

LAPORAN RESPONSI PRAKTIKUM PEMROGRAMAN BERBASIS MOBILE



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya
NIM : 195410257
JURUSAN : Informatika
JENJANG : S1
KELAS : 5

Laboratorium Terpadu

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

UTDI

YOGYAKARTA

2021/2022

LAPORAN INI TERDIRI DARI DUA BAGIAN:

1. File-File yang Dimodifikasi
2. Tampilan Aplikasi Ketika Dijalankan

Bagian 1: File-File yang Dimodifikasi

1. MarsApiService.kt

Retrofit membuat network API untuk aplikasi android berdasarkan konten dari sebuah layanan web. Retrofit mengambil data dari layanan web dan mengarahkannya melalui converter library yang mendekode data dan me-return data tersebut dalam bentuk display objek. Retrofit menyertakan support untuk format data web seperti XML dan JSON. Retrofit menciptakan sebagian besar lapisan jaringan, termasuk detail penting seperti menjalankan permintaan pada background thread.

```
package com.example.android.marsrealestate.network

import com.squareup.moshi.Moshi
import com.squareup.moshi.kotlin.reflect.KotlinJsonAdapterFactory
import retrofit2.Retrofit
import retrofit2.converter.moshi.MoshiConverterFactory
import retrofit2.http.GET
import retrofit2.http.Query

enum class MarsApiFilter(val value: String) {
    SHOW_CUKUP("cukup"),
    SHOW_KURANG("kurang"),
    SHOW_SEMUA("semua") }

private const val BASE_URL = "https://darmanto.akakom.ac.id/"

/**
 * Build the Moshi object that Retrofit will be using, making sure to add the Kotlin
 * adapter for
 * full Kotlin compatibility.
 */
private val moshi = Moshi.Builder()
    .add(KotlinJsonAdapterFactory())
    .build()

/**
 * Use the Retrofit builder to build a retrofit object using a Moshi converter with our
 * Moshi
 * object.
 */
private val retrofit = Retrofit.Builder()
    .addConverterFactory(MoshiConverterFactory.create(moshi))
    .baseUrl(BASE_URL)
    .build()

/**
 * A public interface that exposes the [getProperties] method
 */
interface MarsApiService {
    /**
     * Returns a Coroutine [List] of [MarsProperty] which can be fetched in a Coroutine
     * scope.
     * The @GET annotation indicates that the "realestate" endpoint will be requested
     * with the GET
     * HTTP method
     */
    @GET("android/responsi/")
    suspend fun getProperties(@Query("filter") type: String): List<MarsProperty>
}
```

```
/**
 * A public Api object that exposes the lazy-initialized Retrofit service
 */
object MarsApi {
    val retrofitService : MarsApiService by lazy {
        retrofit.create(MarsApiService::class.java)
    }
}
```

penjelasan:

pertama, saya buat enum class. Tepat di bawah impor, buat enum yang disebut MarsApiFilter untuk menentukan konstanta yang cocok dengan nilai kueri yang diharapkan layanan web. Kotlin enum berisikan daftar nama dari konstanta. Enumeration pada kotlin memiliki tipe khusus yang mengindikasikan sesuatu yang memiliki angka atau nilai yang mungkin. Kotlin enum sendiri berbentuk kelas.

Kedua, saya set base url ke <https://darmanto.akakom.ac.id/android/responsi/>. Base URL berfungsi untuk menghasilkan root URL dari aplikasi. Kapan kita harus menggunakan base_url() ? Fungsi base_url() biasanya sering kita pakai untuk membuat URL atau link file statis seperti gambar, CSS, javascript, video, dan file yang diupload. Lalu, kita gunakan anotasi @GET untuk menentukan jalur titik akhir metode layanan web tersebut dengan menuliskan sisa baris url yaitu android/responsi/.

2. MarsProperty.kt

```
package com.example.android.marsrealestate.network

import android.os.Parcelable
import com.squareup.moshi.Json
import kotlinx.android.parcel.Parcelize

/**
 * This data class defines a Mars property which includes an ID, the image URL, the type
 * (sale or rental) and the price (monthly if it's a rental).
 * The property names of this data class are used by Moshi to match the names of values
 * in JSON.
 */
@Parcelize
data class MarsProperty(
    val nama: String,
    // used to map img_src from the JSON to imgSrcUrl in our class
    @Json(name = "foto") val imgSrcUrl: String,
    val umur: Int,
    val username: String): Parcelable {
    val isCukup
        get() = umur >= 25
}
```

penjelasan:

Kelas MarsProperty menentukan struktur data untuk setiap properti yang disediakan oleh layanan web. Di file ini kita menambahkan beberapa logika ke kelas MarsProperty dengan memberi value ke setiap data JSON seperti "nama", "foto", "umur", "username", dan "umur". Kemudian, kita buat variabel isCukup yang di dalamnya ada getter kustom untuk isCukup yang mengembalikan nilai true jika objek umur bernilai 25 keatas. Di data class tersebut kita juga menggunakan Parcelable yaitu suatu interface pada pemrograman Android, yang memungkinkan suatu instansi dari kelas untuk bisa disimpan dan diambil kembali dari Parcel.

3. grid_view_item.xml

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```

xmlns:tools="http://schemas.android.com/tools">
<data>
    <variable
        name="property"
        type="com.example.android.marsrealestate.network.MarsProperty" />
    <import type="android.view.View"/>
</data>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="170dp">
    <ImageView
        android:id="@+id/mars_image"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:adjustViewBounds="true"
        android:padding="2dp"
        app:imageUrl="@{property.imgSrcUrl}"
        tools:src="@tools:sample/backgrounds/scenic"/>

    <TextView
        android:id="@+id/mars_property_type"
        android:layout_width="105dp"
        android:layout_height="45dp"
        android:layout_gravity="bottom|end"
        android:adjustViewBounds="true"
        android:padding="5dp"
        android:scaleType="fitCenter"
        android:text="Kurang"
        android:textSize="25dp"
        android:textColor="#FFFFFF"
        android:visibility="@{property.cukup ? View.GONE : View.VISIBLE}" />

    <TextView
        android:id="@+id/mars_property_type2"
        android:layout_width="105dp"
        android:layout_height="45dp"
        android:layout_gravity="bottom|end"
        android:adjustViewBounds="true"
        android:padding="5dp"
        android:scaleType="fitCenter"
        android:text="Cukup"
        android:textSize="25dp"
        android:textColor="#FFFFFF"
        android:visibility="@{property.cukup ? View.VISIBLE : View.GONE}" />
</FrameLayout>
</layout>

```

Penjelasan:

Yang saya ganti di sini adalah di bagian text yang ditampilkan di halaman grid utama yang awalnya dari “Dijual” dan “Disewa” menjadi “Kurang” dan “Cukup”. Arti dari baris kode tersebut adalah jika kondisi properti memenuhi cukup (≥ 25) maka textview “Cukup” akan ditampilkan (VISIBLE) dan jika tidak memenuhi kondisi cukup maka textview “Kurang” tidak akan ditampilkan (GONE).

4. OverviewViewModel.kt

```

package com.example.android.marsrealestate.overview

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.android.marsrealestate.network.MarsApi
import com.example.android.marsrealestate.network.MarsApiFilter
import com.example.android.marsrealestate.network.MarsProperty
import kotlinx.coroutines.launch

```

```

enum class MarsApiStatus { LOADING, ERROR, DONE }

/**
 * The [ViewModel] that is attached to the [OverviewFragment].
 */
class OverviewViewModel : ViewModel() {

    // The internal MutableLiveData that stores the status of the most recent request
    private val _status = MutableLiveData<MarsApiStatus>()

    // The external immutable LiveData for the request status
    val status: LiveData<MarsApiStatus>
        get() = _status

    // Internally, we use a MutableLiveData, because we will be updating the List of
    // MarsProperty
    // with new values
    private val _properties = MutableLiveData<List<MarsProperty>>()

    // The external LiveData interface to the property is immutable, so only this class
    // can modify
    val properties: LiveData<List<MarsProperty>>
        get() = _properties

    private val _navigateToSelectedProperty = MutableLiveData<MarsProperty>()
    val navigateToSelectedProperty: LiveData<MarsProperty>
        get() = _navigateToSelectedProperty

    /**
     * Call getMarsRealEstateProperties() on init so we can display status immediately.
     */
    init {
        getMarsRealEstateProperties(MarsApiFilter.SHOW_SEMUA)
    }

    /**
     * Gets Mars real estate property information from the Mars API Retrofit service and
     * updates the
     * [MarsProperty] [List] [LiveData]. The Retrofit service returns a coroutine
     * Deferred, which we
     * await to get the result of the transaction.
     */
    private fun getMarsRealEstateProperties(filter: MarsApiFilter) {

        viewModelScope.launch {
            _status.value = MarsApiStatus.LOADING
            try {
                _properties.value = MarsApi.retrofitService.getProperties(filter.value)
                _status.value = MarsApiStatus.DONE
            } catch (e: Exception) {
                _status.value = MarsApiStatus.ERROR
                _properties.value = ArrayList()
            }
        }
    }

    fun updateFilter(filter: MarsApiFilter) {
        getMarsRealEstateProperties(filter)
    }

    fun displayPropertyDetails(marsProperty: MarsProperty) {
        _navigateToSelectedProperty.value = marsProperty
    }

    fun displayPropertyDetailsComplete() {
        _navigateToSelectedProperty.value = null
    }
}

```

penjelasan:

perubahan yang saya buat disini adalah perubahan sederhana berupa SHOW_ALL menjadi SHOW_SEMUA. Kita membuat MarsApiFilter.SHOW_SEMUA sebagai argumen ke getMarsRealEstateProperties(), untuk menampilkan semua properti saat aplikasi pertama kali dimuat.

5. overflow_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/show_semua_menu"
        android:title="@string/show_semua" />
    <item
        android:id="@+id/show_cukup_menu"
        android:title="@string/show_cukup" />
    <item
        android:id="@+id/show_kurang_menu"
        android:title="@string/show_kurang" />
</menu>
```

Penjelasan:

Aplikasi ini memiliki menu tambahan yang menyediakan tiga opsi yang tersedia: menampilkan semua data, hanya menampilkan kategori cukup, dan menampilkan kategori kurang.

6. OverviewFragment.kt

```
package com.example.android.marsrealestate.overview

import android.os.Bundle
import android.view.*
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.example.android.marsrealestate.R
import com.example.android.marsrealestate.databinding.FragmentOverviewBinding
import com.example.android.marsrealestate.network.MarsApiFilter

/**
 * This fragment shows the the status of the Mars real-estate web services transaction.
 */
class OverviewFragment : Fragment() {

    /**
     * Lazily initialize our [OverviewViewModel].
     */
    private val viewModel: OverviewViewModel by lazy {
        ViewModelProvider(this).get(OverviewViewModel::class.java)
    }

    /**
     * Inflates the layout with Data Binding, sets its lifecycle owner to the
     OverviewFragment
     * to enable Data Binding to observe LiveData, and sets up the RecyclerView with an
     adapter.
     */
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        val binding = FragmentOverviewBinding.inflate(inflater)
        //val binding = GridViewItemBinding.inflate(inflater)

        // Allows Data Binding to Observe LiveData with the lifecycle of this Fragment
        binding.lifecycleOwner = this

        // Giving the binding access to the OverviewViewModel
        binding.viewModel = viewModel

        // Sets the adapter of the photosGrid RecyclerView
        binding.photosGrid.adapter = PhotoGridAdapter(PhotoGridAdapter.OnClickListener {
            viewModel.displayPropertyDetails(it)
        })

        viewModel.navigateToSelectedProperty.observe(this, Observer {
            if ( null != it ) {
                this.findNavController().navigate(
```

```

        OverviewFragmentDirections.actionShowDetail(it))
        viewModel.displayPropertyDetailsComplete()
    }
})

setHasOptionsMenu(true)
return binding.root
}

/**
 * Inflates the overflow menu that contains filtering options.
 */
override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
    inflater.inflate(R.menu.overflow_menu, menu)
    super.onCreateOptionsMenu(menu, inflater)
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    viewModel.updateFilter(
        when (item.itemId) {
            R.id.show_cukup_menu -> MarsApiFilter.SHOW_CUKUP
            R.id.show_kurang_menu -> MarsApiFilter.SHOW_KURANG
            else -> MarsApiFilter.SHOW_SEMUA
        }
    )
    return true
}
}

```

penjelasan:

di baris kode yang diganti, saya menggunakan MarsApiFilter.SHOW_SEMUA untuk nilai filter default, MarsApiFilter.SHOW_CUKUP untuk kategori data cukup umur, MarsApiFilter.SHOW_KURANG untuk kategori data kurang umur. Kembalikan true, karena kita telah menangani item menu.

7. PhotoGridAdapter.kt

```

package com.example.android.marsrealestate.overview

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.example.android.marsrealestate.databinding.GridViewItemBinding
import com.example.android.marsrealestate.network.MarsProperty

/**
 * This class implements a [RecyclerView] [ListAdapter] which uses Data Binding to
 * present [List]
 * data, including computing diffs between lists.
 */
class PhotoGridAdapter( private val onClickListener: OnClickListener ) :
    ListAdapter<MarsProperty,
        PhotoGridAdapter.MarsPropertyViewHolder>(DiffCallback) {

    /**
     * The MarsPropertyViewHolder constructor takes the binding variable from the
     associated
     * GridViewItem, which nicely gives it access to the full [MarsProperty] information.
     */
    class MarsPropertyViewHolder(private var binding: GridViewItemBinding):
        RecyclerView.ViewHolder(binding.root) {
        fun bind(marsProperty: MarsProperty) {
            binding.property = marsProperty
            // This is important, because it forces the data binding to execute
            immediately,
            // which allows the RecyclerView to make the correct view size measurements
            binding.executePendingBindings()
        }
    }
}

```

```

/**
 * Allows the RecyclerView to determine which items have changed when the [List] of
[MarsProperty]
 * has been updated.
 */
companion object DiffCallback : DiffUtil.ItemCallback<MarsProperty>() {
    override fun areItemsTheSame(oldItem: MarsProperty, newItem: MarsProperty):
Boolean {
        return oldItem === newItem
    }

    override fun areContentsTheSame(oldItem: MarsProperty, newItem: MarsProperty):
Boolean {
        return oldItem.nama == newItem.nama
    }
}

/**
 * Create new [RecyclerView] item views (invoked by the layout manager)
 */
override fun onCreateViewHolder(parent: ViewGroup,
                                viewType: Int): MarsPropertyViewHolder {
    return
MarsPropertyViewHolder(GridviewItemBinding.inflate(LayoutInflater.from(parent.context)))
}

/**
 * Replaces the contents of a view (invoked by the layout manager)
 */
override fun onBindViewHolder(holder: MarsPropertyViewHolder, position: Int) {
    val marsProperty = getItem(position)
    holder.itemView.setOnClickListener {
        onClickListener.onClick(marsProperty)
    }
    holder.bind(marsProperty)
}

class OnClickListener(val clickListener: (marsProperty:MarsProperty) -> Unit) {
    fun onClick(marsProperty:MarsProperty) = clickListener(marsProperty)
}
}

```

penjelasan:

hanya ada satu baris yang diganti yaitu return oldItem.nama == newItem.nama

8. strings.xml

```

<resources>
    <string name="app_name">Responsi</string>
    <string name="show_semua">Semua</string>
    <string name="show_cukup">Cukup</string>
    <string name="show_kurang">Kurang</string>
    <string name="umur_cukup">Cukup</string>
    <string name="umur_kurang">Kurang</string>
    <string name="display_type">%s Umur</string>
    <string name="display_username">Username: %s </string>
    <string name="display_nama">Nama: %s </string>
    <string name="display_umur">Umur: %d </string>
</resources>

```

Penjelasan:

Kode ini merupakan kumpulan kode untuk resource string yang saya gunakan untuk membantu membuat string untuk tampilan detail. Semua kode string dari aplikasi MarsRealEstateGrid sebelumnya sudah saya ganti. Nama aplikasinya saya ganti menjadi "Responsi". Opsi pilihan saya ganti dengan "show_semua", "show_cukup", "show_kurang". Tulisan textview utama pada halaman detail saya ganti dengan "display_type" yang nanti value nya diinisialisasikan dengan "umur_cukup" atau "umur_kurang". Terakhir,

di textview bagian sub detail saya gunakan “display_username”, “display_nama”, “display_umur” untuk menampilkan username, nama, dan umur sesuai dengan data yang ada di file JSON.

9. DetailViewModel.kt

```
package com.example.android.marsrealestate.detail

import android.app.Application
import androidx.lifecycle.*
import com.example.android.marsrealestate.R
import com.example.android.marsrealestate.network.MarsProperty

/**
 * The [ViewModel] that is associated with the [DetailFragment].
 */
class DetailViewModel(@Suppress("UNUSED_PARAMETER") marsProperty: MarsProperty,
                      app: Application) : AndroidViewModel(app) {
    private val _selectedProperty = MutableLiveData<MarsProperty>()
    val selectedProperty: LiveData<MarsProperty>
        get() = _selectedProperty

    init {
        _selectedProperty.value = marsProperty
    }

    val displayUsername = Transformations.map(selectedProperty) {
        app.applicationContext.getString(
            when (it.isCukup) {
                true -> R.string.display_username
                false -> R.string.display_username
            }, it.username)
    }

    val displayName = Transformations.map(selectedProperty) {
        app.applicationContext.getString(
            when (it.isCukup) {
                true -> R.string.display_nama
                false -> R.string.display_nama
            }, it.nama)
    }

    val displayUmur = Transformations.map(selectedProperty) {
        app.applicationContext.getString(
            when (it.isCukup) {
                true -> R.string.display_umur
                false -> R.string.display_umur
            }, it.umur)
    }

    val displayStatusUmur = Transformations.map(selectedProperty) {
        app.applicationContext.getString(R.string.display_type,
            app.applicationContext.getString(
                when (it.isCukup) {
                    true -> R.string.umur_cukup
                    false -> R.string.umur_kurang
                }
            ))
    }
}
```

penjelasan:

di sini saya mengganti displayPropertyPrice (menampilkan harga) dengan displayUsername (menampilkan username), val displayName (menampilkan nama), dan val displayUmur (menampilkan umur). Kemudian saya mengganti DisplayPropertyType dengan displayStatusUmur yang nantinya akan menampilkan status umur apakah “Cukup Umur” atau “Kurang Umur”. Transformasi yang ditulis di kode tersebut berfungsi untuk menguji properti yang sudah dituliskan di setiap method tersebut. Kedua properti (cukup dan kurang) akan diproses transformasi untuk memilih string yang sesuai dari resource dengan Kotlin switch when {}. Kedua kategori ini membutuhkan tipe data dari resource di bagian akhir, jadi saya menuliskan it.username, it.nama,

dan it.umur. Untuk displayStatusUmur, kita menggunakan display_type sebagai string dasar awal yang menampilkan "Umur" dan displayStatusUmur akan menentukan apakah data bernilai umur_cukup atau umur_kurang. Jika umur_cukup maka akan dipadukan dengan display_type dan muncul text "Cukup Umur", jika umur_kurang maka "Kurang Umur".

10. fragmen_detail.xml

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="viewModel"
            type="com.example.android.marsrealestate.detail.DetailViewModel" />
    </data>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".DetailFragment">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp">

            <ImageView
                android:id="@+id/main_photo_image"
                android:layout_width="0dp"
                android:layout_height="266dp"
                android:scaleType="centerCrop"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                tools:src="@tools:sample/backgrounds/scenic"
                app:imageUrl="@{viewModel.selectedProperty.imgSrcUrl}" />

            <TextView
                android:id="@+id/status_umur"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:textColor="#de000000"
                android:textSize="39sp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/main_photo_image"
                android:text="@{viewModel.displayStatusUmur}"
                tools:text="Cukup Umur" />

            <TextView
                android:id="@+id/username"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:textColor="#de000000"
                android:textSize="20sp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/status_umur"
                android:text="@{viewModel.displayUsername}"
                tools:text="user1" />

            <TextView
                android:id="@+id/nama"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:textColor="#de000000"
```

```

        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/username"
        android:text="@{viewModel.displayNama}"
        tools:text="NamaUser1" />

        <TextView
            android:id="@+id/umur"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textColor="#de000000"
            android:textSize="20sp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/nama"
            android:text="@{viewModel.displayUmur}"
            tools:text="$21" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
</layout>

```

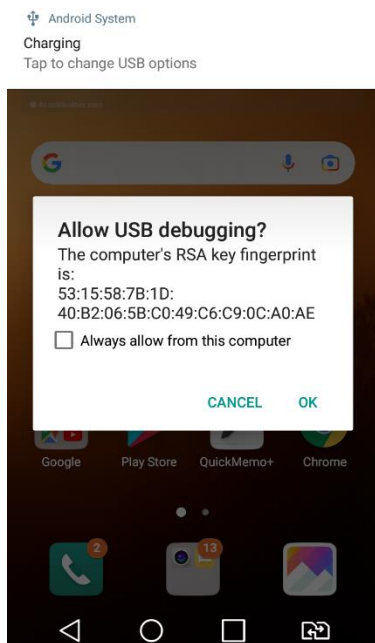
Penjelasan:

Pertama, saya membuat textview status_umur yang menampilkan data yang didapatkan dari method displayStatusUmur. Kemudian saya menampilkan beberapa textview yang menampilkan data-data seperti username, nama, dan umur dengan mengambil data dari method displayUsername, displayName, displayUmur.

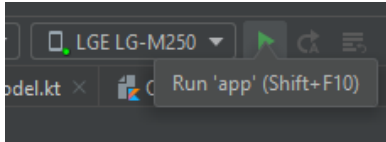
Bagian 2: Tampilan Aplikasi Ketika Dijalankan

PERHATIAN: di bagian ini, saya menjalankan aplikasi menggunakan perangkat fisik dengan smartphone saya (LG K10 2017) karena saya mengalami beberapa masalah koneksi internet pada emulator android studio. Saya melakukan proses screenshot dengan screenshot aplikasi di perangkat fisik lalu saya kirim hasil screenshot tersebut ke laptop lalu saya masukkan ke laporan ini. Terima kasih atas pengertiannya.

1. Pertama-tama, kita harus melakukan USB Debugging untuk menghubungkan perangkat fisik android dengan android studio supaya bisa dijadikan emulator.



2. Pilih perangkat fisik LG K10 2017 (LGE LG-M250) lalu run aplikasi, dan tunggu sampai proses gradle dan instalasi selesai, nanti otomatis aplikasi akan terbuka sendiri di perangkat.



3. Halaman utama pada aplikasi menampilkan tampilan default (show_semua) yang menampilkan kategori Cukup dan Kurang



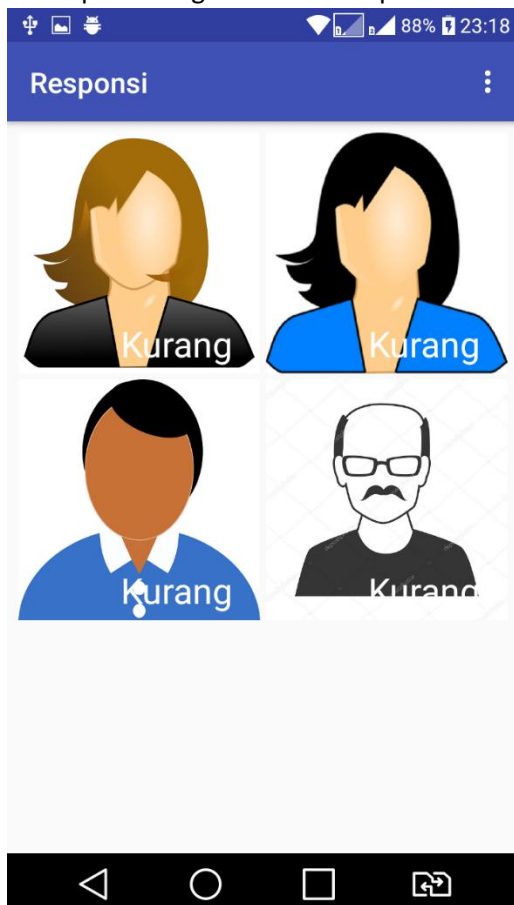
4. Klik 3 titik/dot di kanan atas aplikasi untuk menampilkan semua opsi kategori



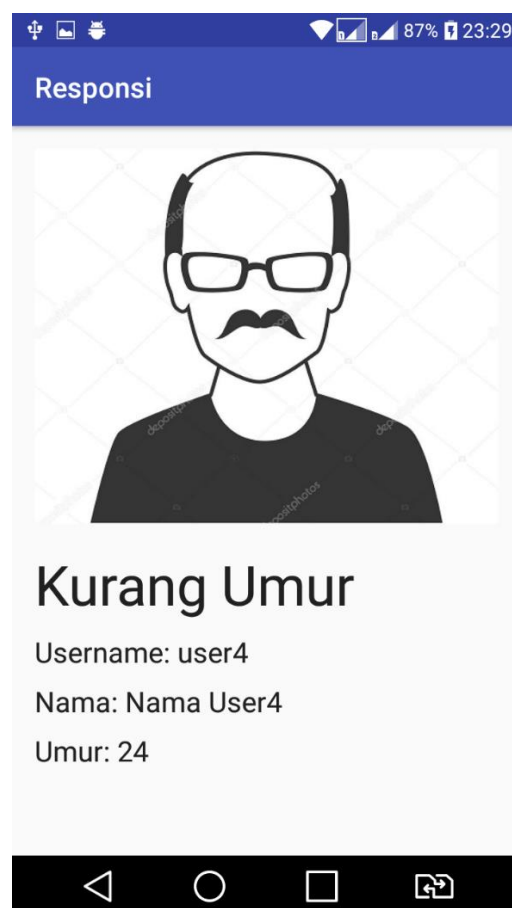
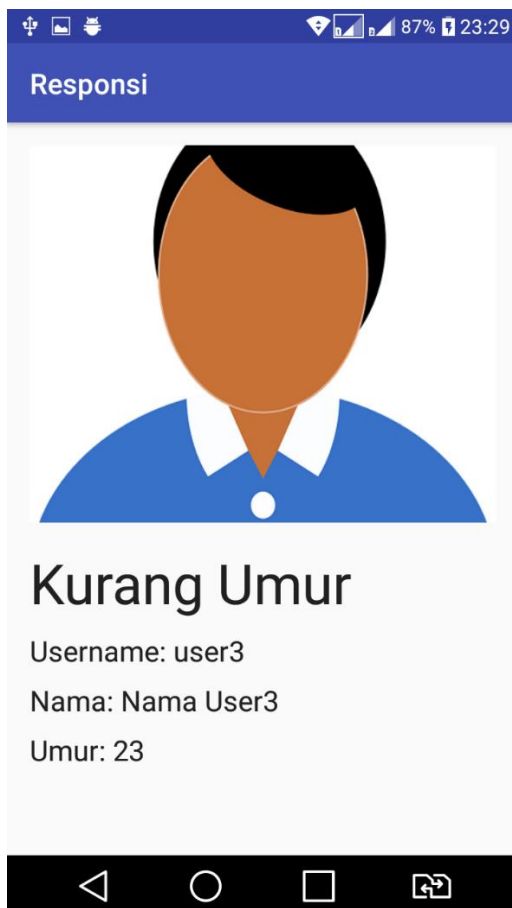
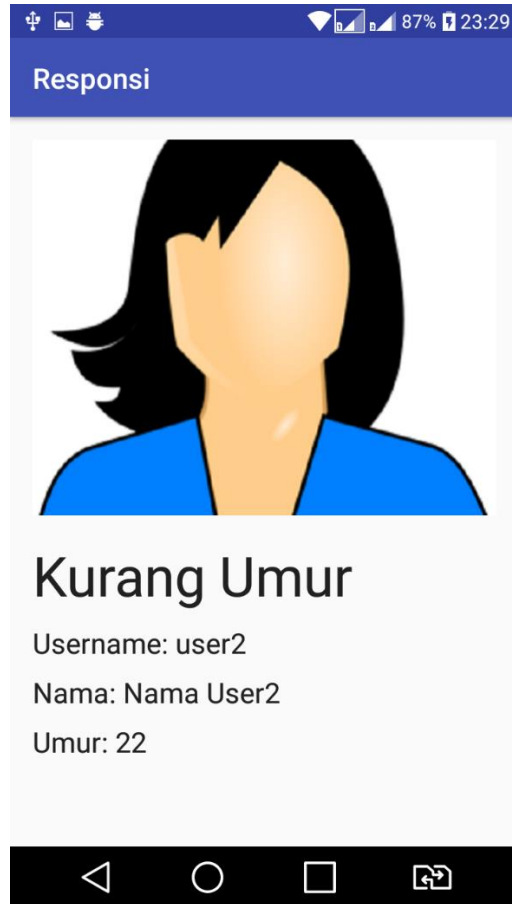
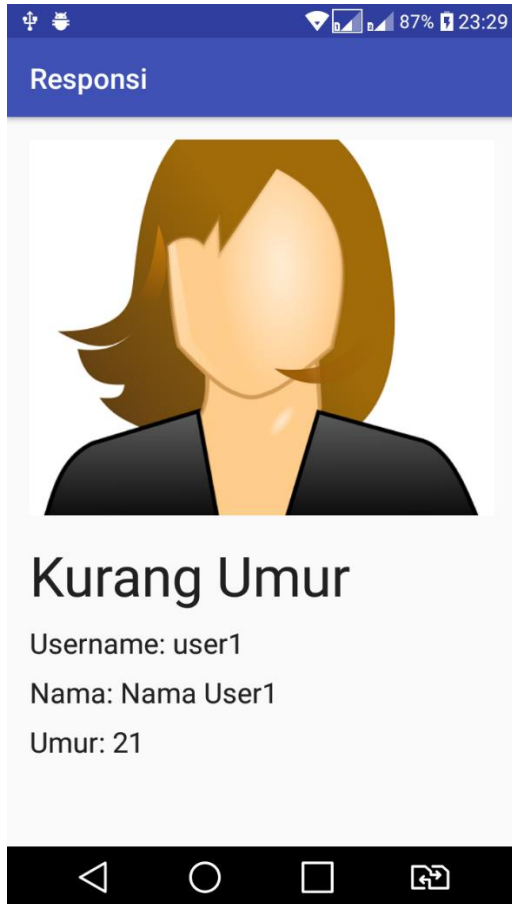
5. Pilih opsi Cukup untuk menampilkan semua data yang memiliki cukup umur (≥ 25 tahun)




6. Pilih opsi Kurang untuk menampilkan semua data yang memiliki kurang umur (< 25 tahun)



7. Sekarang, saya tampilkan semua detail data satu-persatu dari user1 sampai user 10 untuk membuktikan jika data yang ditampilkan di aplikasi sama dengan data yang ada di JSON.




Responsi



Cukup Umur

Username: user5
Nama: Nama User5
Umur: 25


Responsi



Cukup Umur

Username: user6
Nama: Nama User6
Umur: 26


Responsi



Cukup Umur

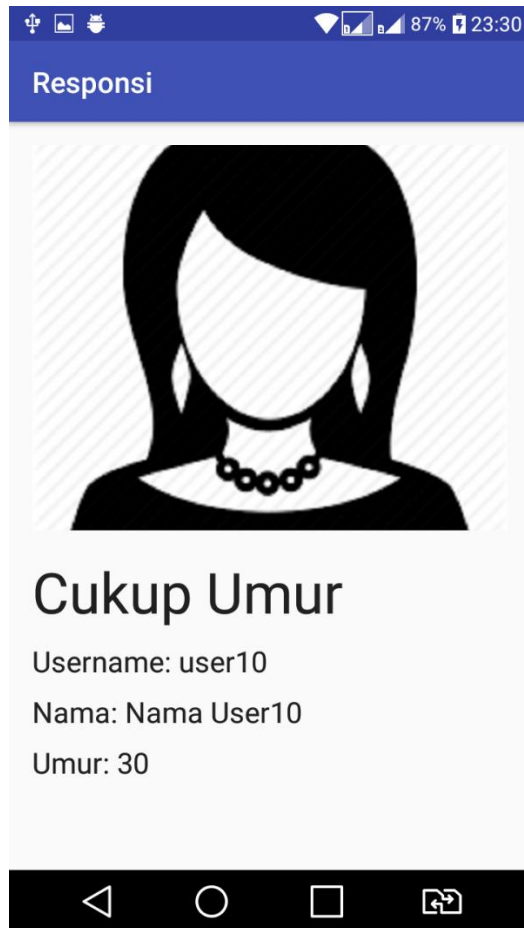
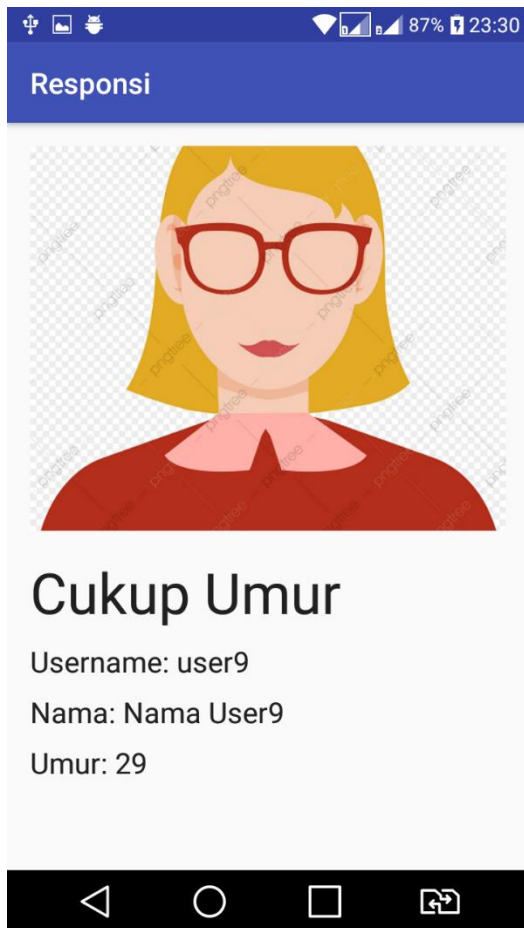
Username: user7
Nama: Nama User7
Umur: 27

Responsi



Cukup Umur

Username: user8
Nama: Nama User8
Umur: 28



Terima Kasih