GoNotes Mobile App 📱

📌 Deskripsi Singkat

GoNotes Mobile adalah aplikasi pencatatan Android berbasis **Kotlin + Jetpack Compose**, yang terhubung ke backend GoNotes (berbasis Go + PostgreSQL + Redis). Aplikasi ini menggunakan arsitektur **MVVM + Clean Architecture** serta integrasi API melalui Retrofit.

Aplikasi dikembangkan langsung di Android Studio tanpa menggunakan Docker di sisi mobile, sedangkan backend tetap dijalankan dalam Docker.

⚙️ Tech Stack

Komponen	Teknologi
Bahasa	Kotlin
UI Framework	Jetpack Compose
Navigation	Jetpack Navigation (Compose)
API Layer	Retrofit + OkHttp
JSON Parser	Moshi / kotlinx.serialization
State Mgmt	StateFlow + ViewModel
DI	Hilt
Storage	DataStore (untuk token/session)

📁 Struktur Folder

```
gonotes-app/
├── data/
│   ├── remote/        # Retrofit API, DTO, ApiService
│   ├── repository/    # Implementasi repository pattern
│   └── local/         # DataStore untuk session management
├── domain/
│   ├── model/         # Business entities
│   ├── repository/    # Repository interfaces
│   └── usecase/       # Business logic per fitur
├── presentation/
```

```
│   ├── auth/          # Login/Register/Logout UI + ViewModel
│   ├── notes/         # CRUD Notes UI + ViewModel
│   ├── profile/       # Profile Management UI + ViewModel
│   ├── sessions/      # Session Management UI + ViewModel
│   ├── components/    # Reusable UI Components
│   └── navigation/    # Navigasi screen & route
├── di/                # Hilt modules (network, repo, dsb.)
├── utils/             # Constants, Extensions, Helpers
├── MainActivity.kt    # Root Activity
└── App.kt             # Root Composable
```

🌐 API Endpoints Overview

Base Configuration

```
val BASE_URL = "http://10.0.2.2:8080/"  // Untuk Android Emulator
// val BASE_URL = "http://localhost:8080/"  // Untuk testing lokal
```

Health Check

```
GET /health
```

Authentication (`/api/v1/auth/`)

- `POST /register` - Registrasi user baru
- `POST /login` - Login dan mendapat JWT tokens
- `POST /refresh` - Refresh access token
- `POST /logout` - Logout dan invalidate refresh token

User Management (`/api/v1/user/`)

- `GET /profile` - Get user profile
- `PUT /profile` - Update user profile
- `GET /sessions` - Get active sessions (legacy)

Advanced Session Management (`/api/v1/user/sessions/`)

- `GET /active` - Get sessions dengan device info & analytics
- `GET /stats` - Get session statistics
- `DELETE /` - Logout dari semua device
- `DELETE /{session_id}` - Logout dari device tertentu
- `POST /invalidate` - Alternative session invalidation

Notes Management (`/api/v1/notes/`)

- `GET /` - Get user notes dengan pagination
- `POST /` - Create new note
- `GET /{note_id}` - Get specific note
- `PUT /{note_id}` - Update note
- `DELETE /{note_id}` - Delete note (soft delete)
- `POST /search` - Search notes dengan filter
- `GET /public` - Get public notes (no auth required)

🔐 Session & Authentication Handling

JWT Token Management

```
// Token disimpan di DataStore
data class AuthTokens(
    val accessToken: String,
    val refreshToken: String,
    val expiresIn: Int // dalam detik (900 = 15 menit)
)
```

Interceptor Setup

```
class AuthInterceptor(private val tokenManager: TokenManager) : Intercept
or {
    override fun intercept(chain: Interceptor.Chain): Response {
        val originalRequest = chain.request()
        val token = tokenManager.getAccessToken()
```

```
        val authenticatedRequest = originalRequest.newBuilder()
            .header("Authorization", "Bearer $token")
            .build()

        return chain.proceed(authenticatedRequest)
    }
}
```

Auto Token Refresh (Opsional)

```
class TokenAuthenticator(
    private val tokenManager: TokenManager,
    private val authApi: AuthApiService
) : Authenticator {

    override fun authenticate(route: Route?, response: Response): Request?
{
        if (response.code == 401) {
            // Attempt to refresh token
            val refreshToken = tokenManager.getRefreshToken()
            val newTokens = authApi.refreshToken(RefreshTokenRequest(refres
hToken))

            if (newTokens.isSuccessful) {
                tokenManager.saveTokens(newTokens.body()!!.data)
                return response.request.newBuilder()
                    .header("Authorization", "Bearer ${newTokens.body()!!.data.ac
cessToken}")
                    .build()
            }
        }
        return null
    }
}
```

🔁 Format Response API Standar

Success Response

```
{
  "status": "success",
  "code": 200,
  "message": "Operation successful",
  "data": {
    // Response data here
  }
}
```

Error Response

```
{
  "status": "error",
  "code": 400,
  "message": "Error description",
  "error": "Detailed error information"
}
```

📋 Data Models

User Model

```
data class User(
    val id: String,
    val email: String,
    val fullName: String,
    val createdAt: String,
    val updatedAt: String
)
```

Note Model

```
data class Note(
    val id: String,
    val title: String,
```

```
    val content: String,
    val tags: List<String> = emptyList(),
    val isPublic: Boolean = false,
    val userId: String,
    val createdAt: String,
    val updatedAt: String
)
```

Session Model

```
data class Session(
    val id: String,
    val userAgent: String,
    val ipAddress: String,
    val createdAt: String,
    val expiresAt: String,
    val isCurrent: Boolean,
    // Extended properties for advanced session management
    val deviceInfo: DeviceInfo? = null,
    val location: String? = null
)

data class DeviceInfo(
    val deviceType: String, // mobile, desktop, tablet
    val os: String,
    val browser: String
)
```

🚀 Roadmap Pengembangan per Batch

✅ Batch 1 — Setup & Authentication

- ☑️ ~~Inisialisasi project Android Studio~~
- ☑️ ~~Setup Hilt DI~~
- ☑️ ~~Setup Retrofit + Moshi/kotlinx~~
- ☑️ ~~Setup DataStore untuk penyimpanan token~~

- [x] ~~Setup basic navigation (`NavHost`)~~
- [x] ~~Login & Register screen dengan validation~~
- [x] ~~Integrasi API `/auth/login` , `/auth/register`~~
- [x] ~~Token management + auto-refresh~~
- [x] ~~Interceptor untuk `Authorization: Bearer ...`~~
- [x] ~~Error handling untuk authentication~~

📝 Batch 2 — Modul Notes Management

- [ ] Halaman daftar catatan dengan pagination (`GET /notes`)
- [ ] Search functionality (`POST /notes/search`)
- [ ] Komponen UI: NoteCard, SearchBar
- [ ] Tambah catatan (`POST /notes`)
- [ ] Edit catatan (`PUT /notes/:id`)
- [ ] Hapus catatan (`DELETE /notes/:id`)
- [ ] Public notes viewer (`GET /notes/public`)
- [ ] Tag management dan filtering
- [ ] Pull-to-refresh implementation
- [ ] Loading states + comprehensive error handler

👤 Batch 3 — Profile & User Management

- [ ] Halaman profil (`GET /user/profile`)
- [ ] Edit profil (`PUT /user/profile`)
- [ ] Cache invalidation setelah update profil
- [ ] Profile picture placeholder
- [ ] Account settings UI

🔐 Batch 4 — Advanced Session Management

- [ ] Halaman active sessions (`GET /user/sessions/active`)
- [ ] Session statistics (`GET /user/sessions/stats`)
- [ ] Device information display

- [ ] Logout dari device tertentu (`DELETE /user/sessions/:id`)
- [ ] Logout dari semua device (`DELETE /user/sessions`)
- [ ] Session security alerts
- [ ] Konfirmasi logout dengan dialog

🔧 Batch 5 — Performance & Polish

- [ ] Offline support dengan Room database
- [ ] Background sync
- [ ] Push notifications (jika ada)
- [ ] Dark theme support
- [ ] Accessibility improvements
- [ ] Performance optimizations
- [ ] Comprehensive testing

🛠️ Environment Variables

```
// BuildConfig atau local.properties
object Config {
    const val BASE_URL = "http://10.0.2.2:8080/"
    const val DEBUG = true
    const val CONNECTION_TIMEOUT = 30L // seconds
    const val READ_TIMEOUT = 30L // seconds
}
```

🧪 Testing Strategy

Unit Tests

- Repository layer testing
- UseCase testing
- ViewModel testing

Integration Tests

- API integration tests
- Database tests
- Authentication flow tests

UI Tests

- Compose UI testing
- Navigation testing
- User journey testing

📱 Device Support

Minimum Requirements

- Android 7.0 (API level 24)
- 2GB RAM
- 100MB storage space

Target Devices

- Phone (4.7" - 6.8")
- Tablet (7" - 12")
- Foldable devices support

🔧 Build & Development Notes

💻 Development Workflow

Code Editor: Cursor AI - untuk AI-assisted coding dan development

Running & Testing: Android Studio - untuk build, run, dan emulator management

Setup Development Environment

1. **Install Tools**

```
# Install Cursor untuk coding
# Download dari https://cursor.sh/
```

```
# Install Android Studio untuk running
# Download dari https://developer.android.com/studio
```

2. **Project Setup**

```
# Clone repository
git clone <repo-url>
cd gonotes-app

# Open di Cursor untuk development
cursor .

# Buka juga di Android Studio untuk running
# File → Open → pilih folder project
```

3. **Development Flow**
   - **Coding**: Gunakan Cursor dengan AI assistance
   - **Sync**: Gradle sync di Android Studio setelah perubahan dependencies
   - **Build & Run**: Android Studio untuk compile dan run di emulator/device
   - **Debugging**: Android Studio untuk debugging dan logcat monitoring
4. **Backend Setup**

```
# Setup backend GoNotes dengan Docker
docker-compose up -d

# Verify backend running
curl http://localhost:8080/health
```

5. **Configuration**
   - Update `BASE_URL` sesuai environment di `Config.kt`
   - Sync project dengan Gradle di Android Studio
   - Setup emulator atau connect physical device

🔄 Recommended Workflow

```
graph LR
    A[Code in Cursor] --> B[Save Changes]
    B --> C[Sync in Android Studio]
    C --> D[Build & Run]
    D --> E[Test in Emulator]
    E --> F[Debug if needed]
    F --> A
```

📱 Emulator Configuration

```
// Untuk Android Emulator di Android Studio
val BASE_URL = "http://10.0.2.2:8080/"

// Untuk Physical Device (same network)
val BASE_URL = "http://192.168.1.xxx:8080/"  // IP komputer
```

📂 IDE & Tools Setup

Cursor AI (Primary Development)

```
# Features yang akan digunakan:
✅ AI-powered code completion
✅ Code generation dan refactoring
✅ Natural language to code conversion
✅ Error detection dan fixing suggestions
✅ Documentation generation
```

Android Studio (Build & Testing)

```
# Digunakan untuk:
✅ Gradle build system
✅ Android Emulator
✅ Device debugging
✅ Performance profiling
✅ Layout inspector
✅ APK generation
```

🔄 Development Sync Strategy

1. **Primary Coding**: Cursor AI untuk semua development
2. **File Watching**: Auto-sync atau manual sync ke Android Studio
3. **Build Process**: Android Studio untuk compile dan dependency management
4. **Testing**: Emulator dan device testing via Android Studio
5. **Debugging**: Android Studio debugger dan logcat untuk troubleshooting

⚠️ Important Notes

- Pastikan kedua IDE mengarah ke folder project yang sama
- Gradle sync di Android Studio setelah perubahan dependencies di Cursor
- Git operations bisa dilakukan dari Cursor atau Android Studio
- File conflicts minimal karena Cursor fokus di coding, Android Studio di building

Security Considerations

- Token disimpan dengan encryption di DataStore
- Network security config untuk HTTPS only (production)
- Certificate pinning (production)
- Obfuscation untuk release build

🪲 Error Handling

Network Errors

```
sealed class ApiResult<T> {
    data class Success<T>(val data: T) : ApiResult<T>()
    data class Error<T>(val code: Int, val message: String) : ApiResult<T>()
    data class NetworkError<T>(val exception: Throwable) : ApiResult<T>()
}
```

Common Error Codes

- `400`: Validation errors

- `401`: Authentication required / token expired
- `403`: Access forbidden
- `404`: Resource not found
- `500`: Server error

📚 Dependencies

Core Dependencies

```
implementation "androidx.compose.ui:ui:$compose_version"
implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"
implementation "androidx.compose.material3:material3:$material3_version"
implementation "androidx.navigation:navigation-compose:$nav_version"
implementation "androidx.hilt:hilt-navigation-compose:$hilt_compose_version"

// Networking
implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
implementation "com.squareup.retrofit2:converter-moshi:$retrofit_version"
implementation "com.squareup.okhttp3:logging-interceptor:$okhttp_version"

// DI
implementation "com.google.dagger:hilt-android:$hilt_version"
kapt "com.google.dagger:hilt-compiler:$hilt_version"

// Storage
implementation "androidx.datastore:datastore-preferences:$datastore_version"
```

Dokumentasi ini akan terus diperbarui seiring dengan perkembangan aplikasi dan penambahan fitur baru.