Implementasi (Rencana Langkah-langkah Rinci)

Berikut adalah rincian langkah demi langkah untuk membangun aplikasi "Mortis".

Tahap 1: Pengaturan Proyek dan Dependensi

1. **Buat Proyek Baru:** Di Android Studio, buat proyek baru dengan "Empty Views Activity".
   - **Name:** Mortis
   - **Language:** Java
   - **Minimum SDK:** API 21 atau lebih tinggi.
2. **Tambahkan Izin Internet:** Buka `AndroidManifest.xml` dan tambahkan izin untuk mengakses internet:
   ```
   <uses-permission android:name="android.permission.INTERNET" />
   ```
3. **Konfigurasi Dependensi:** Buka file `build.gradle` (Module: app) dan tambahkan *library* yang dibutuhkan:
   ```
   dependencies {
       // ... dependensi lainnya
       implementation 'com.squareup.retrofit2:retrofit:2.9.0'
       implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
       implementation 'androidx.cardview:cardview:1.0.0'
       implementation 'androidx.recyclerview:recyclerview:1.3.2'
       implementation 'com.google.ai.client.generativeai:generativeai:0.5.0'
   }
   ```
   Sinkronkan proyek setelah menambahkan dependensi.

Tahap 2: Integrasi History API dan Halaman Utama

1. **Buat Model Data (POJO):** Buat beberapa kelas Java untuk mencocokkan struktur JSON dari History API.
   - `HistoryResponse.java`: Berisi `Data` object.
   - `Data.java`: Berisi `List<Death> Deaths`.
   - `Death.java`: Berisi `String year`, `String text`.
2. **Buat Antarmuka Retrofit:** Buat *interface* Java untuk mendefinisikan *endpoint* API.
   - `HistoryApiService.java`:
     ```
     public interface HistoryApiService {
         @GET("date/{month}/{day}")
         Call<HistoryResponse> getEvents(@Path("month") int month, @Path("day") int day);
     }
     ```
3. **Desain UI Halaman Utama:** Buka `activity_main.xml`. Tambahkan `ProgressBar` dan

RecyclerView.

4. **Desain Item RecyclerView:** Buat file layout baru item_person.xml dengan CardView dan dua TextView.
5. **Buat Adapter RecyclerView:** Buat kelas DeathsAdapter.java dan implementasikan OnClickListener.
6. **Panggil API di** MainActivity.java**:** Panggil API secara asinkron, perbarui adapter pada onResponse, dan tangani onFailure.

Panduan Rinci Tahap 3: Integrasi Gemini API

Berikut adalah panduan lengkap dengan kode untuk mengintegrasikan Gemini API ke dalam aplikasi "Mortis" Anda, memastikan pemanggilan API dilakukan di *background thread* dan UI diperbarui dengan aman.

Langkah 3.1: Buat Model Data (POJO) untuk Respons Gemini

Pertama, buat kelas Java untuk menampung data yang akan Anda terima dari Gemini. Ini membuat proses parsing JSON menjadi jauh lebih mudah.

GeminiResponse.java

```
import java.util.List;
import java.util.Map;

public class GeminiResponse {
    private String name;
    private String birth;
    private String details;
    private List<Map<String, String>> sources;

    // Getters and Setters
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getBirth() { return birth; }
    public void setBirth(String birth) { this.birth = birth; }
    public String getDetails() { return details; }
    public void setDetails(String details) { this.details = details; }
    public List<Map<String, String>> getSources() { return sources; }
    public void setSources(List<Map<String, String>> sources) { this.sources = sources; }
```

```
}
```

Langkah 3.2: Buat Antarmuka Callback

Karena pemanggilan API bersifat asinkron, kita memerlukan *callback interface* untuk mengirimkan hasilnya kembali ke `DetailActivity` setelah selesai.

`GeminiCallback.java`

```
public interface GeminiCallback {
    void onSuccess(GeminiResponse response);
    void onFailure(Exception e);
}
```

Langkah 3.3: Buat Layanan Gemini (GeminiService)

Ini adalah kelas inti yang akan menangani semua logika untuk memanggil Gemini API. Ini akan berjalan di *background thread* agar tidak memblokir UI aplikasi.

`GeminiService.java`

```
import com.google.common.collect.ImmutableList;
import com.google.genai.Client;
import com.google.genai.ResponseStream;
import com.google.genai.types.*;
import com.google.gson.Gson;
import com.google.gson.JsonSyntaxException;

import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class GeminiService {

    private final ExecutorService executor = Executors.newSingleThreadExecutor();
    private final Client client;
    private final Gson gson = new Gson();

```

```
41.    public GeminiService(String apiKey) {
42.        this.client = Client.builder().apiKey(apiKey).build();
43.    }
44.
45.    public void getBio(String personInfo, GeminiCallback callback) {
46.        executor.execute(() -> {
47.            try {
48.                // Konfigurasi model dan sistem prompt
49.                GenerateContentConfig config = GenerateContentConfig.builder()
50.                        .temperature(0.2f)
51.                        .responseMimeType("application/json") // Meminta JSON secara
    eksplisit
52.                        .build();
53.
54.                Content systemInstruction = Content.fromParts(
55.                    Part.fromText("You are an expert historian who is capable of finding
    details about a historical figure from their name and the date of their death. You
    must use grounding search tool to verify the information. Always return the
    response with this JSON format:\n\n{\n\"name\" : \"Full name of the historical
    figure\",\n\"birth\" : \"Birth date if there's any information\",\n\"details\": \"Detailed
    3-paragraph biography about the person (clean format without source
    annotation)\",\n\"sources\": [{\"source number\": \"Links of the information
    sources\"}]\n}")
56.                );
57.
58.                // Membuat riwayat percakapan (poin penting untuk few-shot)
59.                List<Content> contents = ImmutableList.of(
60.                    // Contoh 1: User
61.                    Content.builder()
62.                        .role("user")
63.                        .parts(ImmutableList.of(
64.                            Part.fromText("Who was John III, pope of the Catholic Church
    that was deceased in 574")
65.                        ))
66.                        .build(),
67.                    // Contoh 1: Model (respons yang diharapkan)
68.                    Content.builder()
69.                        .role("model")
70.                        .parts(ImmutableList.of(
```

```
                    Part.fromText("{\n\"name\": \"John III\",\n\"birth\": \"Around 530 AD\",\n\"details\": \"John III, born Catelinus in Rome, was the Pope of the Catholic Church from July 17, 561, to his death on July 13, 574. Born to a distinguished family, his father, Anastasius, held the title of illustris. His papacy occurred during the Lombard invasion of Italy, a period of significant upheaval, resulting in the destruction of many records from his reign.\\n\\nDespite the challenges of his time, John III is remembered as a magnanimous pontiff who was dedicated to the welfare of the people. In one notable act, he intervened on behalf of two bishops, Salonius of Embrun and Sagittarius of Gap, who had been condemned at a synod in Lyons. King Guntram of Burgundy believed they were unjustly condemned and appealed to John, who decided they should be restored to their sees.\\n\\nDuring the Lombard invasion, John III sought assistance from Narses, the governor of Naples, to defend Rome. He even retreated to the catacombs of Praetextatus for several months, where he continued to perform ordinations. After Narses' death, John returned to the Lateran Palace and, with a newfound appreciation for the catacombs, ordered their repair and ensured they received the necessities for Mass. He was buried in St. Peter's.\",\n\"sources\": [\n  {\"1\": \"https://example.com/source1\"},\n  {\"2\": \"https://example.com/source2\"}\n]\n}")
                ))
                .build(),
            // Input aktual dari pengguna
            Content.builder()
                .role("user")
                .parts(ImmutableList.of(
                    Part.fromText(personInfo)
                ))
                .build()
        );

        // Memanggil API
        ResponseStream<GenerateContentResponse> responseStream = client.models.generateContentStream("gemini-pro", contents, config, systemInstruction);

        // Menggabungkan respons streaming menjadi satu string
        StringBuilder fullResponse = new StringBuilder();
        for (GenerateContentResponse res : responseStream) {
            fullResponse.append(res.text());
```

```
            }

            // Membersihkan dan mem-parsing JSON
            String jsonResponse = fullResponse.toString().replace("```json",
    "").replace("```", "").trim();
            GeminiResponse geminiResponse = gson.fromJson(jsonResponse,
    GeminiResponse.class);

            if (geminiResponse != null) {
                callback.onSuccess(geminiResponse);
            } else {
                callback.onFailure(new Exception("Failed to parse JSON response."));
                 }

              } catch (Exception e) {
                 callback.onFailure(e);
              }
           });
       }
    }
```

Langkah 3.4: Panggil `GeminiService` dari `DetailActivity`

Sekarang, di `DetailActivity`, Anda bisa menggunakan `GeminiService` untuk mengambil data dan kemudian memperbarui UI.

`DetailActivity.java`

```
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.widget.ProgressBar;
    import android.widget.TextView;
    import android.widget.Toast;
    import androidx.appcompat.app.AppCompatActivity;

    public class DetailActivity extends AppCompatActivity {

        private TextView tvName, tvBirth, tvDetails, tvSources;
```

```
119.     private ProgressBar progressBar;
120.     private Button btnSave;
121.     private GeminiService geminiService;
122.
123.     @Override
124.     protected void onCreate(Bundle savedInstanceState) {
125.         super.onCreate(savedInstanceState);
126.         setContentView(R.layout.activity_detail);
127.
128.         // Inisialisasi Views
129.         tvName = findViewById(R.id.tv_name);
130.         tvBirth = findViewById(R.id.tv_birth);
131.         tvDetails = findViewById(R.id.tv_details);
132.         tvSources = findViewById(R.id.tv_sources);
133.         progressBar = findViewById(R.id.progress_bar_detail);
134.         btnSave = findViewById(R.id.btn_save);
135.
136.         // Ambil API Key (cara aman, lihat di bawah)
137.         // Untuk sekarang, Anda bisa hardcode sementara untuk tes
138.         String apiKey = "YOUR_GEMINI_API_KEY";
139.         geminiService = new GeminiService(apiKey);
140.
141.         // Ambil data dari Intent
142.         String personText = getIntent().getStringExtra("PERSON_TEXT");
143.         String personYear = getIntent().getStringExtra("PERSON_YEAR");
144.
145.         // Buat prompt dan panggil service
146.         if (personText != null && personYear != null) {
147.             String prompt = "Who was " + personText + " that was deceased in " +
    personYear;
148.             fetchBiography(prompt);
149.         } else {
150.             Toast.makeText(this, "Error: No data received.",
    Toast.LENGTH_SHORT).show();
151.         }
152.     }
153.
154.     private void fetchBiography(String prompt) {
155.         progressBar.setVisibility(View.VISIBLE);
```

```
156.
157.         geminiService.getBio(prompt, new GeminiCallback() {
158.             @Override
159.             public void onSuccess(GeminiResponse response) {
160.                 runOnUiThread(() -> {
161.                     progressBar.setVisibility(View.GONE);
162.                     tvName.setText(response.getName());
163.                     tvBirth.setText("Born: " + response.getBirth());
164.                     tvDetails.setText(response.getDetails());
165.
166.                     // Format dan tampilkan sumber
167.                     StringBuilder sourcesText = new StringBuilder("Sources:\n");
168.                     if (response.getSources() != null) {
169.                         for (int i = 0; i < response.getSources().size(); i++) {
170.                             sourcesText.append(i + 1).append(".
    ").append(response.getSources().get(i).values().iterator().next()).append("\n");
171.                         }
172.                     }
173.                     tvSources.setText(sourcesText.toString());
174.
175.                     // Aktifkan tombol simpan
176.                     btnSave.setEnabled(true);
177.                 });
178.             }
179.
180.             @Override
181.             public void onFailure(Exception e) {
182.                 runOnUiThread(() -> {
183.                     progressBar.setVisibility(View.GONE);
184.                     Toast.makeText(DetailActivity.this, "Failed to load biography: " +
    e.getMessage(), Toast.LENGTH_LONG).show();
185.                 });
186.             }
187.         });
188.     }
189. }
```

PENTING: Menyimpan API Key dengan Aman

Jangan pernah menyimpan API Key langsung di dalam kode (`"YOUR_GEMINI_API_KEY"`). Gunakan `local.properties` untuk menyimpannya dengan aman.

1. Buka file `local.properties` (jika tidak ada, buat file ini di root proyek Anda).
190. Tambahkan key Anda:
GEMINI_API_KEY="KEY_ANDA_DI_SINI"
2.
191. Buka file `build.gradle` (Module: app) dan tambahkan kode untuk membaca key tersebut:
def localProperties = new Properties()
192. def localPropertiesFile = rootProject.file('local.properties')
193. if (localPropertiesFile.exists()) {
194. localPropertiesFile.withReader('UTF-8') { reader ->
195. localProperties.load(reader)
196. }
197. }
198.
199. android {
200. // ...
201. defaultConfig {
202. // ...
203. buildConfigField "String", "GEMINI_API_KEY",
"\"${localProperties.getProperty('GEMINI_API_KEY')}\""
204. }
205. }
3.
4. Sinkronkan proyek Anda. Sekarang Anda dapat mengakses key di kode Java dengan aman:
String apiKey = BuildConfig.GEMINI_API_KEY;

206.

Tahap 4: Navigasi dan Penyelesaian

1. **Implementasi Navigasi:** Gunakan `Intent` untuk berpindah dari `MainActivity` ke `DetailActivity`, mengirimkan data tokoh yang dipilih.
2. **Penanganan Error:** Tambahkan pemeriksaan koneksi dan tangani kasus di mana API tidak mengembalikan data.
3. **Polesan UI:** Sesuaikan warna, font, dan spasi.

Tahap 5: Integrasi Database SQLite (Fitur Baru)

1. **Desain Database (Schema):**
    - **Nama Database:** mortis.db
    - **Versi Database:** 1
    - **Nama Tabel:** saved_figures
    - **Kolom Tabel:**
        - id: INTEGER PRIMARY KEY AUTOINCREMENT
        - name: TEXT NOT NULL
        - birth_date: TEXT
        - death_year: TEXT
        - details: TEXT NOT NULL
        - sources: TEXT (Akan menyimpan daftar sumber sebagai string JSON)
2. **Buat Database Helper:**
    - Buat kelas baru DatabaseHelper.java yang meng-extend SQLiteOpenHelper.
    - Implementasikan metode onCreate untuk mengeksekusi perintah SQL CREATE TABLE.
    - Implementasikan metode onUpgrade (untuk saat ini bisa dibiarkan kosong).
    - Buat metode CRUD (Create, Read, Update, Delete):
        - addFigure(GeminiResponse response, String deathYear): Untuk menyimpan data tokoh ke database.
        - getFigure(int id): Untuk mengambil satu tokoh berdasarkan ID.
        - getAllFigures(): Untuk mengambil semua tokoh yang tersimpan.
        - deleteFigure(int id): Untuk menghapus tokoh dari database.
3. **Implementasi Fitur Simpan:**
    - Di activity_detail.xml, tambahkan Button dengan id btn_save.
    - Di DetailActivity.java, setelah berhasil mendapatkan respons dari Gemini, aktifkan btn_save.
    - Set OnClickListener untuk tombol tersebut. Saat diklik, panggil metode addFigure() dari DatabaseHelper Anda. Tampilkan pesan konfirmasi (misalnya, menggunakan Toast) bahwa data telah disimpan.
4. **Buat Halaman "Tersimpan":**
    - Buat "Empty Views Activity" baru bernama SavedActivity.java.
    - Desain layout-nya (activity_saved.xml) dengan sebuah RecyclerView untuk menampilkan daftar tokoh yang disimpan.
    - Buat Adapter baru (SavedFiguresAdapter.java) yang mirip dengan DeathsAdapter untuk menampilkan data dari database.
    - Di SavedActivity.java, panggil getAllFigures() dari DatabaseHelper dan perbarui adapter.

5. **Navigasi ke Halaman "Tersimpan":**
    - Di MainActivity.java, tambahkan sebuah Button atau item menu untuk membuka SavedActivity.
    - Implementasikan OnClickListener untuk memulai SavedActivity menggunakan Intent.