untuk Provinsi, Kabupaten/Kota, Kecamatan, dan Kelurahan/Desa, memastikan konsistensi dan akurasi data alamat. Pengembangan `ProfileScreen` memungkinkan pengguna untuk viewing dan editing data pribadi mereka dengan interface yang user-friendly dan validasi yang ketat. Implementasi image picker untuk foto profil dilengkapi dengan compression otomatis untuk mengoptimalkan storage usage tanpa mengorbankan kualitas visual. Setup data synchronization antara local storage dan Firebase Firestore memastikan data pengguna selalu tersinkronisasi dan dapat diakses offline ketika diperlukan.



Gambar 3.3. Layar Login dengan Branding GMLS yang Kuat

Gambar 3.3 menampilkan implementasi layar login yang menampilkan branding GMLS secara prominent dengan logo organisasi yang distinctive. Banner

organisasi menegaskan identitas dan memberikan konteks yang jelas tentang aplikasi. Formulir login yang sederhana menampilkan field-field essential dengan ikon yang konsisten untuk menjaga user experience yang familiar. Link pemulihan password memberikan opsi recovery yang mudah diakses, sementara tombol login menggunakan color scheme yang kontras untuk visibility yang optimal. Desain memprioritaskan kemudahan akses dengan meminimalkan hambatan untuk pengguna yang ingin segera menggunakan aplikasi dalam situasi darurat.

D Tahap 4: Pengembangan Fitur Inti Aplikasi

D.1 Dashboard dan Navigasi Utama

Pengembangan pusat informasi dan navigasi aplikasi dirancang sebagai hub utama yang memberikan akses cepat ke semua fitur penting dan informasi terkini tentang situasi bencana. Implementasi DashboardScreen menggunakan card-based layout untuk menyajikan informasi ringkas yang mudah dipahami dengan scanning cepat, sangat penting dalam situasi darurat. Pengembangan LocationTrackingCard dilengkapi dengan toggle yang memungkinkan pengguna mengaktifkan atau menonaktifkan real-time location sharing, memberikan kontrol privacy kepada pengguna sambil memungkinkan monitoring oleh admin saat diperlukan.

Komponen UI kustom dan reaktif dikembangkan untuk menampilkan data analitik dengan AdminAnalyticsCard yang dirancang khusus sebagai komponen Composable yang dapat digunakan kembali. Komponen ini menampilkan data secara visual dengan baik termasuk progress bar dan ikon yang dinamis, dilengkapi dengan animasi yang halus menggunakan animateFloatAsState untuk memberikan feedback visual yang engaging kepada pengguna, seperti yang ditunjukkan pada Kode 3.14.

```
1 // src/main/java/com/example/gmls/ui/components/AdminAnalyticsCard
  .kt
2 @Composable
3 fun AdminAnalyticsCard(
4     title: String,
5     value: String,
6     subtitle: String? = null,
7     icon: ImageVector,
8     iconColor: Color = MaterialTheme.colorScheme.primary,
9     progress: Float? = null,
```

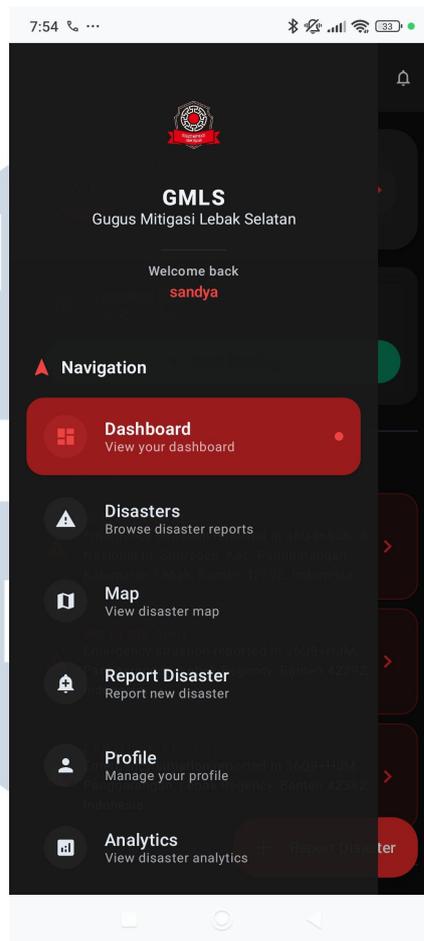
```

10 ) {
11     var animationPlayed by remember { mutableStateOf(false) }
12     val animatedProgress by animateFloatAsState(
13         targetValue = if (animationPlayed && progress != null)
progress else 0f,
14         animationSpec = tween(durationMillis = 1000),
15         label = "progress_animation"
16     )
17
18     LaunchedEffect(Unit) {
19         animationPlayed = true
20     }
21
22     Card(...) {
23         Column(modifier = Modifier.padding(16.dp)) {
24             if (progress != null) {
25                 Spacer(modifier = Modifier.height(16.dp))
26                 Box(...) {
27                     CircularProgressIndicator(
28                         progress = { animatedProgress },
29                     )
30                     Text(text = "${(animatedProgress * 100).toInt
31                         ()}%")
32                 }
33             }
34         }
35     }

```

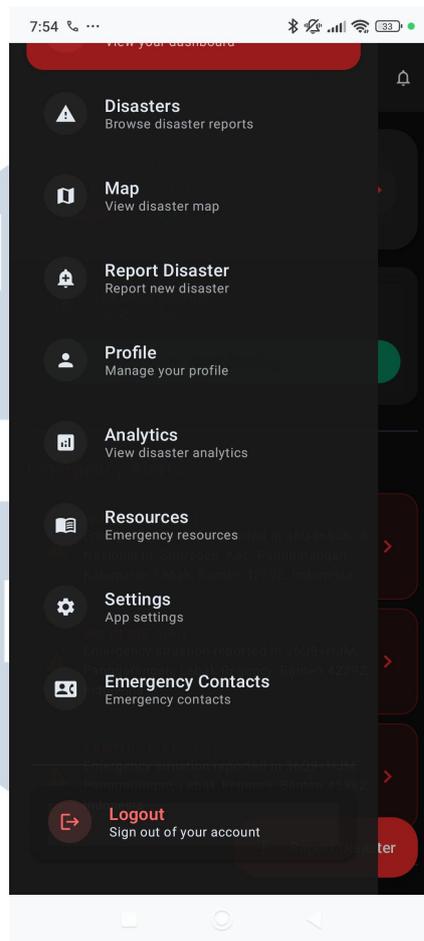
Kode 3.14: Komponen UI Kustom dengan Animasi

Implementasi emergency alerts section dirancang khusus untuk menampilkan bencana dengan severity tinggi menggunakan color coding yang mencolok, memastikan informasi kritis mendapat perhatian segera. Pengembangan recent disasters list menggunakan infinite scrolling untuk performa optimal, memungkinkan pengguna melihat laporan terbaru tanpa menunggu loading yang lama. Setup bottom navigation menggunakan ikon intuitif dan state management yang robust untuk memastikan navigasi yang smooth dan konsisten di seluruh aplikasi.



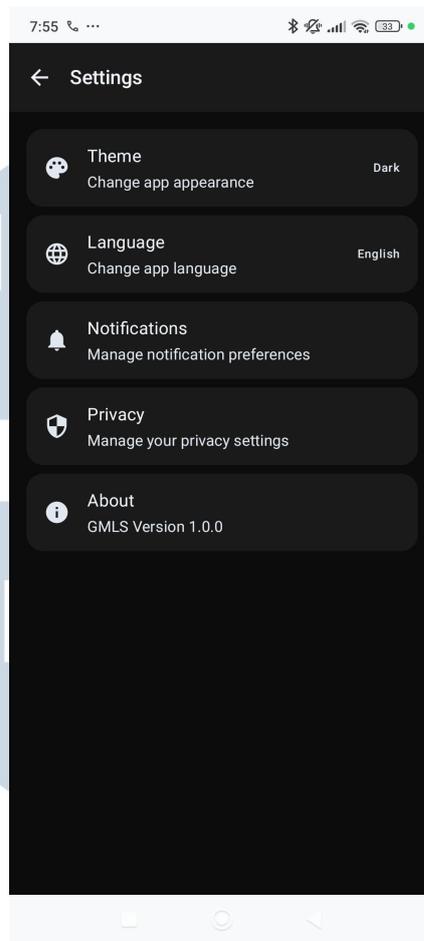
Gambar 3.4. Menu Navigasi Utama dengan Akses Komprehensif ke Semua Fitur Aplikasi

Gambar 3.4 mendemonstrasikan implementasi menu navigasi utama yang menyediakan akses komprehensif ke semua fitur aplikasi GMLS dengan organisasi yang logis dan hierarkis. Menu dimulai dengan aksi utama yang mendapat emphasis visual khusus. Section navigasi utama mencakup fitur-fitur inti seperti disaster management, peta, pelaporan, dan profil pengguna dengan ikon yang intuitif. Section fitur lanjutan menyediakan akses ke analytics, resources, pengaturan, dan kontak darurat. Tombol logout ditempatkan di bagian bawah dengan styling yang berbeda untuk memberikan emphasis pada aksi yang signifikan. Setiap menu item menggunakan ikon yang intuitif dan deskripsi yang jelas, memastikan navigasi yang mudah dipahami bahkan dalam situasi darurat.



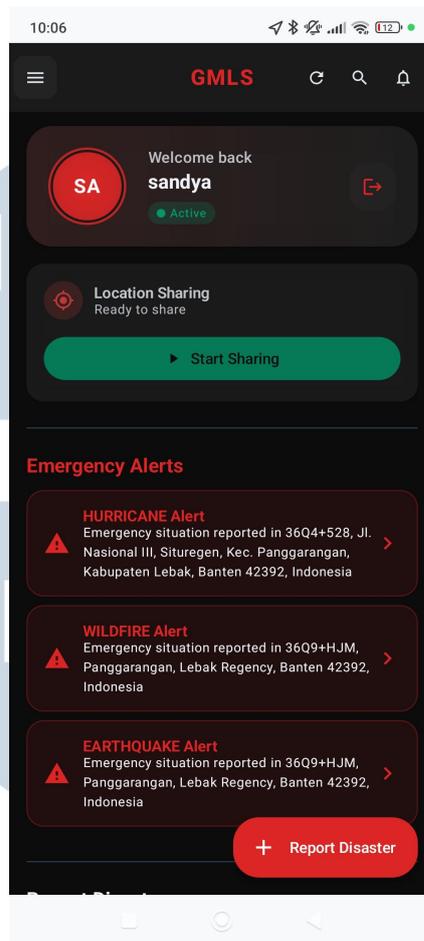
Gambar 3.5. Menu Navigasi dengan *Header Personal* dan *Dashboard yang Highlighted*

Gambar 3.5 menunjukkan varian menu navigasi yang menampilkan personalisasi dan hierarchy yang lebih jelas dengan header yang prominent. Logo GMLS dan banner organisasi memberikan branding yang konsisten di bagian atas. Welcome message dengan nama pengguna menciptakan pengalaman yang personal dan ramah. Section navigasi memberikan kategori yang jelas untuk menu items. Item dashboard mendapat treatment visual khusus, mengindikasikan bahwa ini adalah halaman aktif atau aksi utama yang direkomendasikan. Menu items lainnya ditampilkan dengan styling yang konsisten menggunakan ikon yang intuitif dan deskripsi yang jelas. Hierarchy visual yang jelas dengan dashboard sebagai focal point membantu pengguna memahami struktur navigasi dan mengakses fitur utama dengan cepat.



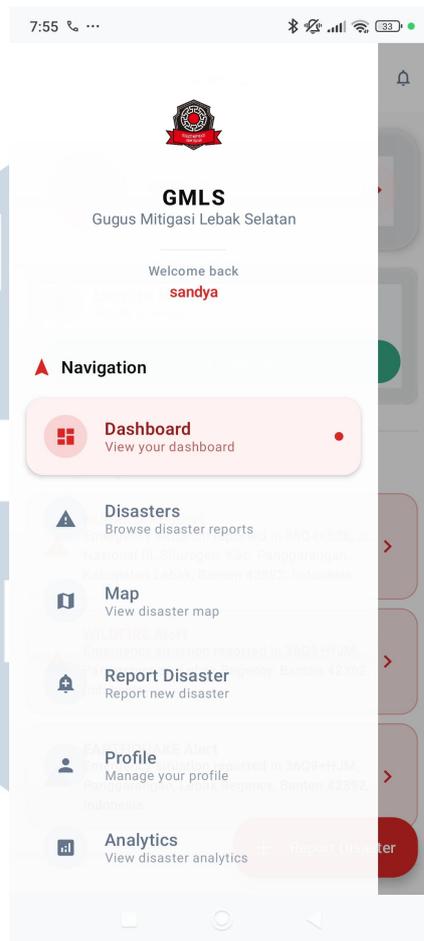
Gambar 3.6. Layar Pengaturan Aplikasi dengan Opsi Konfigurasi Lengkap

Gambar 3.6 mendemonstrasikan implementasi layar pengaturan aplikasi dengan berbagai opsi konfigurasi yang lengkap untuk personalisasi pengguna. Header menampilkan logo GMLS yang iconic dan banner organisasi untuk branding yang konsisten. Welcome message dengan nama pengguna menciptakan pengalaman yang personal dan engaging. Section navigasi memberikan kategorisasi yang jelas untuk menu items. Item dashboard mendapat treatment visual khusus dengan styling yang prominent dan indicator visual, mengindikasikan bahwa ini adalah halaman yang sedang aktif atau memiliki status khusus. Menu items lainnya ditampilkan dengan styling yang konsisten menggunakan ikon yang intuitif untuk berbagai fungsi seperti disaster management, peta, pelaporan, profil, dan analytics. Setiap menu item menggunakan ikon yang intuitif dan deskripsi yang jelas, memastikan navigasi yang mudah dipahami dan diakses dengan cepat.



Gambar 3.7. *Dashboard* Utama dengan Informasi Terpusat dan Navigasi Intuitif

Gambar 3.7 menunjukkan implementasi dashboard utama yang berfungsi sebagai pusat informasi dan navigasi aplikasi GMLS. Header dashboard menampilkan sambutan personal dengan avatar pengguna, menciptakan pengalaman yang personal dan ramah. Kartu location tracking dengan toggle switch memungkinkan pengguna mengontrol pelacakan lokasi real-time mereka, fitur yang sangat penting untuk monitoring saat terjadi situasi darurat. Section emergency alerts menampilkan peringatan bencana dengan tingkat keparahan tinggi menggunakan color coding untuk memastikan informasi kritis mendapat perhatian segera. Section recent disasters menyajikan daftar laporan bencana terbaru dalam format card yang dapat di-scroll, dengan setiap card menampilkan informasi essential tentang kejadian. Bottom navigation bar menggunakan ikon yang intuitif dengan highlight untuk halaman aktif, memastikan navigasi yang mudah dan konsisten.



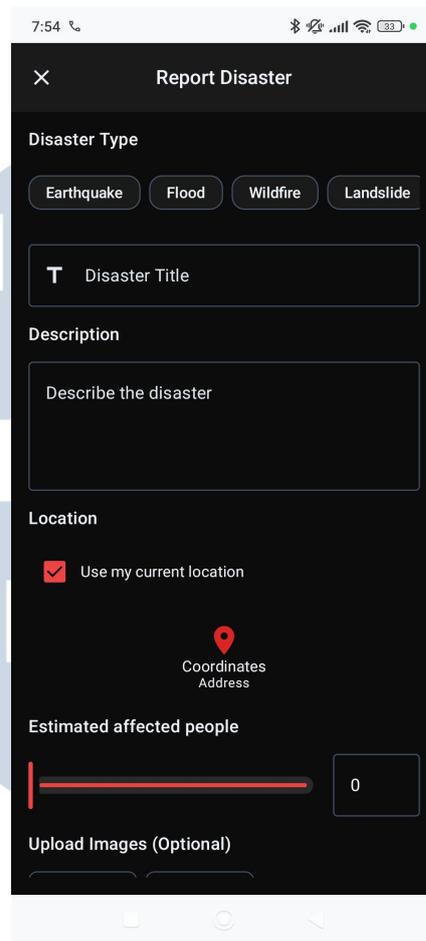
Gambar 3.8. *Dashboard dengan Emergency Alerts dan Logout Confirmation Dialog*

Gambar 3.8 menunjukkan implementasi dashboard dalam dark theme dengan fokus pada emergency alerts dan sistem logout confirmation yang user-friendly. Header aplikasi menampilkan branding dengan styling yang prominent dan ikon-ikon untuk akses cepat ke berbagai fungsi. User profile section menampilkan avatar, welcome message, dan status indicator yang memberikan feedback visual tentang status koneksi pengguna. Section location sharing memberikan informasi tentang status pelacakan lokasi. Modal dialog logout confirmation muncul di tengah layar dengan background overlay, menampilkan pesan konfirmasi dengan dua tombol aksi yang menggunakan color coding untuk membedakan aksi destructive. Di background terlihat emergency alerts untuk berbagai jenis bencana dengan informasi lokasi, menunjukkan sistem notifikasi real-time yang berfungsi. Floating action button untuk pelaporan bencana tetap accessible untuk akses cepat. Design menggunakan dark theme yang konsisten dengan contrast yang baik untuk readability dalam berbagai kondisi pencahayaan.

D.2 Sistem Pelaporan Bencana

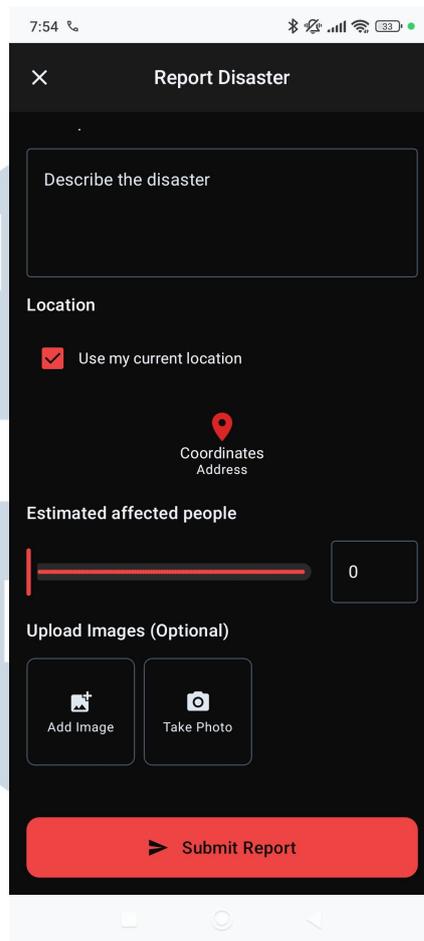
Fitur inti untuk melaporkan kejadian bencana dirancang dengan fokus pada kemudahan dan kecepatan akses, mengingat situasi darurat memerlukan pelaporan yang efisien. Implementasi ReportDisasterScreen menggunakan form input yang comprehensive namun tetap streamlined, mengumpulkan informasi essential seperti jenis bencana, lokasi, tingkat kerusakan, dan deskripsi kejadian. Pengembangan disaster type selection disesuaikan dengan kategori bencana yang umum terjadi di Indonesia seperti banjir, gempa bumi, tanah longsor, kebakaran, dan angin kencang. Integrasi GPS service memungkinkan automatic location detection untuk akurasi maksimal, dengan fallback manual untuk situasi di mana GPS tidak tersedia atau tidak akurat. Implementasi LocationPickerScreen menggunakan Google Maps untuk manual location selection, sangat berguna ketika pengguna melaporkan kejadian dari jarak jauh atau untuk lokasi yang sulit dideteksi GPS. Pengembangan image capture dan gallery picker mendukung multiple image support, memungkinkan pengguna melampirkan beberapa foto dokumentasi untuk memberikan gambaran yang lebih komprehensif. Implementasi image compression dan upload progress indicator memastikan efisiensi bandwidth sambil memberikan feedback visual kepada pengguna. Setup form validation dan error handling yang robust mencegah submission data yang tidak lengkap atau invalid.





Gambar 3.9. Formulir Pelaporan Bencana Tahap Pertama dengan *Interface Streamlined*

Gambar 3.9 menunjukkan implementasi formulir pelaporan bencana tahap pertama yang dirancang dengan interface yang streamlined dan user-friendly untuk situasi darurat. Header dengan tombol close memungkinkan pengguna keluar dari proses pelaporan jika diperlukan. Text area untuk deskripsi bencana memberikan ruang yang cukup untuk pengguna menjelaskan kejadian secara detail. Section location menyediakan checkbox untuk menggunakan lokasi saat ini yang sudah tercentang secara default, memungkinkan aplikasi menggunakan GPS untuk mendeteksi lokasi otomatis dengan visual indicator yang jelas. Field untuk estimasi jumlah orang terdampak menggunakan slider interface yang intuitif dengan input numerik. Section upload images menyediakan dua opsi untuk dokumentasi visual: memilih foto dari galeri dan mengambil foto langsung menggunakan kamera, keduanya dengan ikon yang jelas dan mudah dipahami. Tombol submit report di bagian bawah menggunakan styling yang mencolok untuk memberikan emphasis pada aksi final mengirim laporan.



Gambar 3.10. Varian Formulir Pelaporan dengan Kategorisasi Jenis Bencana yang Spesifik

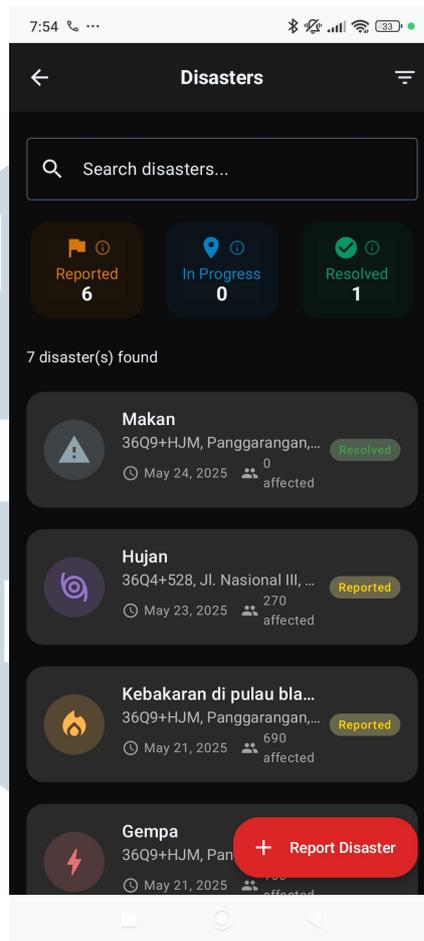
Gambar 3.10 mendemonstrasikan varian formulir pelaporan bencana yang menyediakan opsi kategorisasi jenis bencana yang lebih spesifik dan terstruktur. Header dengan tombol close konsisten dengan desain sebelumnya untuk navigasi yang familiar. Section disaster type menampilkan kategori utama bencana dalam bentuk pill buttons, memungkinkan pengguna memilih jenis bencana dengan cepat dan akurat sesuai dengan kondisi yang terjadi. Field disaster title memberikan ruang untuk pengguna memberikan judul singkat yang deskriptif untuk laporan bencana. Text area description menyediakan ruang yang luas untuk penjelasan detail kejadian, memungkinkan pelapor memberikan informasi komprehensif tentang situasi yang terjadi. Section location tetap mempertahankan checkbox untuk menggunakan lokasi saat ini yang tercentang secara default dengan visual indicator yang jelas, memastikan konsistensi dalam pengumpulan data lokasi. Field untuk estimasi jumlah orang terdampak menggunakan slider interface yang intuitif untuk memperkirakan dampak bencana terhadap populasi. Section upload images

di bagian bawah mengindikasikan adanya opsi dokumentasi visual yang konsisten dengan desain formulir sebelumnya.

D.3 Sistem Monitoring dan Daftar Bencana

Fitur untuk melihat dan memonitor laporan bencana dirancang untuk memberikan overview yang komprehensif tentang situasi bencana di wilayah Lebak Selatan. Implementasi DisasterListScreen menggunakan RecyclerView dengan custom adapter yang dioptimalkan untuk menampilkan data dalam jumlah besar dengan performa yang smooth. Pengembangan search functionality dengan real-time filtering memungkinkan pengguna mencari laporan spesifik berdasarkan kata kunci seperti lokasi atau deskripsi kejadian. Implementasi filter berdasarkan jenis bencana, severity level, dan rentang waktu memberikan fleksibilitas kepada pengguna untuk menyaring informasi sesuai kebutuhan mereka. Pengembangan DisasterDetailScreen menyajikan informasi lengkap tentang setiap laporan bencana termasuk foto dokumentasi, lokasi detail, timestamp, dan status penanganan. Setup pull-to-refresh dan pagination memastikan performa optimal bahkan dengan data yang besar, memberikan user experience yang responsive. Implementasi bookmark functionality memungkinkan pengguna menyimpan laporan penting untuk referensi cepat di masa mendatang.

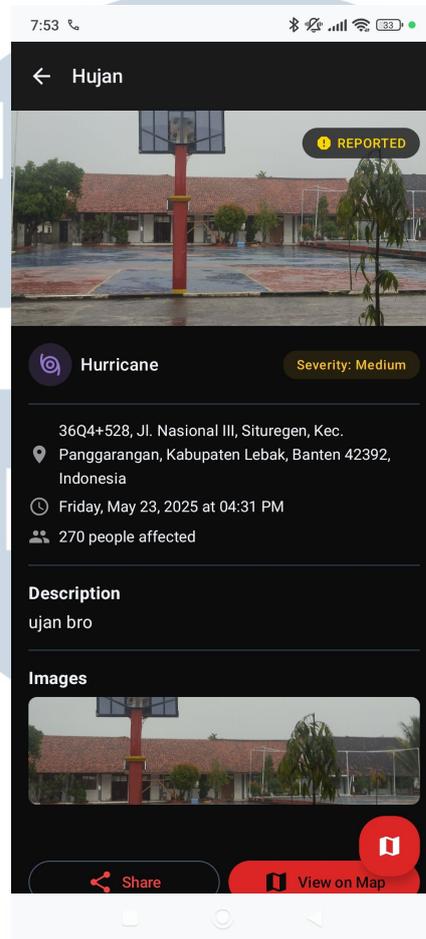




Gambar 3.11. Daftar Laporan Bencana dengan *Filtering* dan *Status Tracking* yang Komprehensif

Gambar 3.11 mendemonstrasikan implementasi layar daftar bencana yang menyediakan overview komprehensif tentang semua laporan bencana dengan sistem filtering dan status tracking yang efektif. Header dengan tombol back dan filter menu memberikan navigasi yang jelas dan akses ke opsi filtering lanjutan. Search bar memungkinkan pengguna mencari laporan spesifik dengan cepat berdasarkan kata kunci. Dashboard statistik menampilkan kategori status dengan color coding yang intuitif, memberikan overview cepat tentang status penanganan bencana berdasarkan jumlah laporan yang dilaporkan, sedang diproses, dan telah diselesaikan. Indikator total memberikan informasi jumlah laporan yang tersedia. Daftar laporan ditampilkan dalam format card dengan informasi essential seperti nama bencana, lokasi, timestamp kejadian, jumlah orang terdampak, dan status badge dengan color coding yang konsisten. Setiap card menggunakan ikon yang spesifik untuk jenis bencana untuk identifikasi visual yang cepat. Floating

action button di pojok kanan bawah memberikan akses cepat untuk melaporkan bencana baru.



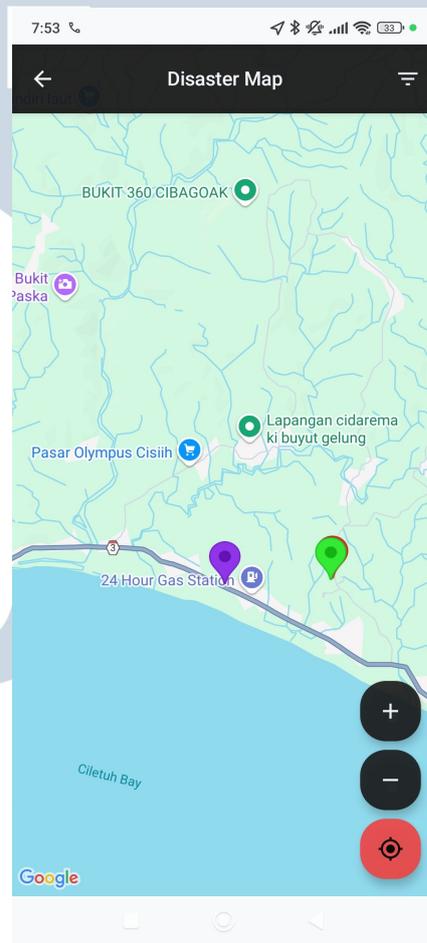
Gambar 3.12. Layar Detail Laporan Bencana dengan Informasi Komprehensif

Gambar 3.12 mendemonstrasikan layar detail bencana yang menampilkan informasi komprehensif tentang laporan bencana dengan layout yang terorganisir dengan baik. Header menampilkan foto dokumentasi bencana yang menunjukkan kondisi lapangan, dilengkapi dengan overlay status yang memberikan indikasi visual cepat tentang status laporan saat ini. Judul bencana dengan ikon dan severity level memberikan informasi cepat tentang jenis dan tingkat keparahan bencana. Informasi bencana disajikan secara terstruktur dengan ikon yang intuitif untuk setiap jenis data seperti lokasi, timestamp, dan jumlah orang yang terdampak. Section description menampilkan informasi dari pelapor, sementara section images menampilkan thumbnail foto dokumentasi yang dapat di-expand untuk melihat detail lebih lanjut. Tombol aksi di bagian bawah memberikan opsi untuk berbagi informasi atau melihat lokasi pada peta interaktif.

E Tahap 5: Pengembangan Fitur Lanjutan

E.1 Implementasi Peta Interaktif

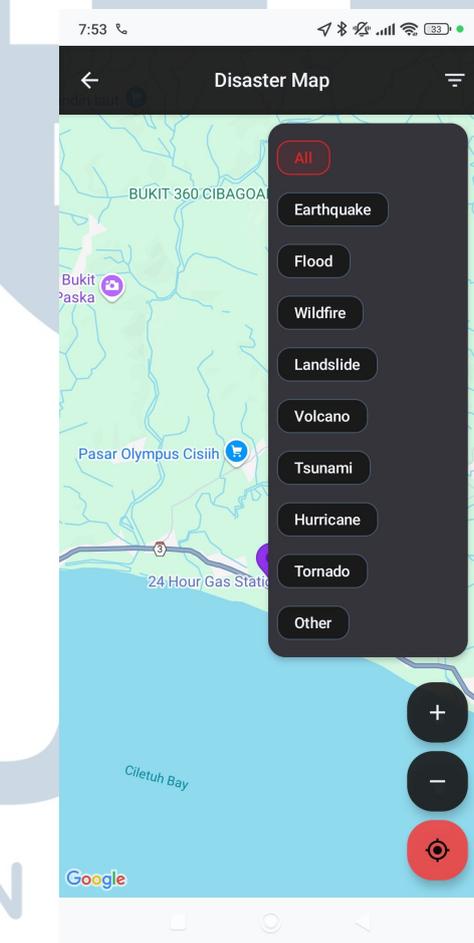
Pengembangan peta interaktif menggunakan Google Maps SDK dirancang untuk memberikan visualisasi geografis yang comprehensive dari semua laporan bencana di wilayah Lebak Selatan. Implementasi MapFragment terintegrasi dengan Jetpack Compose melalui AndroidView, memungkinkan embedding native map component dalam declarative UI framework. Custom map styling diterapkan untuk memberikan appearance yang konsisten dengan theme aplikasi dan meningkatkan readability dalam berbagai kondisi pencahayaan, termasuk night mode untuk penggunaan dalam kondisi low-light.



Gambar 3.13. Peta Interaktif dengan Visualisasi Geografis Laporan Bencana

Gambar 3.13 mendemonstrasikan implementasi peta interaktif menggunakan Google Maps SDK yang menampilkan visualisasi geografis

dari laporan bencana di wilayah target. Peta menggunakan satellite view yang memberikan konteks geografis yang jelas tentang kondisi terrain dan infrastruktur di area yang terdampak bencana. Custom marker dengan ikon yang berbeda-beda digunakan untuk menandai lokasi bencana berdasarkan jenisnya, memudahkan identifikasi cepat jenis bencana dari tampilan peta. Floating action button di pojok kanan bawah memungkinkan pengguna menambahkan laporan bencana baru langsung dari tampilan peta dengan menggunakan lokasi yang dipilih. Implementasi clustering algorithm memastikan bahwa peta tetap readable meskipun terdapat banyak laporan dalam area yang berdekatan.



Gambar 3.14. Detail Marker dan Info Window pada Peta Bencana

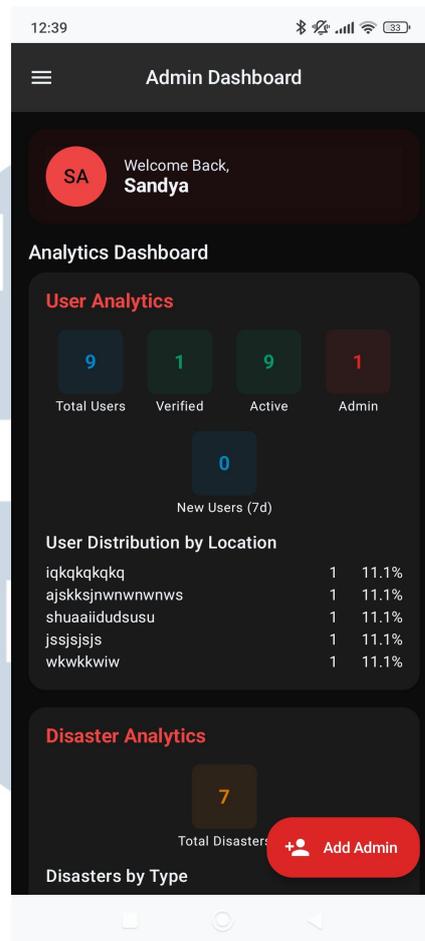
Gambar 3.14 menunjukkan implementasi lanjutan dari fitur peta dengan fokus pada interaksi marker dan info window yang lebih detail. Peta menampilkan multiple markers yang tersebar di berbagai lokasi dengan clustering yang efektif untuk area dengan kepadatan laporan tinggi. Setiap marker

menggunakan color coding yang konsisten untuk mengindikasikan severity level atau jenis bencana yang berbeda, memungkinkan pengguna untuk dengan cepat mengidentifikasi prioritas penanganan. Info window yang aktif menampilkan informasi komprehensif tentang laporan bencana yang dipilih, termasuk thumbnail foto dokumentasi, judul bencana, alamat, dan timestamp. Design info window menggunakan card-based layout dengan shadow yang memberikan depth dan hierarchy visual yang jelas. Control buttons untuk zoom dan location targeting ditempatkan secara ergonomis untuk kemudahan akses saat penggunaan satu tangan, yang penting dalam situasi darurat di lapangan.

E.2 Panel Administrasi

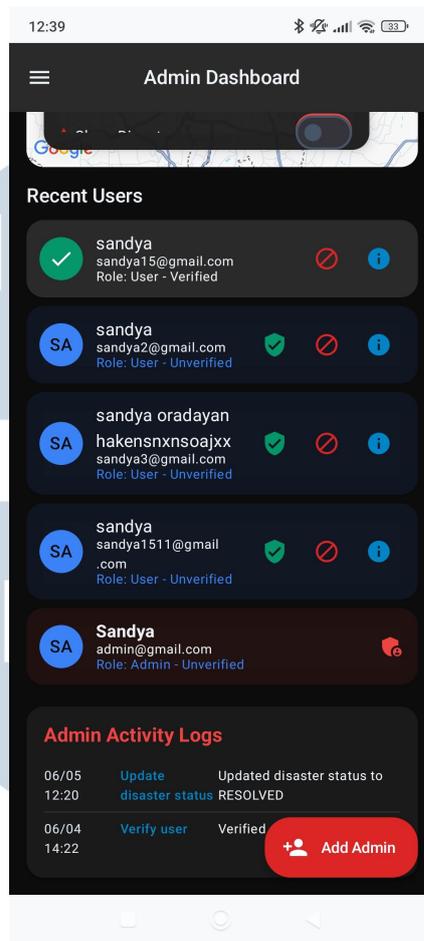
Pengembangan panel administrasi dirancang sebagai command center untuk mengelola seluruh aspek sistem pelaporan bencana, memberikan administrator kontrol penuh atas user management, disaster verification, dan system monitoring. Implementasi AdminDashboard menggunakan card-based analytics layout yang menampilkan key metrics dalam format yang mudah dipahami dan actionable. Dashboard menampilkan real-time statistics seperti jumlah total pengguna, pengguna terverifikasi, laporan bencana aktif, dan trend data dalam periode tertentu dengan visualisasi chart yang informatif.





Gambar 3.15. *Admin Dashboard* dengan *Analytics* Komprehensif dan *User Management*

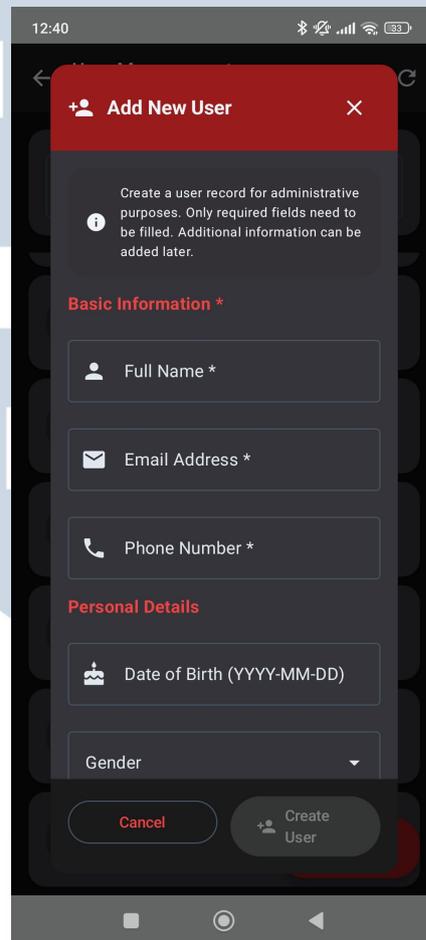
Gambar 3.15 mendemonstrasikan implementasi Admin Dashboard yang menyediakan overview komprehensif tentang sistem melalui analytics real-time dan user management tools. Header dengan hamburger menu memberikan akses ke berbagai fungsi administratif. Section welcome menampilkan sambutan personal dengan avatar, menciptakan pengalaman yang personal untuk administrator. Analytics Dashboard terbagi menjadi kategori utama: User Analytics dan Disaster Analytics. User Analytics menampilkan metrics penting dengan color coding yang intuitif untuk total users, verified users, active users, admin count, dan new users dalam periode tertentu. Section user distribution by location menampilkan distribusi geografis pengguna dengan data lokasi yang memberikan insight tentang sebaran geografis pengguna aplikasi. Disaster Analytics menampilkan total disasters yang memberikan overview cepat tentang jumlah total laporan bencana dalam sistem. Floating action button di pojok kanan bawah memberikan akses cepat untuk menambahkan administrator baru ke sistem.



Gambar 3.16. Admin Dashboard dengan User Management dan Activity Logs

Gambar 3.16 menunjukkan kelanjutan dari Admin Dashboard yang fokus pada user management dan activity monitoring. Bagian atas menampilkan embedded Google Maps dengan toggle switch untuk mengaktifkan/menonaktifkan tampilan peta, memberikan konteks geografis untuk operasi admin. Section recent users menampilkan daftar pengguna terbaru dengan informasi lengkap dan action buttons yang komprehensif. Setiap user card menampilkan avatar dengan inisial, nama lengkap, email address, dan role dengan status verifikasi. User cards menampilkan berbagai status verifikasi dengan action buttons yang sesuai untuk verify, disable, dan info dengan color coding yang konsisten. Beberapa user menampilkan status verified dengan checkmark, sementara yang lain menampilkan status unverified dengan action buttons yang tersedia. Admin accounts mendapat perlakuan khusus dengan action buttons yang berbeda, mengindikasikan perlakuan khusus untuk akun administrator. Section admin activity logs di bagian bawah menampilkan riwayat aktivitas admin dengan timestamp yang akurat, menunjukkan

berbagai aktivitas seperti update disaster status dan verify user dengan detail yang relevan. Floating action button tetap tersedia untuk menambahkan administrator baru dengan cepat.



Gambar 3.17. Modal Add New User untuk Administrative User Creation

Gambar 3.17 mendemonstrasikan implementasi modal add new user yang memungkinkan administrator membuat akun pengguna baru untuk keperluan administratif. Header modal menampilkan judul dengan ikon user dan tombol close di pojok kanan atas untuk navigasi yang jelas. Information panel dengan ikon info memberikan instruksi yang jelas tentang tujuan pembuatan user record dan field yang perlu diisi. Section basic information dengan asterisk mengindikasikan field yang wajib diisi, mencakup full name, email address, dan phone number dengan ikon yang sesuai dan input fields yang jelas. Section personal details menyediakan field tambahan untuk informasi yang lebih lengkap seperti date of birth dengan format yang standar dan gender dengan dropdown selector. Action buttons di

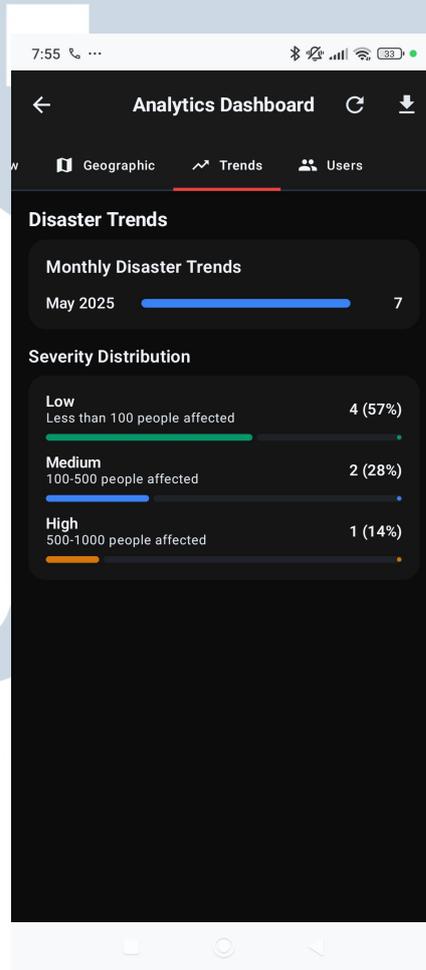
bagian bawah terdiri dari cancel dengan styling outline untuk membatalkan operasi, dan create user untuk mengkonfirmasi pembuatan akun baru. Design modal menggunakan dark theme yang konsisten dengan aplikasi utama, dengan contrast yang optimal untuk readability dan hierarchy visual yang jelas antara required dan optional fields.

Pengembangan AdminMapScreen dilengkapi dengan real-time user location tracking yang memungkinkan administrator memonitor lokasi pengguna yang mengaktifkan location sharing, sangat penting untuk koordinasi tim penyelamat saat terjadi bencana. Setup role-based access control dengan security checks memastikan hanya pengguna dengan role admin yang dapat mengakses fitur-fitur sensitif, dengan multiple layer authentication untuk keamanan maksimal. Implementasi audit logging melakukan tracking semua admin activities untuk transparansi dan accountability, mencatat setiap tindakan admin dengan timestamp dan user ID.



Gambar 3.18. Analytics Dashboard dengan Key Metrics dan Recent Activity Monitoring

Gambar 3.18 mendemonstrasikan implementasi Analytics Dashboard yang menyediakan overview komprehensif tentang situasi bencana melalui key metrics dan monitoring aktivitas terkini. Header dengan tombol back, refresh, dan download memberikan navigasi yang jelas dan akses ke fungsi export data. Tab navigation menampilkan kategori utama seperti overview, geographic, dan trends untuk berbagai jenis analisis. Section key metrics menggunakan card-based layout dengan color coding yang intuitif untuk total disasters, active incidents, resolved cases, dan people affected dengan statistik yang relevan. Section recent activity menampilkan timeline aktivitas terbaru dengan status indicators untuk berbagai jenis kejadian seperti disaster resolved dan new disaster reported dengan informasi temporal yang akurat. Setiap entry activity mencakup informasi lokasi dan detail yang memberikan traceability yang komprehensif untuk audit dan monitoring.



Gambar 3.19. Analytics Dashboard dengan Trends Analysis dan Severity Distribution

Gambar 3.19 menunjukkan implementasi tab trends dalam Analytics

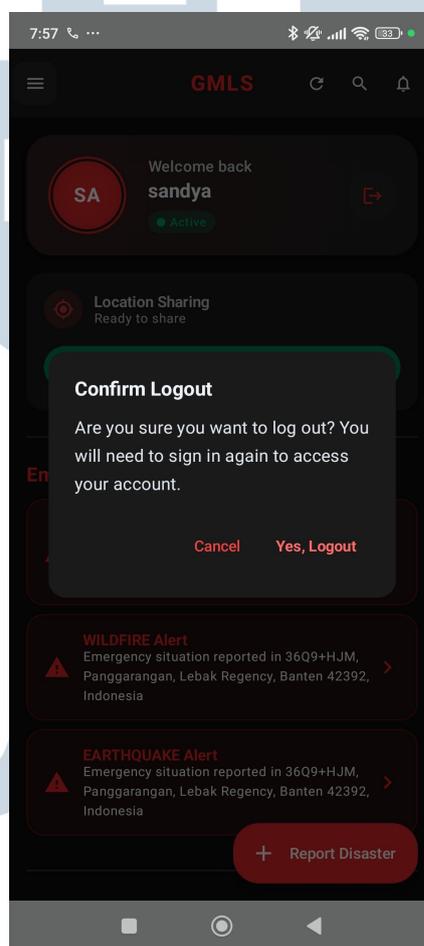
Dashboard yang fokus pada analisis temporal dan distribusi severity bencana dengan visualisasi data yang informatif. Header konsisten dengan navigation yang sama, namun tab trends kini aktif dengan highlight visual. Section disaster trends menyediakan analisis temporal dengan subsection monthly disaster trends yang menampilkan data periode tertentu dengan progress bar dan angka yang memberikan visualisasi cepat tentang volume kejadian per bulan. Section severity distribution menggunakan horizontal bar charts dengan color coding yang intuitif untuk kategori severity dengan range dampak yang berbeda-beda. Setiap kategori dilengkapi dengan deskripsi range dampak yang jelas dan persentase distribusi, memberikan insight yang valuable untuk risk assessment dan resource allocation. Visualisasi menggunakan progress bars dengan panjang yang proporsional terhadap nilai, memudahkan perbandingan visual antar kategori severity.

E.3 Sistem Notifikasi dan Alert

Push notification system dirancang sebagai backbone komunikasi darurat yang memungkinkan penyebaran informasi kritis secara real-time kepada seluruh pengguna aplikasi. Integrasi Firebase Cloud Messaging menyediakan infrastruktur push notifications yang reliable dan scalable, dengan automatic retry mechanism untuk memastikan delivery notification bahkan dalam kondisi koneksi yang tidak stabil. Implementasi notification channels untuk kategorisasi alert memungkinkan pengguna mengatur preferensi notifikasi berdasarkan jenis bencana atau tingkat keparahan, memberikan kontrol personal atas jenis informasi yang ingin mereka terima. Pengembangan emergency broadcast system memungkinkan admin mengirim alert massal kepada semua pengguna atau pengguna di area geografis tertentu, dengan geofencing capability untuk targeting yang presisi. Setup notification scheduling menyediakan functionality untuk reminder dan follow-up otomatis, seperti mengingatkan pengguna untuk update status keselamatan atau memberikan informasi lanjutan tentang perkembangan penanganan bencana. Implementasi in-app notifications dengan custom UI memberikan pengalaman yang seamless ketika pengguna sedang aktif menggunakan aplikasi, dengan rich media support untuk menampilkan gambar atau peta lokasi dalam notifikasi. Pengembangan notification history dan management memungkinkan pengguna melihat riwayat notifikasi yang pernah diterima dan mengatur preferensi notifikasi sesuai kebutuhan mereka.

E.4 Fitur Settings dan Konfigurasi Aplikasi

Pengembangan fitur Settings dirancang untuk memberikan kontrol penuh kepada pengguna atas preferensi dan konfigurasi aplikasi sesuai dengan kebutuhan personal mereka. Implementasi SettingsScreen menggunakan card-based layout yang terorganisir dengan baik untuk memudahkan akses ke berbagai opsi pengaturan. Setiap kategori settings dilengkapi dengan ikon yang intuitif dan deskripsi yang jelas untuk membantu pengguna memahami fungsi setiap pengaturan.



Gambar 3.20. Layar *Settings* dengan Konfigurasi Komprehensif untuk Personalisasi Aplikasi

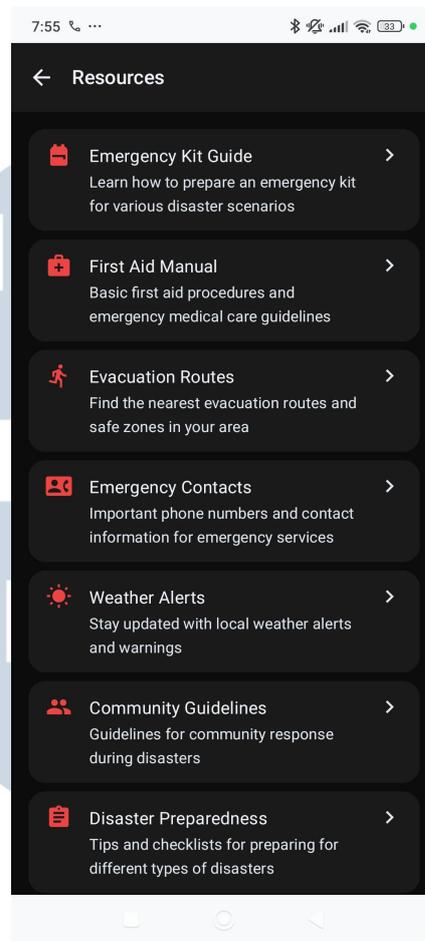
Gambar 3.20 mendemonstrasikan implementasi layar Settings yang menyediakan kontrol komprehensif atas konfigurasi aplikasi dengan interface yang clean dan user-friendly. Header dengan tombol back memberikan navigasi yang jelas dan konsisten dengan design pattern aplikasi. Section theme dengan ikon

palette memberikan opsi untuk mengubah appearance aplikasi dengan deskripsi dan setting saat ini, memungkinkan pengguna memilih antara light dan dark theme sesuai preferensi atau kondisi pencahayaan. Section language dengan ikon globe menyediakan opsi untuk mengubah bahasa interface aplikasi dengan deskripsi dan setting saat ini, mendukung multilingual untuk aksesibilitas yang lebih luas. Section notifications dengan ikon bell memberikan akses ke pengaturan preferensi notifikasi, memungkinkan pengguna mengontrol jenis dan frekuensi notifikasi yang ingin diterima. Section privacy dengan ikon shield menyediakan pengaturan privasi, memberikan kontrol atas data sharing dan privacy preferences. Section about dengan ikon information menampilkan informasi aplikasi termasuk versi yang digunakan, memberikan transparency tentang aplikasi. Setiap section menggunakan card design dengan background yang subtle dan spacing yang konsisten, menciptakan hierarchy visual yang jelas dan memudahkan navigasi. Dark theme yang diterapkan memberikan pengalaman visual yang nyaman dan modern, dengan contrast yang optimal untuk readability.

E.5 Fitur Resources dan Emergency Information

Pengembangan fitur Resources dirancang sebagai pusat informasi komprehensif yang menyediakan panduan dan referensi penting untuk kesiapsiagaan dan respons bencana. Implementasi ResourcesScreen menggunakan card-based layout yang terorganisir dengan baik untuk memudahkan akses ke berbagai jenis informasi emergency. Setiap kategori resources dilengkapi dengan ikon yang intuitif dan deskripsi yang jelas untuk membantu pengguna memahami konten yang tersedia.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A



Gambar 3.21. Layar *Resources* dengan Panduan Komprehensif untuk *Emergency Preparedness*

Gambar 3.21 mendemonstrasikan implementasi layar *Resources* yang menyediakan akses komprehensif ke berbagai panduan dan informasi penting untuk kesiapsiagaan bencana. Header dengan tombol back memberikan navigasi yang jelas dan konsisten dengan design pattern aplikasi. Section emergency kit guide dengan ikon yang sesuai menyediakan panduan lengkap tentang cara mempersiapkan emergency kit untuk berbagai skenario bencana dengan deskripsi yang informatif. Section first aid manual dengan ikon medical memberikan akses ke prosedur pertolongan pertama dasar dan panduan perawatan medis darurat, essential untuk situasi emergency. Section evacuation routes dengan ikon yang relevan membantu pengguna menemukan rute evakuasi terdekat dan zona aman di area mereka, informasi yang critical untuk perencanaan evakuasi yang efektif. Section emergency contacts menyediakan nomor telepon penting dan informasi kontak untuk layanan darurat, memungkinkan akses cepat ke bantuan profesional.

Section weather alerts memberikan update terkini tentang peringatan cuaca lokal, membantu pengguna tetap informed tentang kondisi cuaca yang berpotensi berbahaya. Section community guidelines menyediakan panduan untuk respons komunitas selama bencana, promoting koordinasi antar warga. Section disaster preparedness menawarkan tips dan checklist untuk persiapan menghadapi berbagai jenis bencana, membantu pengguna membangun resilience dan readiness. Setiap section menggunakan arrow indicator di sebelah kanan yang mengindikasikan bahwa konten dapat diakses dengan tap, memberikan affordance yang jelas untuk interaksi.

3.4 Kendala dan Solusi yang Ditemukan

Selama proses pengembangan aplikasi Android untuk pelaporan, monitoring, dan analisis bencana pada GMLS, ditemukan beberapa kendala yang memerlukan solusi. Berikut adalah kendala utama yang dihadapi beserta solusi yang diterapkan:

1. Keterbatasan spesifikasi laptop untuk development

Android Studio membutuhkan spesifikasi laptop yang cukup tinggi untuk development, sedangkan laptop yang tersedia saat trip memiliki spesifikasi terbatas sehingga proses development menjadi lambat. Solusi yang diterapkan adalah melakukan optimasi project dengan mengurangi plugin yang tidak perlu, menggunakan emulator ringan, dan melakukan development fitur utama di laptop dengan spesifikasi lebih baik sebelum trip.

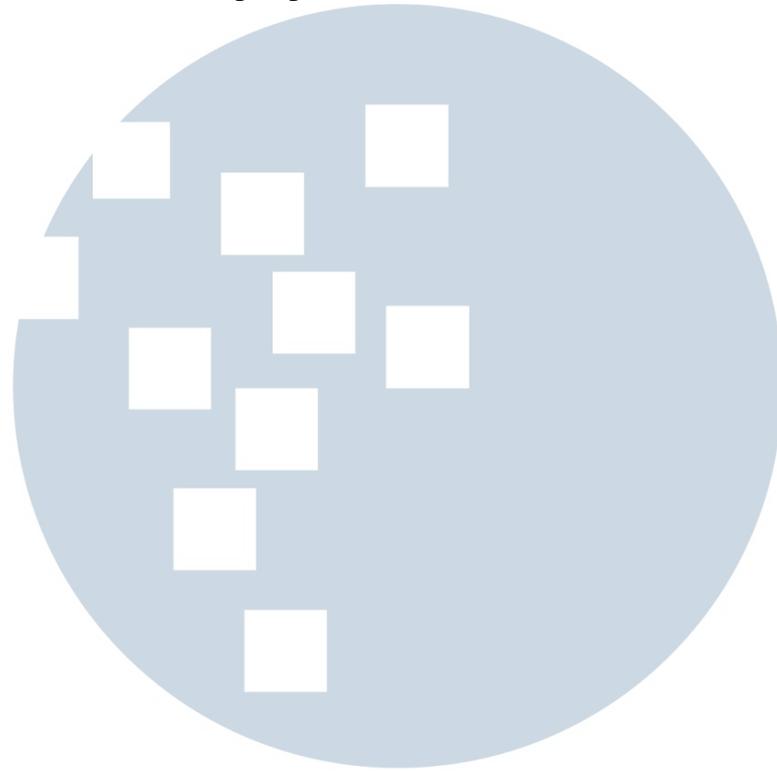
2. Keterbatasan infrastruktur internet di wilayah remote

Wilayah Lebak Selatan memiliki coverage internet yang tidak merata, dengan beberapa area memiliki koneksi yang lambat atau tidak stabil. Solusi yang diterapkan adalah mengembangkan fitur offline untuk menyimpan data sementara ketika tidak ada koneksi internet, kemudian melakukan sinkronisasi otomatis saat koneksi tersedia.

3. Variasi tingkat literasi digital pengguna

Pengguna aplikasi memiliki tingkat literasi digital yang beragam, sehingga memerlukan interface yang mudah dipahami semua kalangan. Solusi yang diterapkan adalah membuat desain interface yang sederhana

dengan icon yang jelas, menambahkan tutorial penggunaan aplikasi, dan menyederhanakan alur pelaporan bencana.



UMMN

UNIVERSITAS
MULTIMEDIA
NUSANTARA