

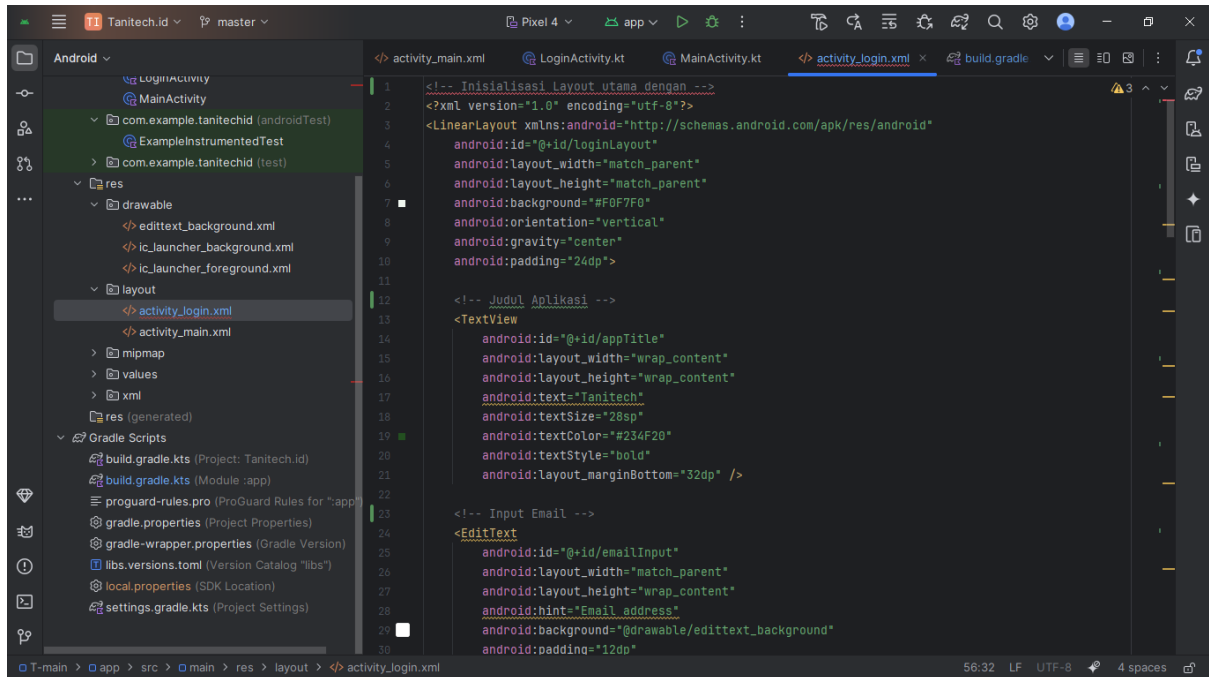
DOKUMENTASI

Nama : Lorensius Firngadi

NPM : 24382001P

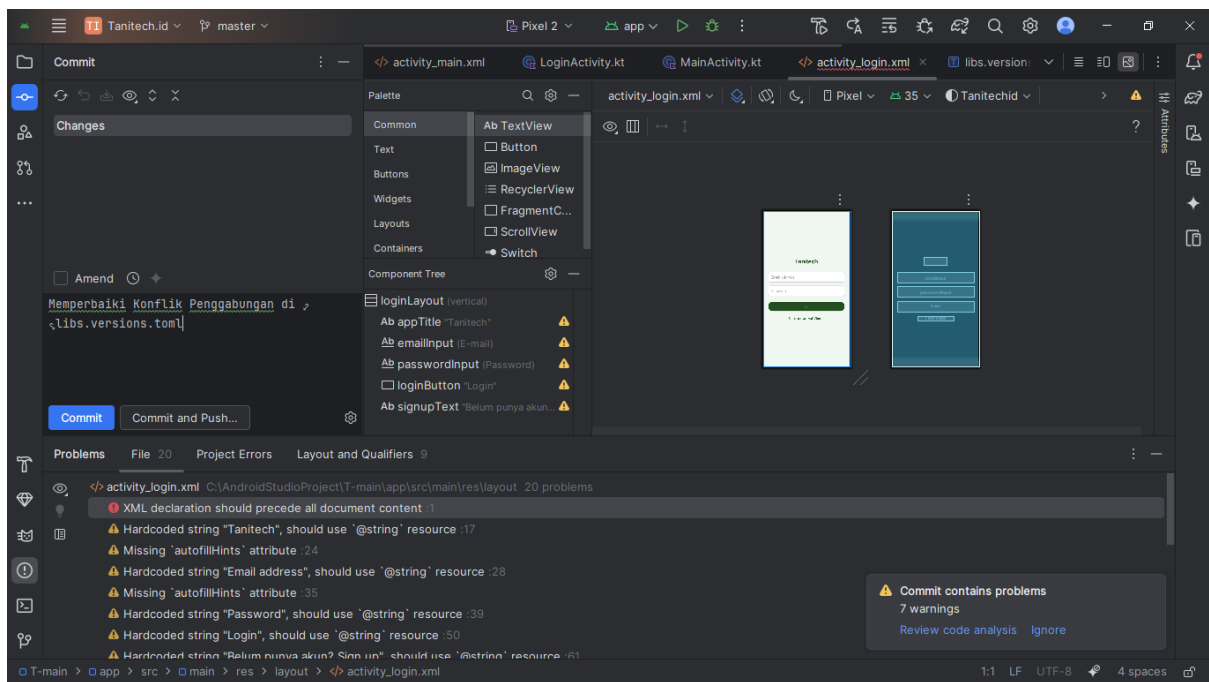
Kelas : IF 23 FX

1. Commit and Push



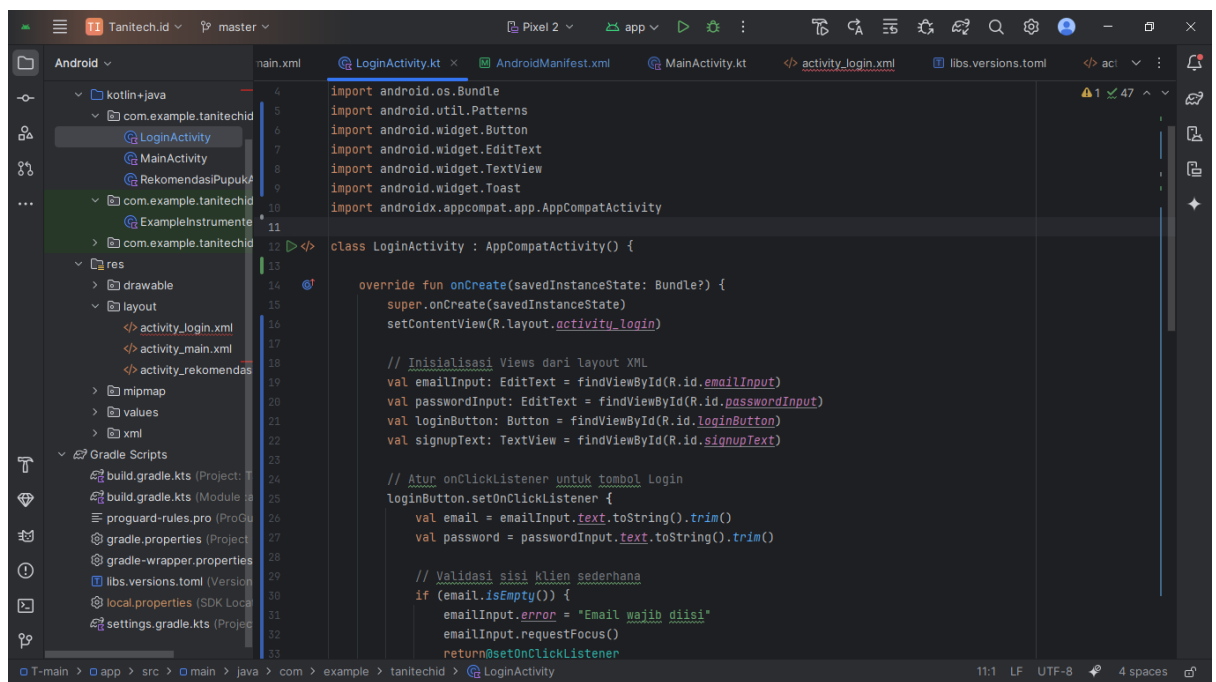
Memberikan komentar deskripsi pada kode layout activity_login.xml agar lebih mudah dalam mencari dan menemukan bagian kode bila melakukan perubahan dan pencariannya.

2. Commit and Push



Memperbaiki kesalahan pada comit and push pada Langkah 1. Muncul pesan error “Design Editor is Unavailable Until After a Successful Project Sync”. Sintaksis TOML yang valid dan muncul ketika Git mencoba menggabungkan perubahan yang saling bertentangan dalam sebuah berkas.

3. Commit and Push



```
import android.os.Bundle
import android.util.Patterns
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class LoginActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        // Inisialisasi Views dari layout XML
        val emailInput: EditText = findViewById(R.id.emailInput)
        val passwordInput: EditText = findViewById(R.id.passwordInput)
        val loginButton: Button = findViewById(R.id.loginButton)
        val signUpText: TextView = findViewById(R.id.signUpText)

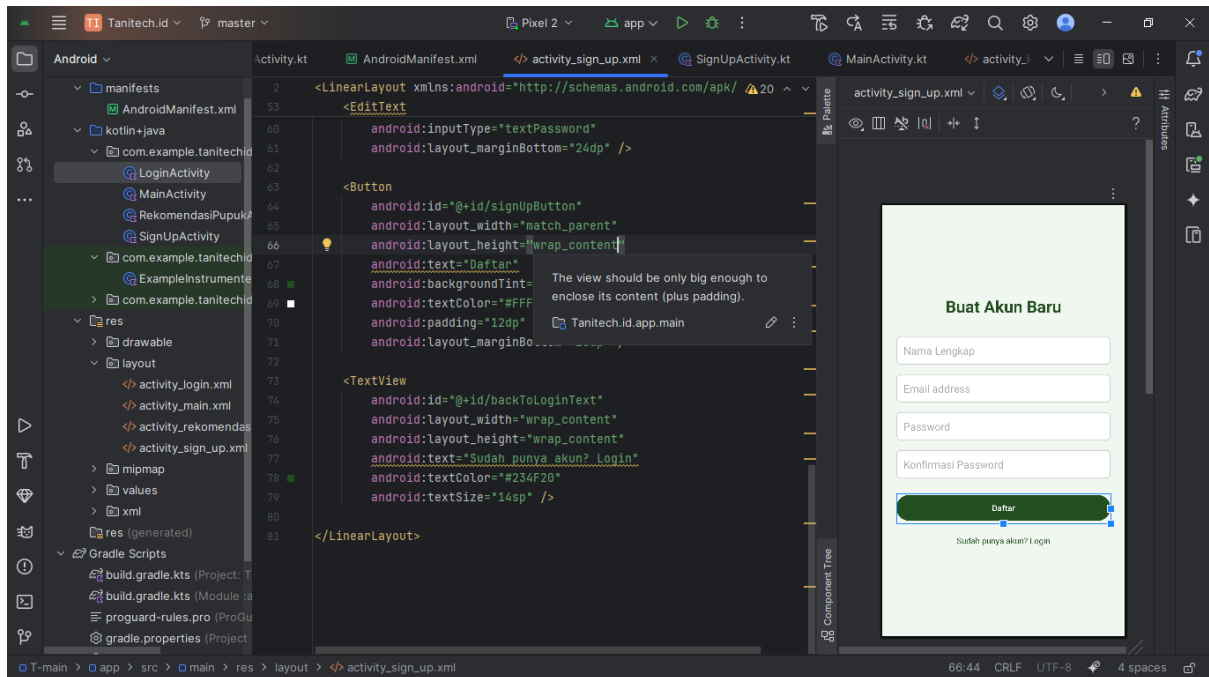
        // Atur onClickListener untuk tombol Login
        loginButton.setOnClickListener {
            val email = emailInput.text.toString().trim()
            val password = passwordInput.text.toString().trim()

            // Validasi sisi klien sederhana
            if (email.isEmpty()) {
                emailInput.error = "Email wajib diisi"
                emailInput.requestFocus()
                return@setOnClickListener
            }
        }
    }
}
```

Kode ini menangani proses login pengguna dengan langkah-langkah berikut:

- **Inisialisasi Tampilan:** Menghubungkan elemen-elemen UI seperti kolom input email dan password, tombol login, serta teks "Sign up" dari layout XML ke kode Kotlin.
- **Validasi Input:** Saat tombol login diklik, kode akan memvalidasi input email dan password. Ini termasuk memastikan bahwa:
 - ✓ Kolom email dan password tidak kosong.
 - ✓ Format email valid.
 - ✓ Panjang password memenuhi persyaratan minimum (misalnya, minimal 6 karakter).
 - ✓ Jika ada masalah validasi, pesan kesalahan akan ditampilkan kepada pengguna.
- **Logika Autentikasi (Contoh):** Setelah validasi berhasil, ada contoh sederhana untuk memeriksa kredensial pengguna (email: "user@example.com", password: "password123").
 - ✓ Jika kredensial cocok, pesan "Login berhasil!" akan muncul. Di aplikasi nyata, Anda akan menggantinya dengan logika autentikasi ke backend server.
 - ✓ Jika tidak cocok, pesan "Email atau password salah" akan ditampilkan.
- **Navigasi:**
 - ✓ Setelah login berhasil, kode ini menunjukkan di mana Anda bisa menambahkan intent untuk berpindah ke aktivitas utama aplikasi (misalnya, HomeActivity).
 - ✓ Ada juga OnClickListener untuk teks "Sign up" yang berfungsi untuk menavigasi pengguna ke layar pendaftaran (SignUpActivity).

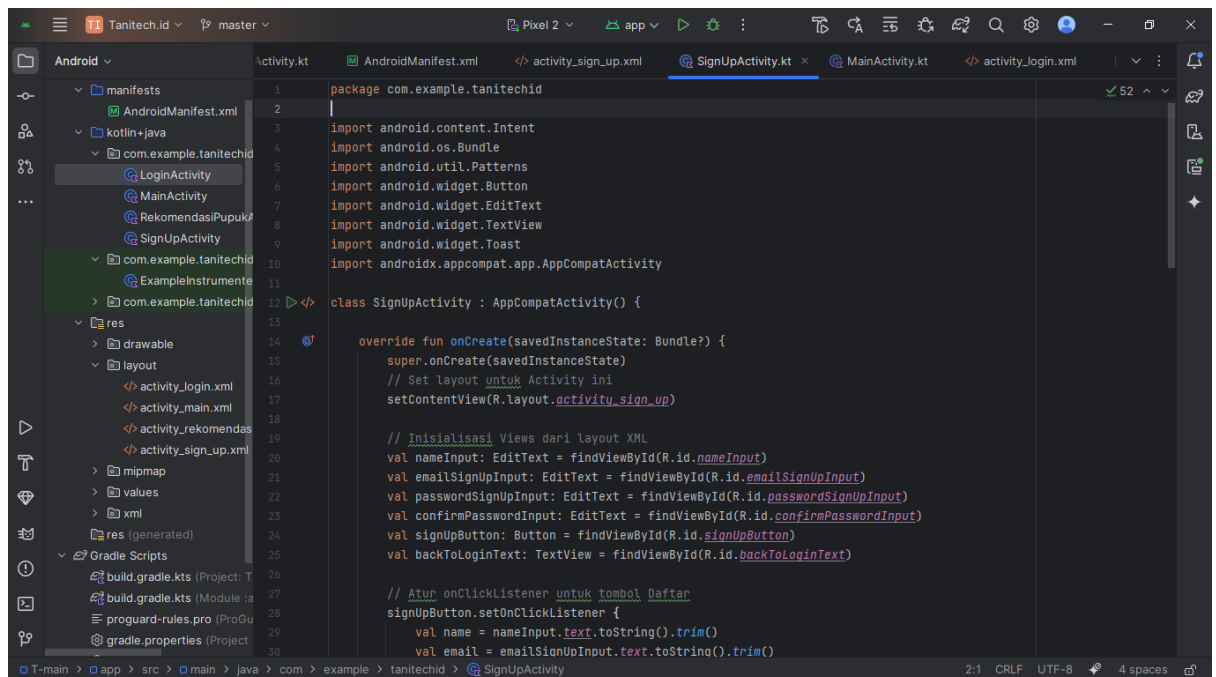
4. Commit and Push : Membuat halaman activity_sign_up



Kode ini mengatur tata letak dan isi dari sebuah halaman pendaftaran, seperti:

- Judul Halaman: Ada teks besar bertuliskan "Buat Akun Baru".
- Kolom Input: Tersedia beberapa kotak isian (mirip formulir) untuk pengguna memasukkan:
 - Nama Lengkap
 - Alamat Email
 - Kata Sandi (Password)
 - Konfirmasi Kata Sandi (untuk memastikan password yang dimasukkan benar).
- Tombol "Daftar": Tombol yang akan diklik pengguna setelah mengisi semua informasi.
- Tautan "Kembali ke Login": Teks yang bisa diklik jika pengguna sudah memiliki akun dan ingin kembali ke halaman login.

5. Commit and Push : Membuat halaman SignUpActivity

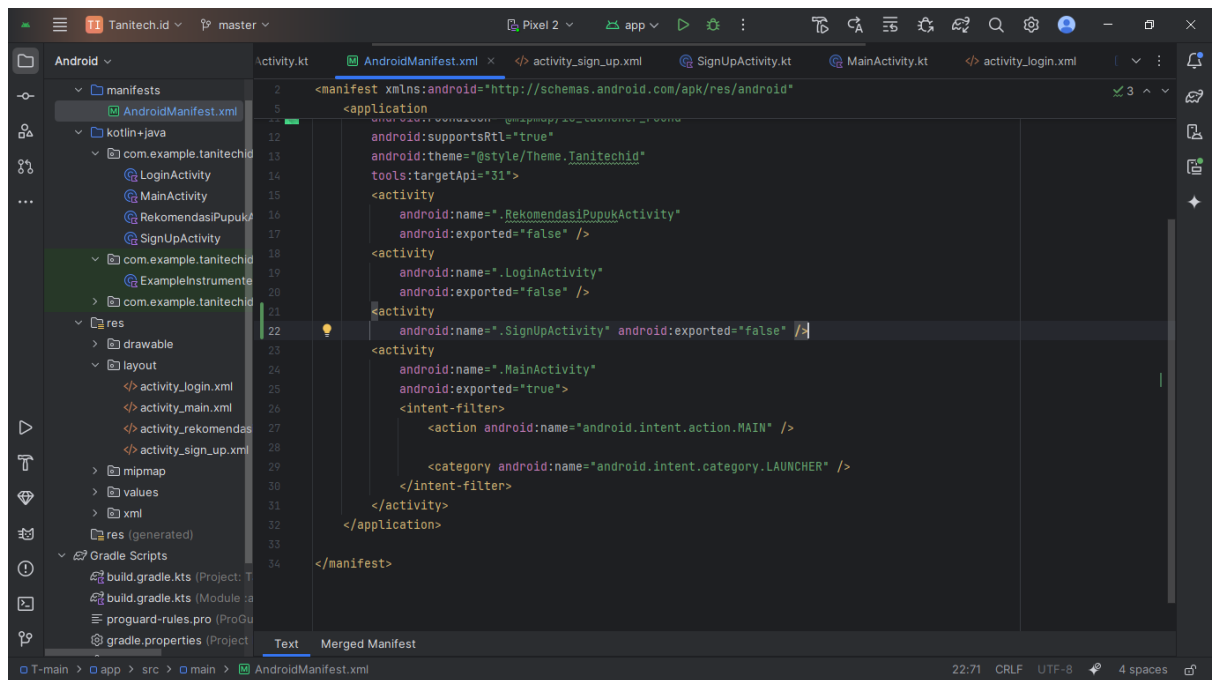


Kode ini membuat layar pendaftaran berfungsi penuh dengan validasi input yang membantu pengguna mengisi data dengan benar, dan kemudian mengarahkan mereka kembali ke layar login.

Kode ini mengatur bagaimana layar "Daftar" berfungsi, meliputi:

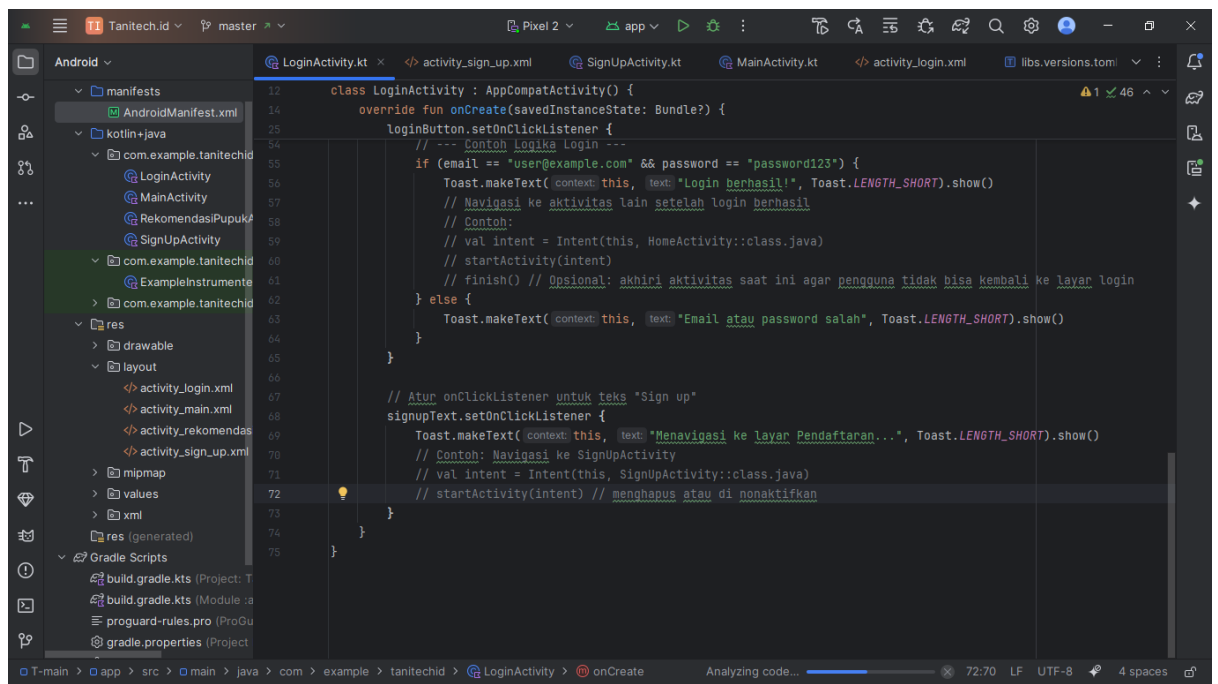
- **Menghubungkan Tampilan:** Kode ini mengambil semua kotak isian (nama, email, password, konfirmasi password), tombol "Daftar", dan teks "Sudah punya akun? Login" yang ada di tampilan (XML) agar bisa digunakan di dalam program.
- **Validasi Otomatis:** Saat pengguna menekan tombol "Daftar", kode ini akan memeriksa setiap informasi yang dimasukkan. Ini memastikan:
 - ✓ Semua kolom penting (nama, email, password) tidak kosong.
 - ✓ Email yang dimasukkan punya format yang benar.
 - ✓ Password minimal 6 karakter.
 - ✓ Password dan konfirmasi password harus sama persis.
 - ✓ Jika ada kesalahan, pesan akan muncul di dekat kolom yang salah.
- **Proses Pendaftaran:** Jika semua informasi benar dan valid, kode akan menampilkan pesan "Pendaftaran berhasil!". Di aplikasi nyata, di bagian inilah data pendaftaran akan dikirim ke server atau disimpan ke database.
- **Navigasi:** Setelah pendaftaran berhasil, atau jika pengguna mengklik "Sudah punya akun? Login", aplikasi akan otomatis membawa pengguna kembali ke layar Login.

6. Commit and Push : Mendaftarkan SignUpActivity di AndroidManifest.xml



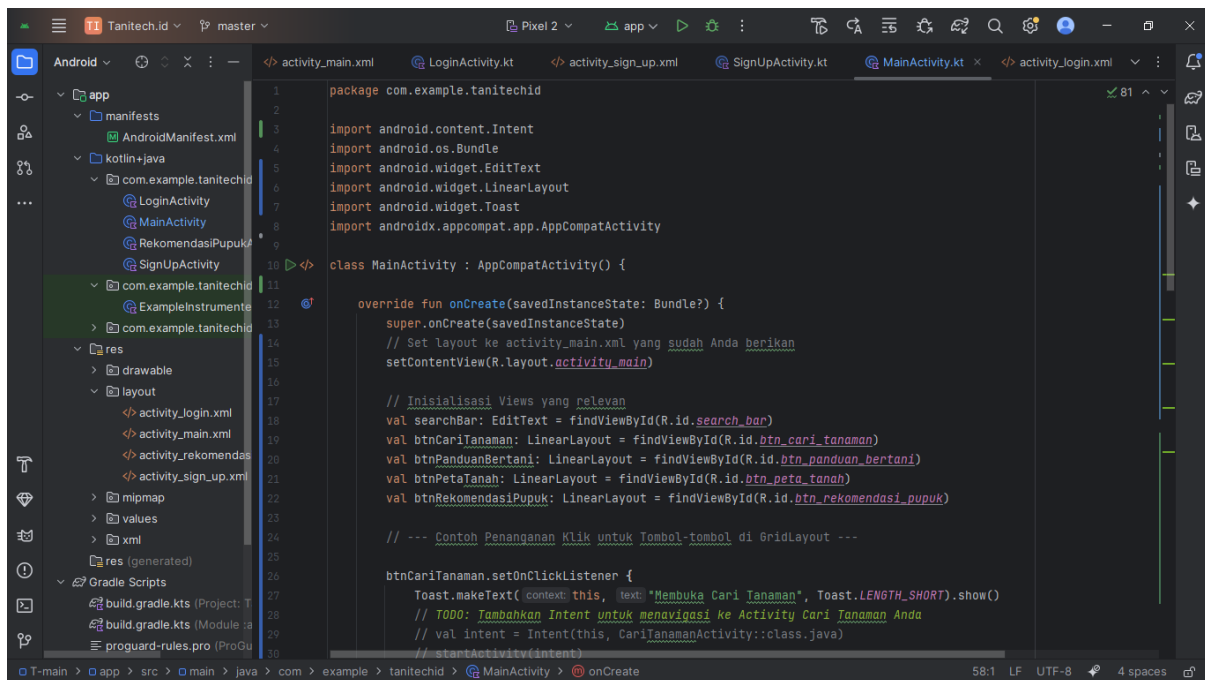
Setiap Activity yang di buat harus dideklarasikan dalam file AndroidManifest.xml agar sistem Android mengetahuinya.

7. Commit and Push : Perbarui LoginActivity untuk Navigasi



SignUpActivity yang berfungsi penuh dengan layout dan logika validasinya.

8. Commit and Push : Memberikan koding pada MainActivity.kt.



1. Mempersiapkan Tampilan (Saat Layar Pertama Kali Muncul)

- `setContentView(R.layout.activity_main)`: Ini adalah perintah pertama. Artinya, "menampilkan desain visual yang sudah dibuat di file `activity_main.xml`." Kode ini menghubungkan dengan tampilan atau *layout* yang sudah ada.

2. Mengenali Tombol-Tombol dan Elemen Lainnya

- `findViewById(...)`: Setelah tampilan muncul, kode ini mulai "mengenali" setiap elemen penting di layar, seperti:
 - Kolom pencarian (`search_bar`).
 - Tombol "Cari Tanaman".
 - Tombol "Panduan Bertani".
 - Tombol "Peta Jenis Tanah".
 - Tombol "Rekomendasi Pupuk".

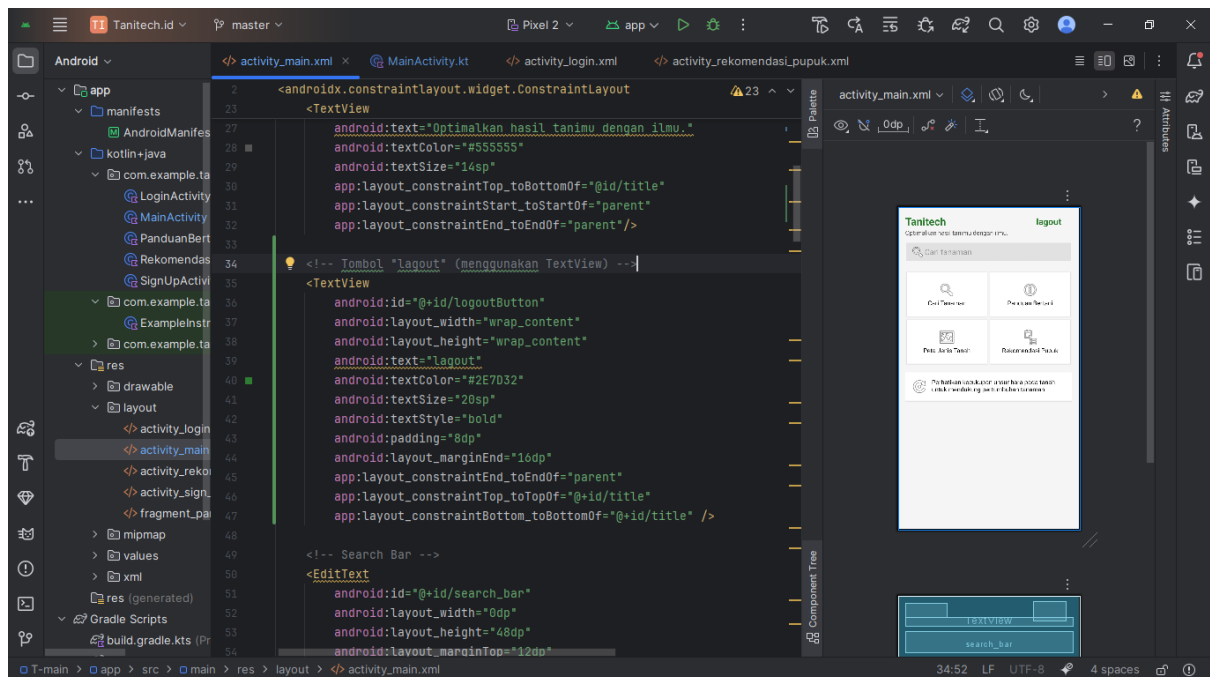
3. Memberi Perintah "Jika Ditekan"

Kode ini menempelkan sebuah "catatan instruksi" pada setiap tombol.

- Untuk Tombol Cari Tanaman, Panduan Bertani, dan Peta Jenis Tanah:
 - Perintahnya: "Jika tombol ini ditekan, tampilkan pesan singkat (Toast) di layar yang memberitahu pengguna fitur apa yang sedang dibuka."

- Untuk Tombol Rekomendasi Pupuk:
 - Perintahnya: "Jika tombol ini ditekan, lakukan dua hal:
 1. Tampilkan pesan singkat 'Membuka Rekomendasi Pupuk'.
 2. Segera buka halaman baru, yaitu halaman Rekomendasi Pupuk Activity.
- Untuk Kolom Pencarian (searchBar):
 - Perintahnya: "Jika kolom ini ditekan, tampilkan pesan 'Mengaktifkan fitur pencarian'.
 - Fungsi pencarian detailnya juga masih dalam tahap perencanaan.

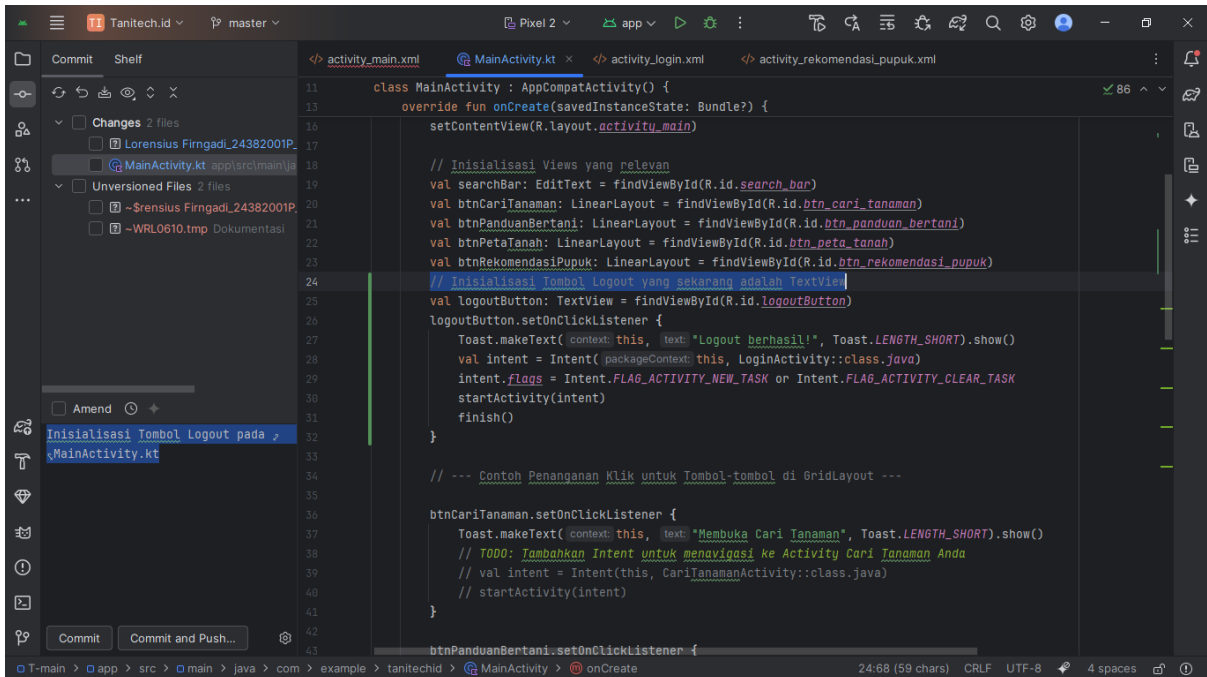
9. Commit and Push : Membuat TextView Logout.



Kode TextView Logout

- Membuat tombol Logout. Jika tombol Logout ditekan, kembalikan pengguna ke halaman Login dan tutup semua halaman sebelumnya agar tidak bisa kembali.
- Mengganti ConstraintLayout >
app:layout_constraintTop_toBottomOf="@id/subtitle". posisi di bawah judul dan subjudul, bukan di bagian paling atas layar.

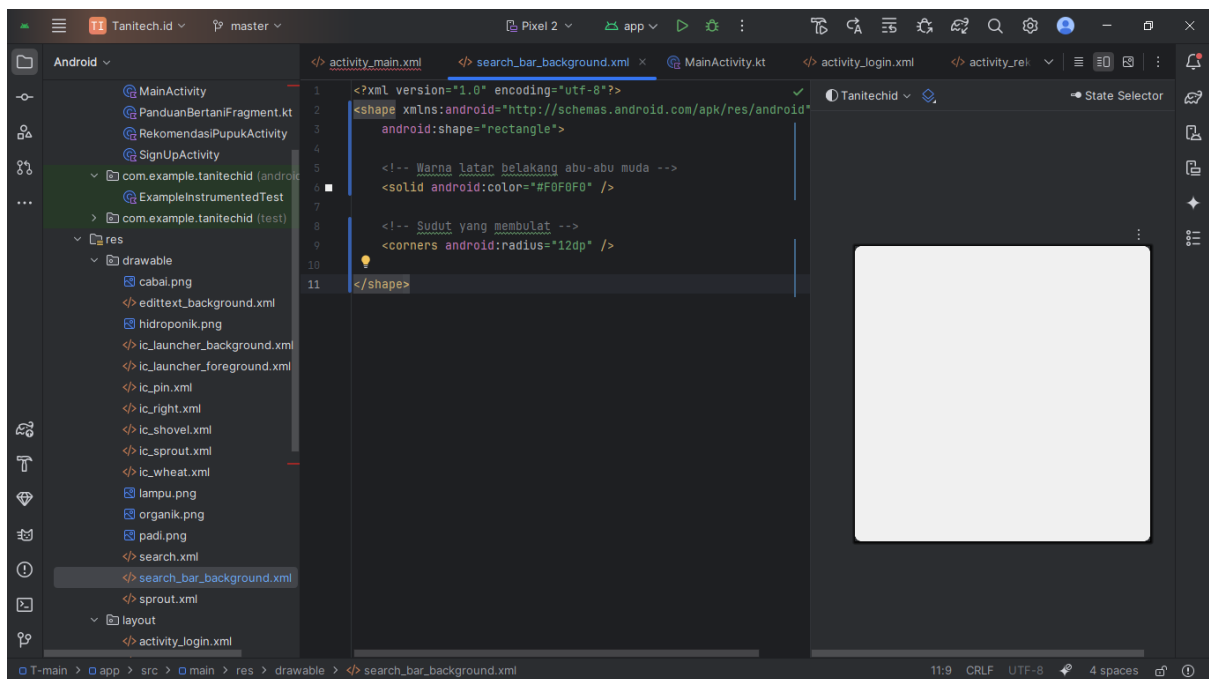
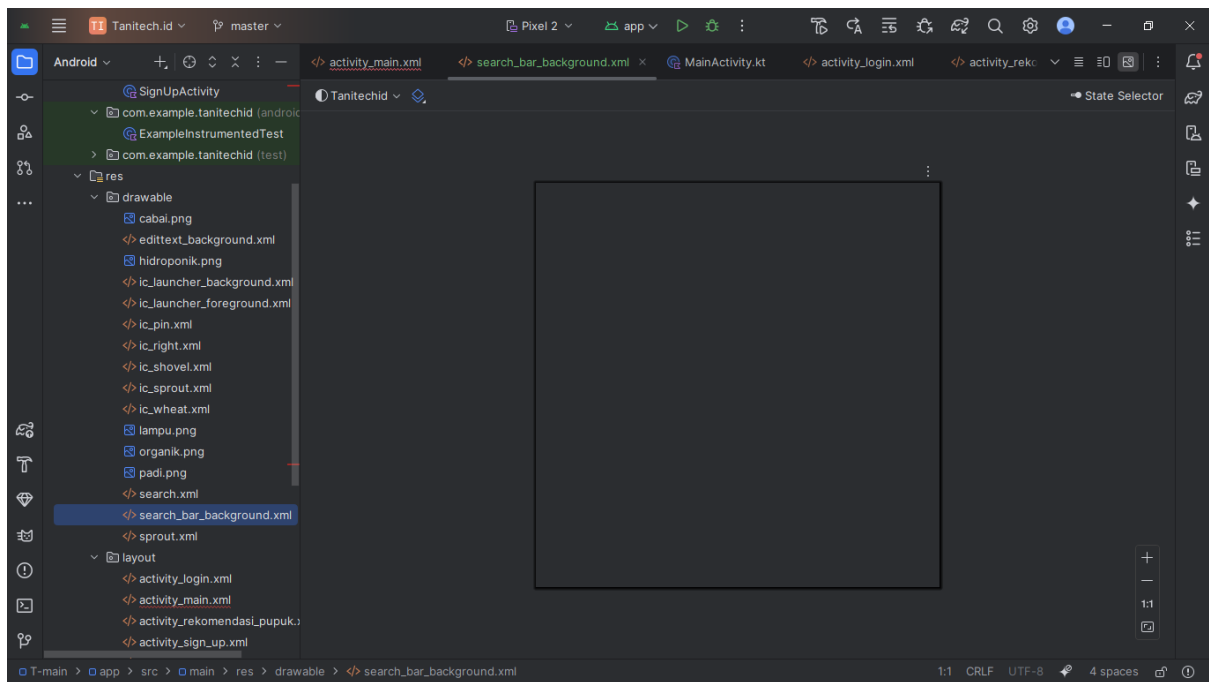
10. Commit and Push : Inisialisasi Tombol Logout pada MainActivity.kt



```
11 class MainActivity : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         setContentView(R.layout.activity_main)
14
15         // Inisialisasi Views yang relevan
16         val searchBar: EditText = findViewById(R.id.search_bar)
17         val btnCariTanaman: LinearLayout = findViewById(R.id.btn_cari_tanaman)
18         val btnPanduanBertani: LinearLayout = findViewById(R.id.btn_panduan_bertani)
19         val btnPetaTanah: LinearLayout = findViewById(R.id.btn_peta_tanah)
20         val btnRekomendasiPupuk: LinearLayout = findViewById(R.id.btn_rekomendasi_pupuk)
21
22         // Inisialisasi Tombol Logout yang sekarang adalah TextView
23         val logoutButton: TextView = findViewById(R.id.logoutButton)
24         logoutButton.setOnClickListener {
25             Toast.makeText(context, this, "Logout berhasil!", Toast.LENGTH_SHORT).show()
26             val intent = Intent(packageContext, LoginActivity::class.java)
27             intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
28             startActivity(intent)
29             finish()
30         }
31
32         // --- Contoh Penanganan Klik untuk Tombol-tombol di GridLayout ---
33
34         btnCariTanaman.setOnClickListener {
35             Toast.makeText(context, this, "Membuka Cari Tanaman", Toast.LENGTH_SHORT).show()
36             // TODO: Tambahkan Intent untuk menavigasi ke Activity Cari Tanaman Anda
37             // val intent = Intent(this, CariTanamanActivity::class.java)
38             // startActivity(intent)
39         }
40
41         btnPanduanBertani.setOnClickListener {
42
43         }
```

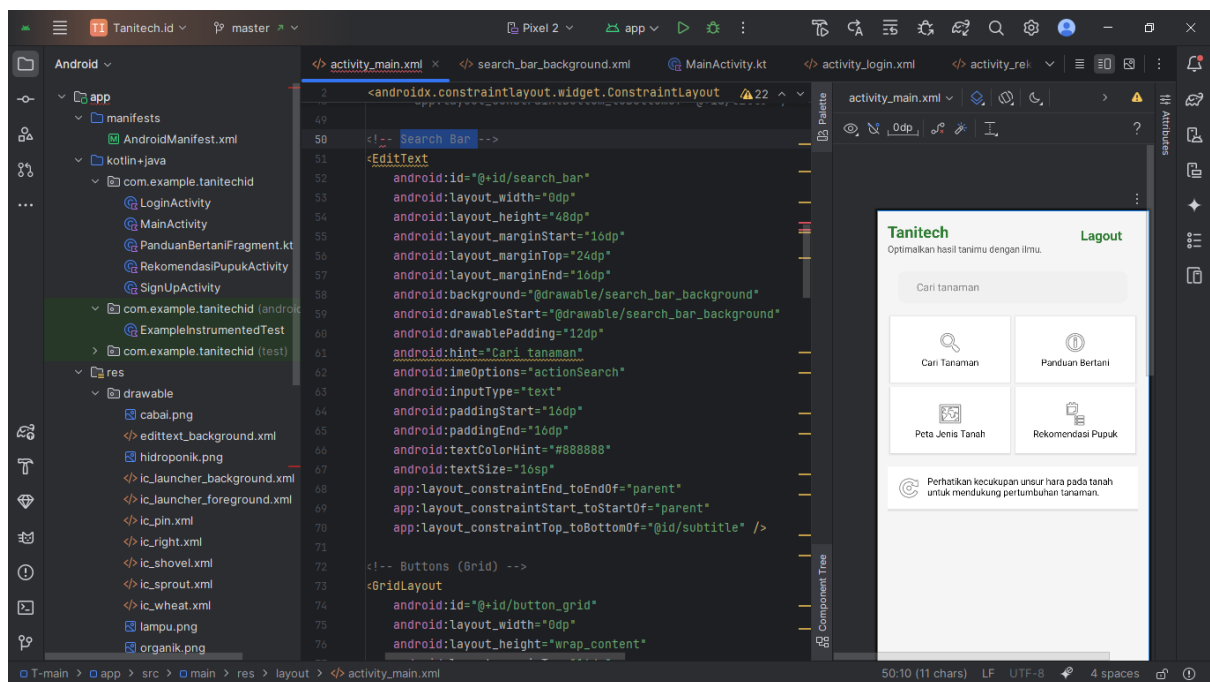
Kode ini membuat tulisan "logout" menjadi tombol fungsional yang, saat disentuh, akan mengeluarkan pengguna dari aplikasi dengan aman, mengembalikannya ke halaman login, dan memastikan pengguna tidak bisa kembali ke halaman utama tanpa login lagi.

11. Commit and Push : Buat File Background Kustom (search_bar_background.xml)



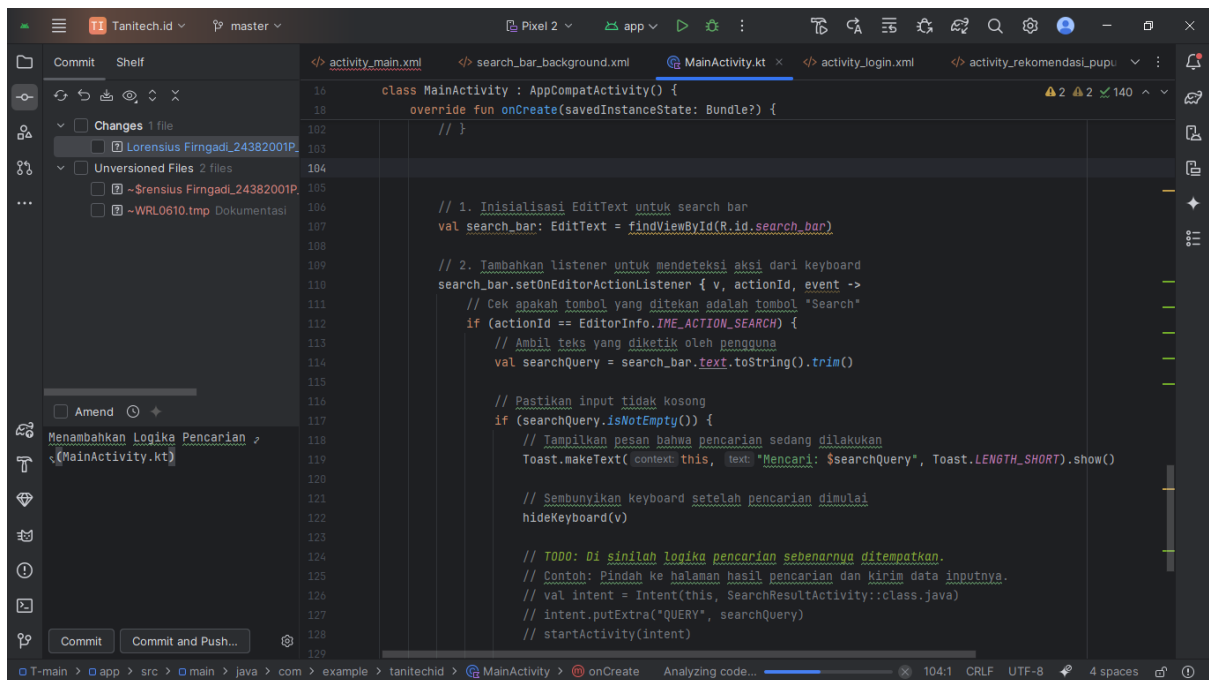
Untuk mendefinisikan sebuah bentuk persegi panjang dengan warna latar #F0F0F0 (abu-abu sangat muda) dan sudut yang melengkung sebesar 12dp.

12. Commit and Push :Perbaikan kode pada Search Bar, EditText di activity_main.xml.



- android:background="@drawable/search_bar_background": Menggunakan background kustom.
- layout_width="0dp" dengan marginStart dan marginEnd: ConstraintLayout untuk membuat elemen memenuhi lebar dengan margin.
- android:drawablePadding="12dp": Memberi jarak antara ikon dan teks.
- android:imeOptions="actionSearch": Mengubah tombol "Enter" di keyboard menjadi ikon "Search" (kaca pembesar).
- android:inputType="text": Memberitahu keyboard bahwa input yang diharapkan adalah teks biasa.

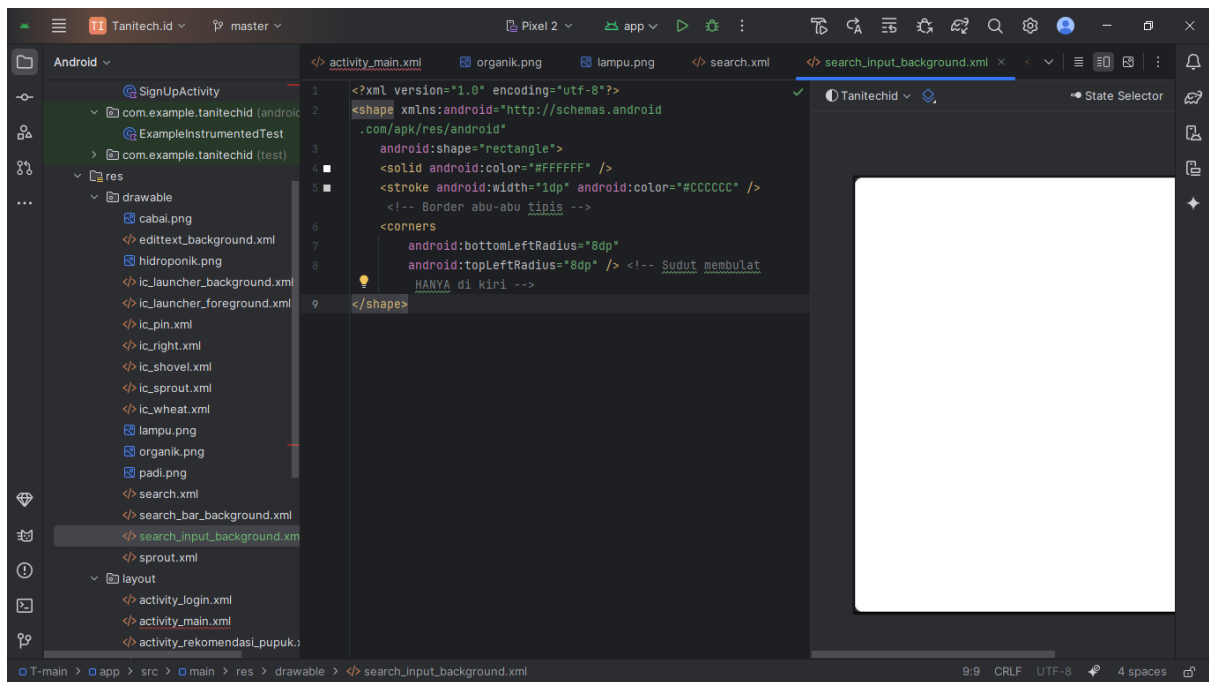
13. Commit and Push : Menambahkan Logika Pencarian (MainActivity.kt)



```
16 class MainActivity : AppCompatActivity() {
17     override fun onCreate(savedInstanceState: Bundle?) {
18         // }
19
20         // 1. Inisialisasi EditText untuk search bar
21         val search_bar: EditText = findViewById(R.id.search_bar)
22
23         // 2. Tambahkan listener untuk mendeteksi aksi dari keyboard
24         search_bar.setOnEditorActionListener { v, actionId, event ->
25             // Cek apakah tombol yang ditekan adalah tombol "Search"
26             if (actionId == EditorInfo.IME_ACTION_SEARCH) {
27                 // Ambil teks yang diketik oleh pengguna
28                 val searchQuery = search_bar.text.toString().trim()
29
30                 // Pastikan input tidak kosong
31                 if (searchQuery.isNotEmpty()) {
32                     // Tampilkan pesan bahwa pencarian sedang dilakukan
33                     Toast.makeText(context, this, text: "Mencari: $searchQuery", Toast.LENGTH_SHORT).show()
34
35                     // Sembunyikan keyboard setelah pencarian dimulai
36                     hideKeyboard(v)
37
38                     // TODO: Di sinilah logika pencarian sebenarnya ditempatkan.
39                     // Contoh: Pindah ke halaman hasil pencarian dan kirim data inputnya.
40                     // val intent = Intent(this, SearchResultActivity::class.java)
41                     // intent.putExtra("QUERY", searchQuery)
42                     // startActivity(intent)
43                 }
44             }
45         }
46     }
47 }
```

- Mendengarkan: Menunggu pengguna menekan tombol "Search" di keyboard ponsel.
- Membaca: Mengambil teks yang diketik oleh pengguna.
- Merespons: Menampilkan pesan singkat (misal: "Mencari: Padi") dan langsung menyembunyikan keyboard agar tampilan kembali rapi.

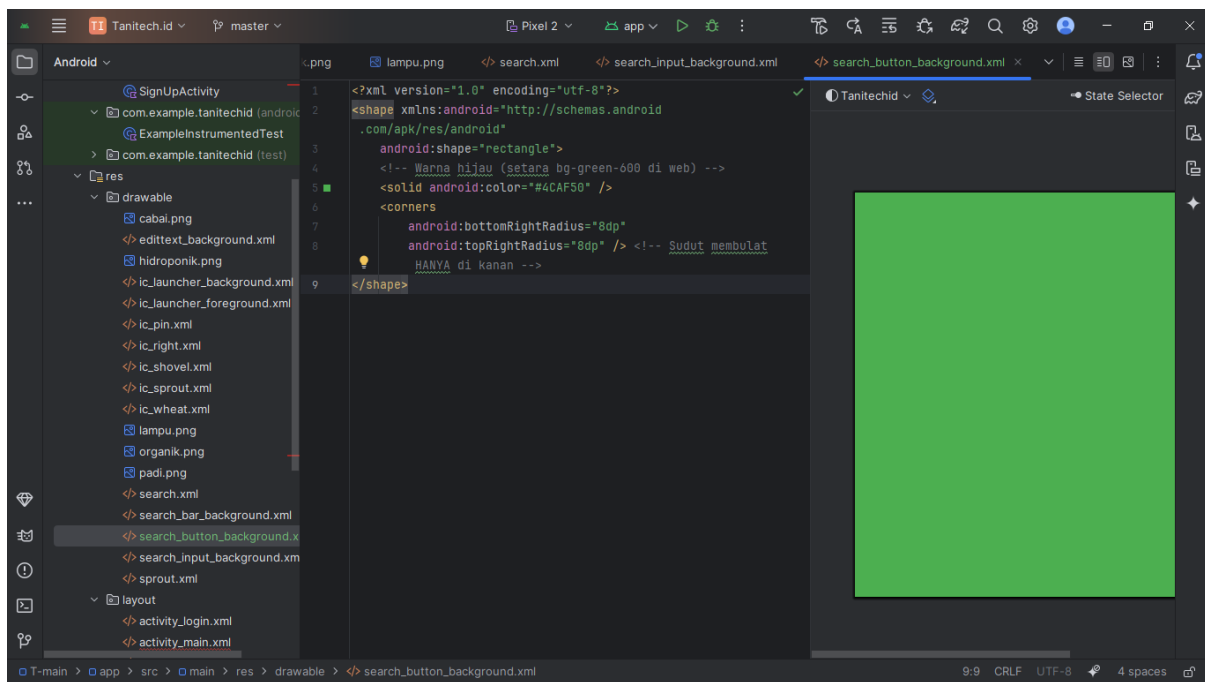
14. Commit and Push : Membuat Background untuk Kotak Input.



Kode XML ini membuat sebuah bentuk persegi panjang (rectangle) dengan beberapa karakteristik visual:

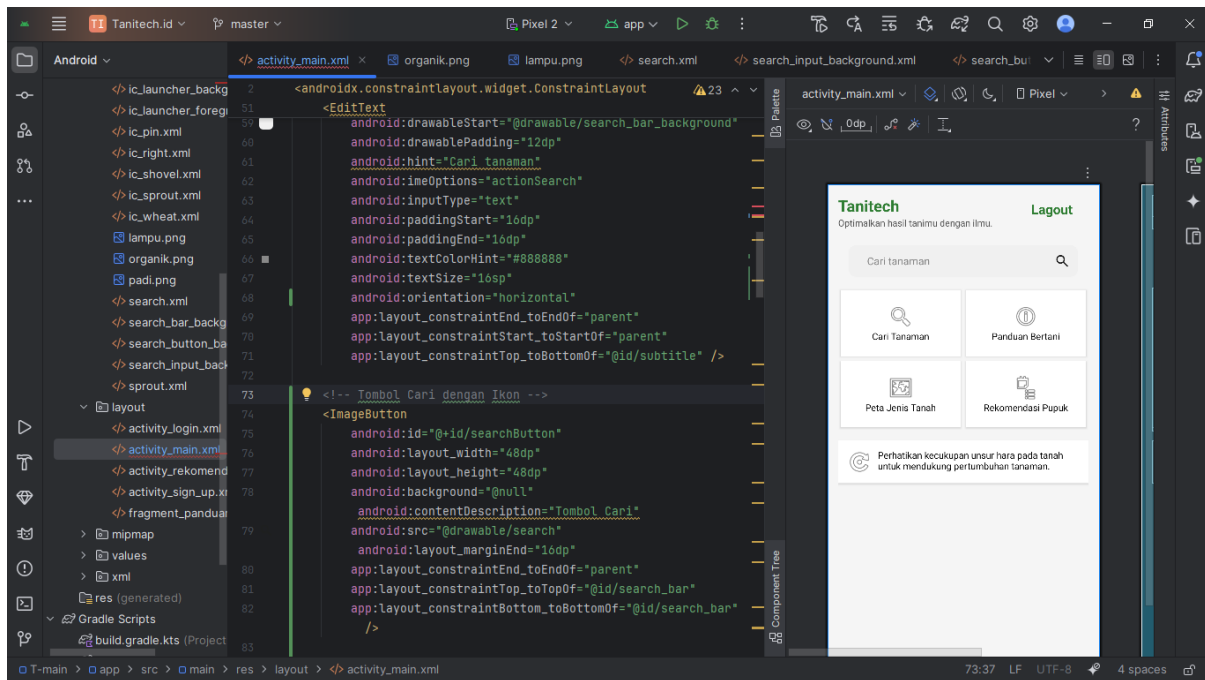
- Bentuk Dasar: `android:shape="rectangle"` menentukan bahwa bentuk yang dibuat adalah persegi panjang.
- Warna Solid: `<solid android:color="#FFFFFF" />` mengisi seluruh area bentuk dengan warna putih (#FFFFFF).
- Garis Tepi (Border): `<stroke android:width="1dp" android:color="#CCCCCC" />` menambahkan garis tepi (border) setebal 1dp (density-independent pixel) dengan warna abu-abu (#CCCCCC) di sekeliling bentuk.
- Sudut Membulat (Rounded Corners):
 - ✓ `<corners android:bottomLeftRadius="8dp" />` memberikan sudut membulat dengan radius 8dp pada bagian kiri bawah.
 - ✓ `<corners android:topLeftRadius="8dp" />` memberikan sudut membulat dengan radius 8dp pada bagian kiri atas.

15. Commit and Push : Buat Background untuk Tombol Cari (search_button_background.xml)



Kode XML ini mendefinisikan sebuah bentuk persegi panjang (rectangle) yang berwarna hijau solid (#4CAF50). Bentuk ini memiliki sudut membulat (rounded corners) hanya di bagian kanan atas (topRightRadius="8dp") dan kanan bawah (bottomRightRadius="8dp"), dengan radius pembulatan sebesar 8dp. Sementara itu, sudut di sisi kirinya tetap tajam.

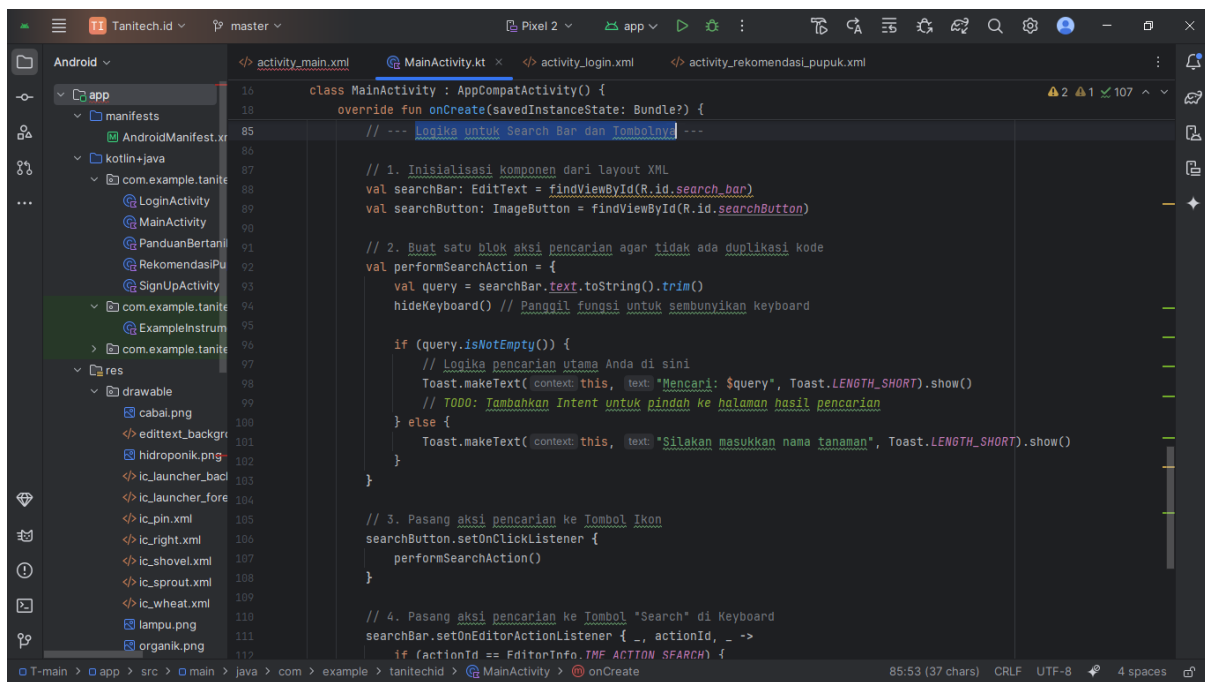
16. Commit and Push : Menambahkan kode Tombol Cari dengan Ikon pada Layout di activity_main.xml.



- `android:id="@+id/searchButton"`: Ini adalah pengidentifikasi unik (ID) untuk tombol ini. Dengan ID ini, Anda bisa merujuk dan berinteraksi dengan tombol ini dari kode Java atau Kotlin Anda (misalnya, untuk menambahkan event listener saat tombol diklik).
- `android:layout_width="48dp"` dan `android:layout_height="48dp"`: Atribut ini mengatur ukuran tombol. Dalam kasus ini, tombol memiliki lebar dan tinggi 48dp (density-independent pixels), yang merupakan ukuran umum untuk ikon atau tombol aksi.
- `android:background="@null"`: Atribut ini mengatur latar belakang tombol. Nilai `@null` berarti tombol ini tidak memiliki latar belakang solid yang terlihat. Ini sering digunakan ketika Anda ingin ikon tampil murni tanpa box latar belakang yang mengelilinginya.
- `android:contentDescription="Tombol Cari"`: Ini adalah deskripsi konten untuk tujuan aksesibilitas. Pembaca layar (screen reader) akan membacakan teks ini kepada pengguna yang memiliki gangguan penglihatan, memberikan informasi tentang fungsi tombol.
- `android:src="@drawable/search"`: Atribut ini menentukan gambar atau ikon yang akan ditampilkan di dalam ImageButton. Dalam kasus ini, tombol akan menampilkan drawable yang bernama `search` (misalnya, ikon kaca pembesar untuk pencarian) yang harus ada di folder `res/drawable` proyek Android Anda.
- `android:layout_marginEnd="16dp"`: Ini adalah margin (jarak) dari tepi kanan elemen induknya sebesar 16dp.

- `app:layout_constraintEnd_toEndOf="parent"`: Ini adalah batasan ConstraintLayout. Atribut ini menempatkan ujung kanan tombol sejajar dengan ujung kanan dari parent layout-nya.
- `app:layout_constraintTop_toTopOf="@id/search_bar"`: Batasan ini menempatkan bagian atas tombol sejajar dengan bagian atas dari elemen dengan ID `search_bar` (dalam konteks ini, kemungkinan besar adalah EditText atau search bar Anda).
- `app:layout_constraintBottom_toBottomOf="@id/search_bar"`: Batasan ini menempatkan bagian bawah tombol sejajar dengan bagian bawah dari elemen dengan ID `search_bar`.

17. Commit and Push : Logika untuk Search Bar dan Tombolnya



```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        // --- Logika untuk Search Bar dan Tombolnya ---

        // 1. Inisialisasi komponen dari layout XML
        val searchBar: EditText = findViewById(R.id.search_bar)
        val searchButton: ImageButton = findViewById(R.id.searchButton)

        // 2. Buat satu blok aksi pencarian agar tidak ada duplikasi kode
        val performSearchAction = {
            val query = searchBar.text.toString().trim()
            hideKeyboard() // Panggil fungsi untuk sembunyikan keyboard

            if (query.isNotEmpty()) {
                // Logika pencarian utama Anda di sini
                Toast.makeText(context, "Mencari: $query", Toast.LENGTH_SHORT).show()
                // TODO: Tambahkan Intent untuk pindah ke halaman hasil pencarian
            } else {
                Toast.makeText(context, "Silakan masukkan nama tanaman", Toast.LENGTH_SHORT).show()
            }
        }

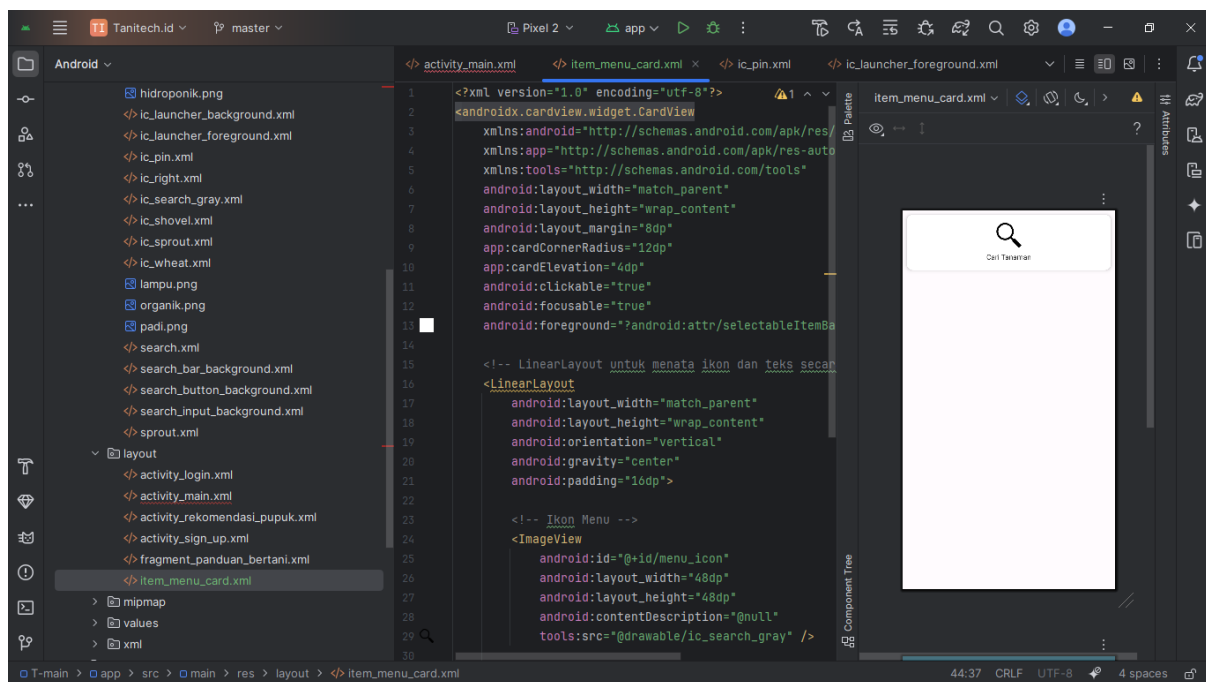
        // 3. Pasang aksi pencarian ke Tombol Ikon
        searchButton.setOnClickListener {
            performSearchAction()
        }

        // 4. Pasang aksi pencarian ke Tombol "Search" di Keyboard
        searchBar.setOnEditorActionListener { _, actionId, _ ->
            if (actionId == EditorInfo.IME_ACTION_SEARCH) {
                performSearchAction()
            }
        }
    }
}

```

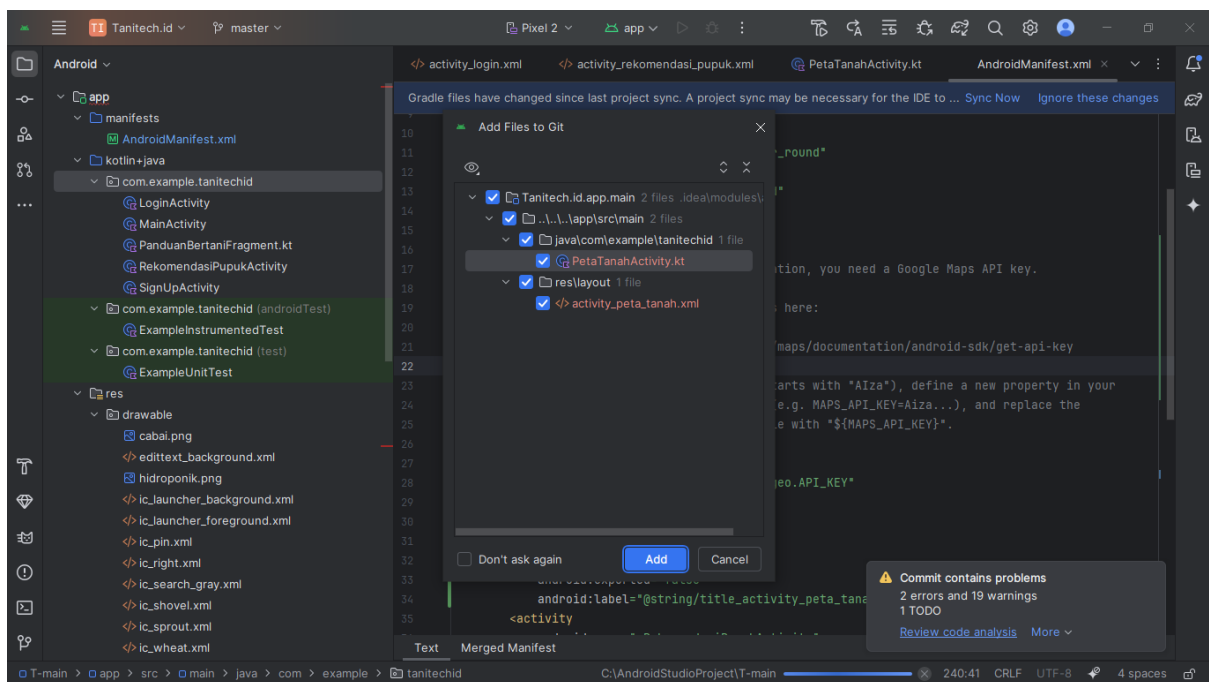
Perbaiki koding untuk menghubungkan dua elemen visual (kotak teks dan tombol ikon) dan dua aksi pengguna (klik tombol dan tekan keyboard) ke satu blok logika pencarian yang sama, sehingga menciptakan fitur yang konsisten, efisien, dan rapi.

18. Commit and Push : Membuat layout untuk satu kartu menu (item_menu_card.xml)



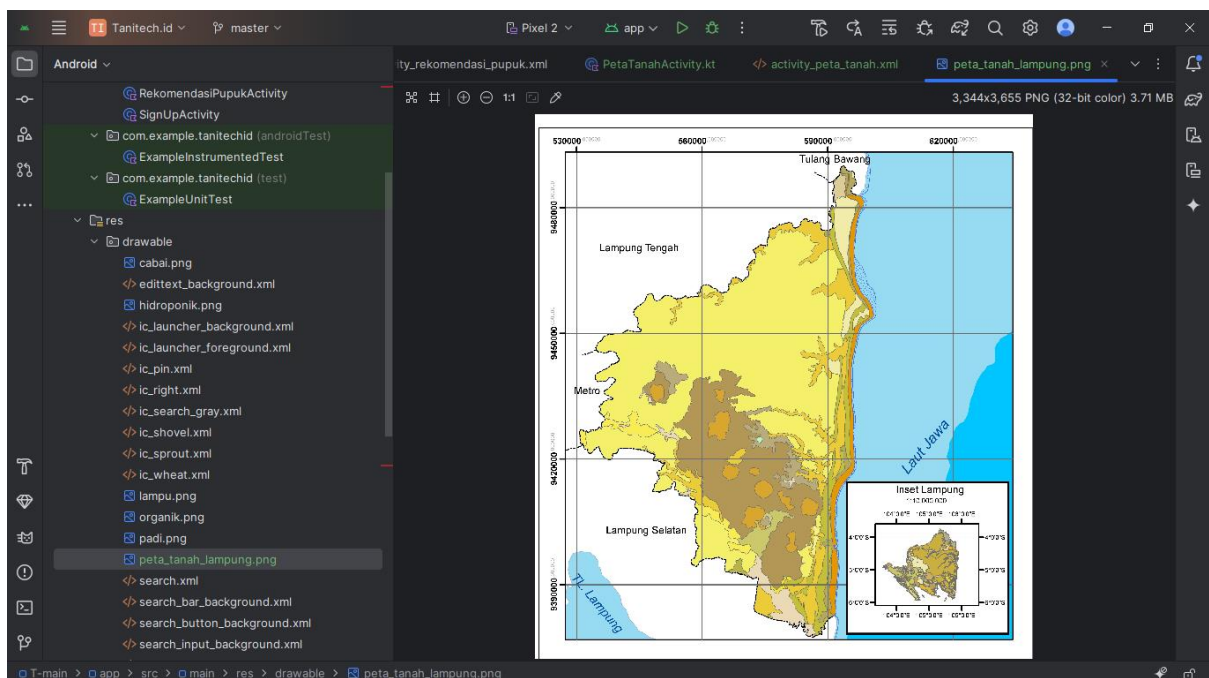
Kode ini membuat sebuah kartu interaktif yang dapat diklik. Kartu ini memiliki sudut membulat dan bayangan (shadow), memberikan tampilan yang menarik. Di dalamnya, kartu berisi sebuah ikon (ikon pencarian) dan teks di bawahnya ("Cari Tanaman"), keduanya ditata secara vertikal dan berada di tengah kartu. Ketika disentuh, kartu akan menampilkan efek riak (ripple effect) sebagai umpan balik visual.

19. Commit and Push : Membuat Halaman Peta (PetaTanahActivity)



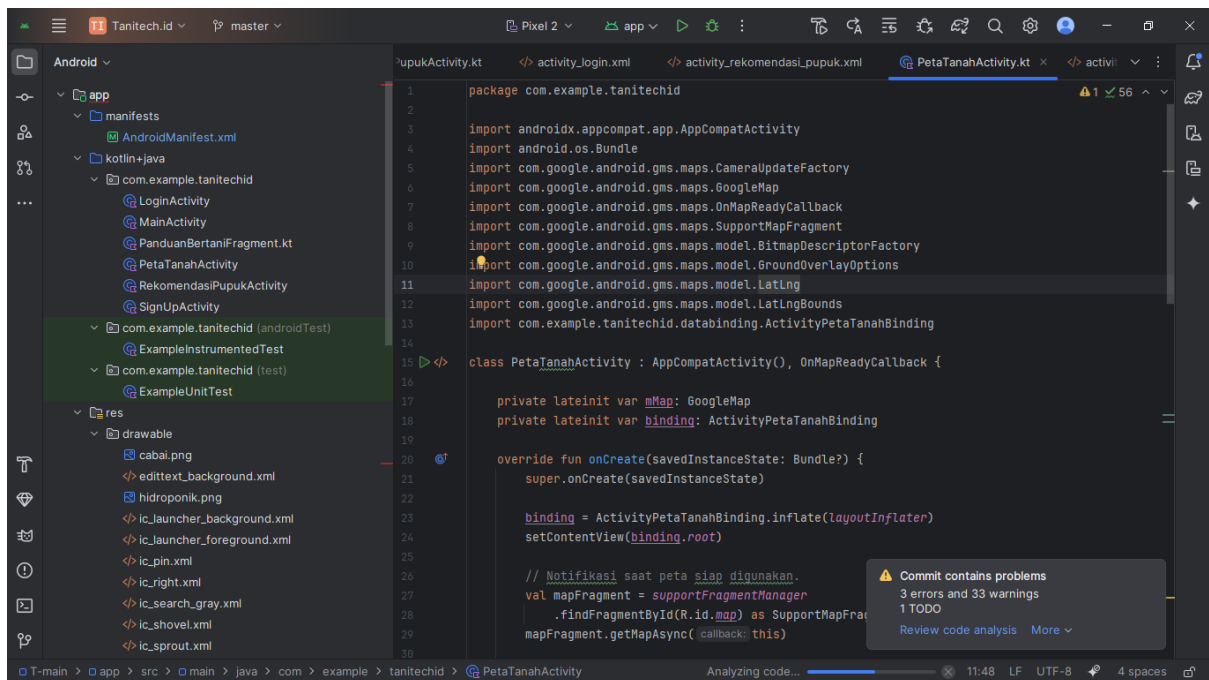
Membuat halaman mengikuti activity_main

20. Commit and Push : Menambahkan peta jenis tanah Lampung



Ini file gambar untuk mendukung halaman activity_peta_tanah.xml

21. Commit and Push : Input koding PetaTanahActivity



Kode ini mendefinisikan sebuah *activity* (layar) pada aplikasi Android yang berfungsi untuk menampilkan peta jenis tanah menggunakan Google Maps.

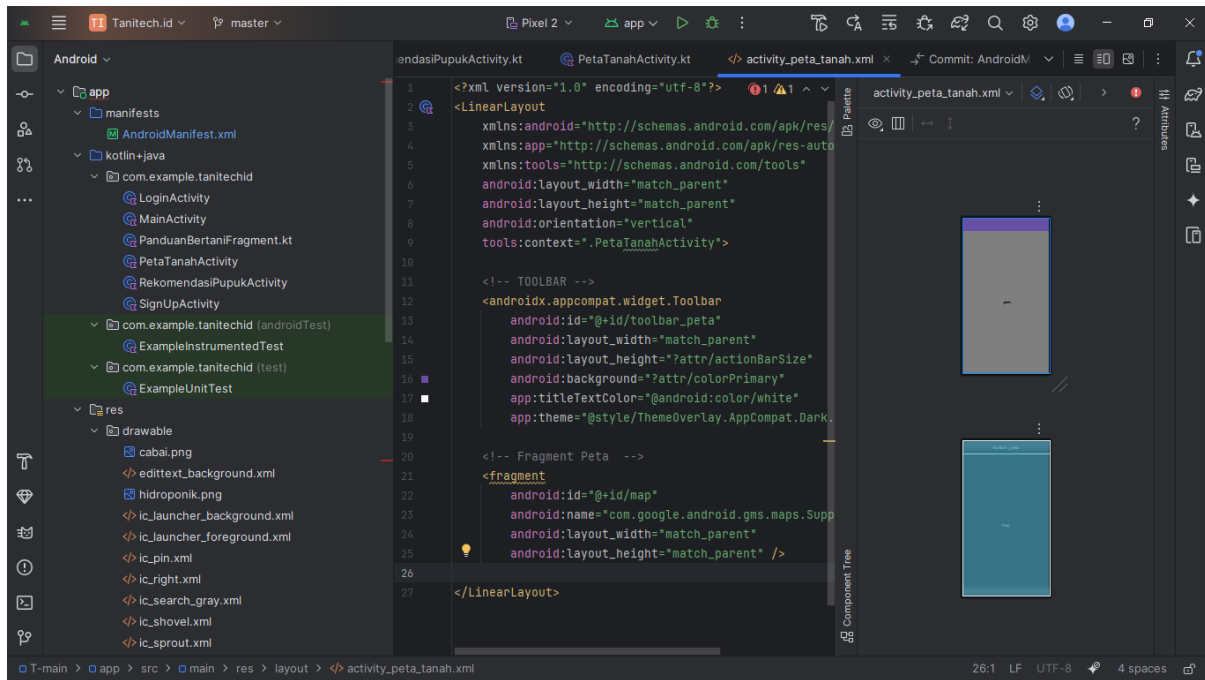
Fitur utamanya adalah menempatkan sebuah gambar kustom (dalam kasus ini, `peta_tanah_lampung.png`) di atas peta Google Maps pada lokasi geografis yang spesifik. Ini disebut *Ground Overlay*.

Secara ringkas, alur kerja kode ini adalah sebagai berikut:

1. **Inisialisasi:** Saat layar dibuka (`onCreate`), kode ini menyiapkan tampilan, mengaktifkan Toolbar dengan judul "Peta Jenis Tanah" dan tombol kembali, serta memulai proses pemuatan peta Google Maps secara asinkron.
2. **Peta Siap:** Setelah Google Maps berhasil dimuat, fungsi `onMapReady` akan dipanggil.
3. **Menambahkan Overlay:** Di dalam `onMapReady`, kode melakukan hal-hal berikut:
 - o **Menentukan Area:** Menetapkan koordinat geografis (Latitude & Longitude) untuk sudut Barat Daya dan Timur Laut, yang mendefinisikan sebuah area persegi panjang di peta (dalam contoh ini, area di sekitar Lampung).
 - o **Menyiapkan Gambar:** Mengambil gambar `peta_tanah_lampung` dari *resource* aplikasi.
 - o **Menempatkan Gambar:** "Menempelkan" gambar tersebut ke area yang telah ditentukan di peta dan memberinya sedikit transparansi agar peta dasar masih terlihat.
4. **Fokus Kamera:** Secara otomatis mengarahkan tampilan kamera peta ke tengah area gambar tersebut dengan tingkat zoom yang sesuai, sehingga pengguna langsung melihat peta tanah yang ditambahkan.

5. Penanda (Marker): Menambahkan sebuah penanda di tengah area sebagai informasi tambahan.
6. Navigasi: Mengatur fungsionalitas tombol kembali pada Toolbar agar pengguna bisa kembali ke layar sebelumnya.

22. Commit and Push : Menambahkan koding pada activity_peta_tanah.xml



Kode ini adalah file layout XML yang mendefinisikan tampilan antarmuka (UI) untuk PetaTanahActivity. File ini berfungsi sebagai "cetak biru" visual, yang mengatur posisi dan tampilan elemen-elemen di layar.

Struktur utamanya menggunakan LinearLayout dengan orientasi vertikal, yang berarti semua komponen di dalamnya akan disusun dari atas ke bawah.

Komponen utama dalam layout ini adalah:

1. `androidx.appcompat.widget.Toolbar`:
 - o Fungsi: Ini adalah Toolbar atau bilah aplikasi yang ditempatkan di bagian paling atas layar.
 - o ID: Diberi ID `toolbar_peta`, yang digunakan oleh kode Kotlin untuk mengaturnya sebagai *Action Bar* (memberi judul, tombol kembali, dll.).
 - o Tampilan: Didesain untuk memiliki tinggi standar, menggunakan warna utama aplikasi (`colorPrimary`) sebagai latar belakang, dan menampilkan teks judul berwarna putih.
2. `fragment`:

- Fungsi: Ini adalah wadah (container) khusus yang ditempatkan di bawah Toolbar dan mengisi sisa ruang layar.
- Tipe: Secara spesifik didefinisikan sebagai `com.google.android.gms.maps.SupportMapFragment`. Ini adalah komponen dari Google Maps SDK yang bertugas untuk menampilkan dan mengelola peta interaktif.
- ID: Diberi ID map, yang sangat penting agar kode Kotlin dapat menemukan wadah ini dan memuat peta Google ke dalamnya.

Hubungan dengan Kode Kotlin (PetaTanahActivity.kt)

- `findViewById(R.id.toolbar_peta)` dalam kode Kotlin merujuk langsung ke Toolbar yang didefinisikan di XML ini.
- `supportFragmentManager.findFragmentById(R.id.map)` merujuk ke fragment peta di XML ini, yang memungkinkan kode Kotlin untuk mengontrol peta tersebut (misalnya, menambahkan overlay, marker, dan mengatur kamera).