

LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

Android Fundamental 4 - WEEK 10



Disusun oleh:

Nama : Alfath Roziq Widhayaka

Nim : L0122012

Kelas : A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

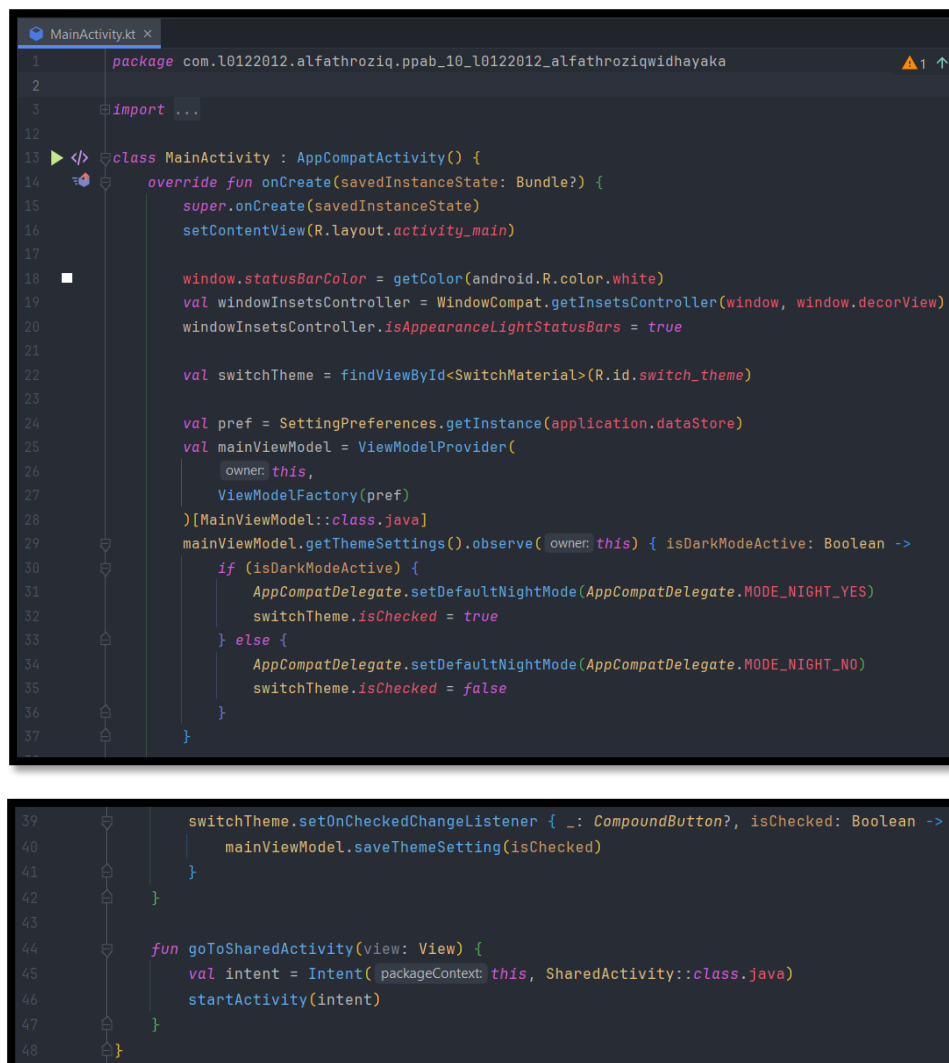
2024

1. Screenshot Source Code

Berikut adalah source code yang saya gunakan pada praktikum minggu ke-10 dalam mata kuliah Praktikum Pengembangan Aplikasi Bergerak. Dalam pengerjaannya, saya menerapkan **SharedPreferences** untuk mekanisme penyimpanan foto dan data diri yang saya miliki dan **DataStore** untuk menerapkan light/dark mode pada aplikasi. Berikut penjelasan masing-masing source code-nya.

A. Membuat Main Activity untuk membuat tampilan awal pada aplikasi dan diberi tombol light/dark mode.

▪ MainActivity.kt



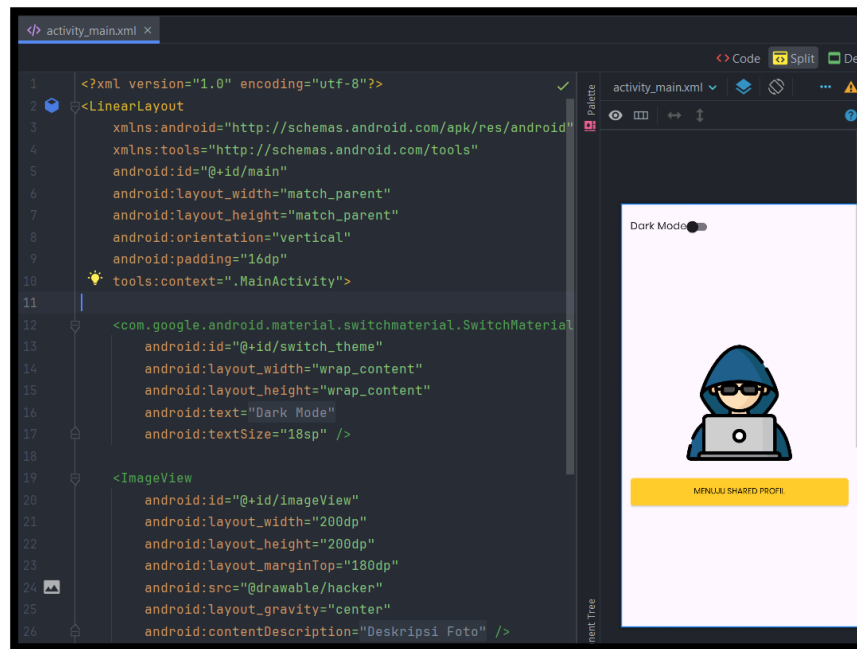
```
1 package com.l0122012.alfathroziq.ppab_10_l0122012_alfathroziqwidhayaka
2
3 import ...
4
5 class MainActivity : AppCompatActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContentView(R.layout.activity_main)
9
10        window.statusBarColor = getColor(android.R.color.white)
11        val windowInsetsController = WindowCompat.getInsetsController(window, window.decorView)
12        windowInsetsController.isAppearanceLightStatusBars = true
13
14        val switchTheme = findViewById<SwitchMaterial>(R.id.switch_theme)
15
16        val pref = SettingPreferences.getInstance(application.dataStore)
17        val mainViewModel = ViewModelProvider(
18            owner: this,
19            ViewModelFactory(pref)
20        )[MainViewModel::class.java]
21        mainViewModel.getThemeSettings().observe(owner: this) { isDarkModeActive: Boolean ->
22            if (isDarkModeActive) {
23                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
24                switchTheme.isChecked = true
25            } else {
26                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
27                switchTheme.isChecked = false
28            }
29        }
30
31        switchTheme.setOnCheckedChangeListener { _: CompoundButton?, isChecked: Boolean ->
32            mainViewModel.saveThemeSetting(isChecked)
33        }
34
35        fun goToSharedActivity(view: View) {
36            val intent = Intent(packageContext: this, SharedActivity::class.java)
37            startActivity(intent)
38        }
39    }
40
41    }
42
43
44
45
46
47
48 }
```

Gambar 1 MainActivity.kt

Source code di atas adalah MainActivity.kt untuk mengelola tema aplikasi **terang** atau **gelap** menggunakan **DataStore** untuk menyimpan preferensi pengguna. MainActivity mengatur tampilan status bar menjadi terang dan inisialisasi **SwitchMaterial** untuk mengubah tema. DataStore dikelola oleh **SettingPreferences**, sementara **MainViewModel** menangani logika pengaturan tema. Pengaturan terhadap **getThemeSettings** menentukan apakah mode gelap aktif dan menyesuaikan **AppCompatDelegate** serta status saklar. Ketika tombol switch ditekan,

saveThemeSetting di **MainViewModel** memperbarui preferensi tema di **DataStore**. Metode **goToSharedActivity** mengarahkan pengguna ke **SharedActivity**.

- **activity_main.xml**



Gambar 2 activity_main.xml

activity_main.xml di atas merupakan layout vertikal untuk **MainActivity** menggunakan **LinearLayout**. Layout ini mencakup tiga elemen utama yaitu **SwitchMaterial** untuk mengubah tema aplikasi, **ImageView** yang menampilkan gambar di tengah dengan margin atas 180dp, dan **AppCompatButton** yang menggunakan drawable khusus sebagai background serta memanggil metode **goToSharedActivity** saat diklik.

▪ SettingPreferences.kt

```
12 val Context.dataStore: DataStore<Preferences> by preferencesDataStore(name = "settings")
13
14 class SettingPreferences private constructor(private val datastore: DataStore<Preferences>) {
15
16     private val themeKey = booleanPreferencesKey(name = "theme_setting")
17
18     fun getThemeSetting(): Flow<Boolean> {
19         return datastore.data.map { preferences ->
20             preferences[themeKey] ?: false
21         }
22     }
23
24     suspend fun saveThemeSetting(isDarkModeActive: Boolean) {
25         datastore.edit { preferences ->
26             preferences[themeKey] = isDarkModeActive
27         }
28     }
29
30     companion object {
31         @Volatile
32         private var INSTANCE: SettingPreferences? = null
33
34         fun getInstance(dataStore: DataStore<Preferences>): SettingPreferences {
35             return INSTANCE ?: synchronized(lock = this) {
36                 val instance = SettingPreferences(dataStore)
37                 INSTANCE = instance
38                 instance
39             }
40         }
41     }
42 }
```

Gambar 3 SettingPreferences.kt

Kode di atas merupakan kelas **SettingPreferences** yang menggunakan **DataStore** untuk menyimpan preferensi tema aplikasi. Ini menyediakan metode untuk mengambil dan menyimpan preferensi tema dalam bentuk boolean. Metode **getInstance** memastikan hanya ada satu instance dari kelas ini, sementara metode **getThemeSetting** mengambil status tema saat ini dan **saveThemeSetting** menyimpan perubahan preferensi. Dengan demikian, kelas ini menyediakan antarmuka sederhana untuk mengelola preferensi tema aplikasi melalui **DataStore**.

▪ MainViewModel.kt

```
1 package com.l0122012.alfathroziq.ppab_l0122012_alfathroziqwidhayaka
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.ViewModel
5 import androidx.lifecycle.asLiveData
6 import androidx.lifecycle.viewModelScope
7 import kotlinx.coroutines.launch
8
9 class MainViewModel(private val pref: SettingPreferences) : ViewModel() {
10     fun getThemeSettings(): LiveData<Boolean> {
11         return pref.getThemeSetting().asLiveData()
12     }
13
14     fun saveThemeSetting(isDarkModeActive: Boolean) {
15         viewModelScope.launch { this: CoroutineScope
16             pref.saveThemeSetting(isDarkModeActive)
17         }
18     }
19 }
```

Gambar 4 MainViewModel.kt

Kode MainViewModel.kt di atas adalah kelas **MainViewModel** yang menghubungkan antarmuka pengguna dengan logika pengaturan tema menggunakan **DataStore**. Metode **getThemeSettings** mengambil status tema sebagai **LiveData**, sedangkan **saveThemeSetting** memungkinkan UI untuk memperbarui preferensi tema secara asinkron melalui **SettingPreferences**. Dengan demikian, **MainViewModel** menyederhanakan interaksi antara UI dan penyimpanan preferensi tema ketika aplikasi ditutup lalu dibuka lagi maka tema akan tetap disimpan.

- **ViewModelFactory.kt**

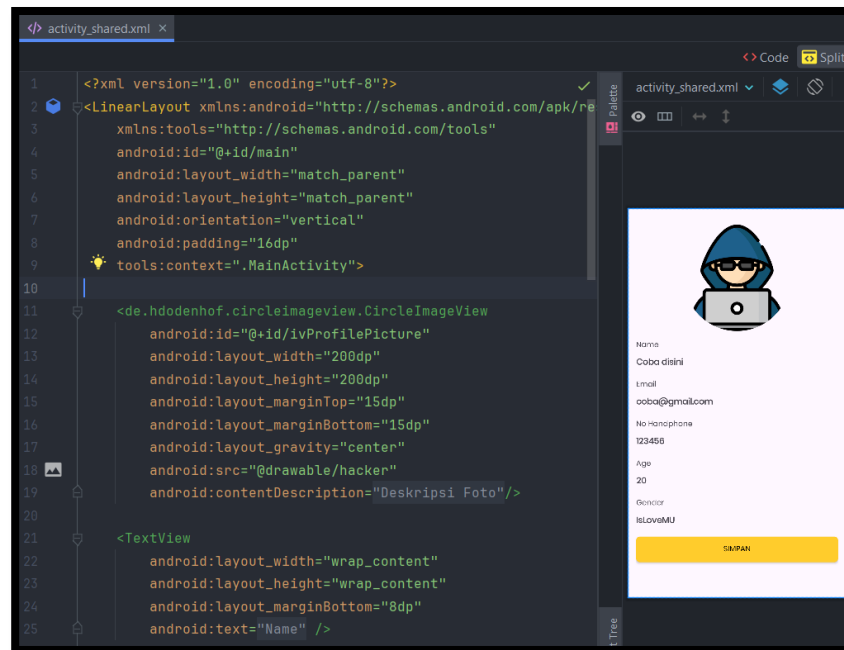
```
1 package com.l0122012.alfathroziq.ppab_10_l0122012_alfathroziqwidhayaka
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5
6 class ViewModelFactory(private val pref: SettingPreferences) : ViewModelProvider.NewInstanceFactory()
7
8 @Suppress("UNCHECKED_CAST")
9 override fun <T : ViewModel> create(modelClass: Class<T>): T {
10     if (modelClass.isAssignableFrom(MainViewModel::class.java)) {
11         return MainViewModel(pref) as T
12     }
13     throw IllegalArgumentException("Unknown ViewModel class: " + modelClass.name)
14 }
15 }
```

Gambar 5 ViewModelFactory.kt

Kode di atas merupakan kelas **ViewModelFactory** yang bertanggung jawab untuk membuat instance dari **MainViewModel** dengan menyediakan dependensi yang diperlukan. Metode **create** mengimplementasikan logika untuk membuat instance **MainViewModel** dengan memeriksa tipe model yang diminta. Jika tipe model yang diminta adalah **MainViewModel**, maka instance baru dari **MainViewModel** akan dibuat dengan menyediakan **SettingPreferences** yang diperlukan. Jika tipe model yang diminta tidak dikenali, akan dilemparkan **IllegalArgumentException**. Dengan demikian, **ViewModelFactory** memastikan bahwa dependensi diberikan dengan benar saat membuat instance **MainViewModel**.

B. Membuat SharedActivity dan FormUserPreferenceActivity untuk membuat tampilan menggunakan mekanisme SharedPreferences dengan formulir.

▪ activity_shared.xml



Gambar 6 activity_shared.xml

Kode XML diatas merupakan antarmuka pada **SharedActivity.kt**. Kode ini menggunakan LinearLayout sebagai wadah utama dengan beberapa elemen UI seperti ImageView untuk gambar profil, TextView untuk menampilkan informasi pengguna, dan sebuah tombol untuk menyimpan data. **SharedPreferences** disini digunakan untuk menyimpan data pengguna saat tombol "Save" ditekan. Ini memungkinkan pengguna untuk melanjutkan sesi mereka dengan data yang disimpan sebelumnya setelah aplikasi ditutup dan dibuka kembali.

▪ UserModel.kt

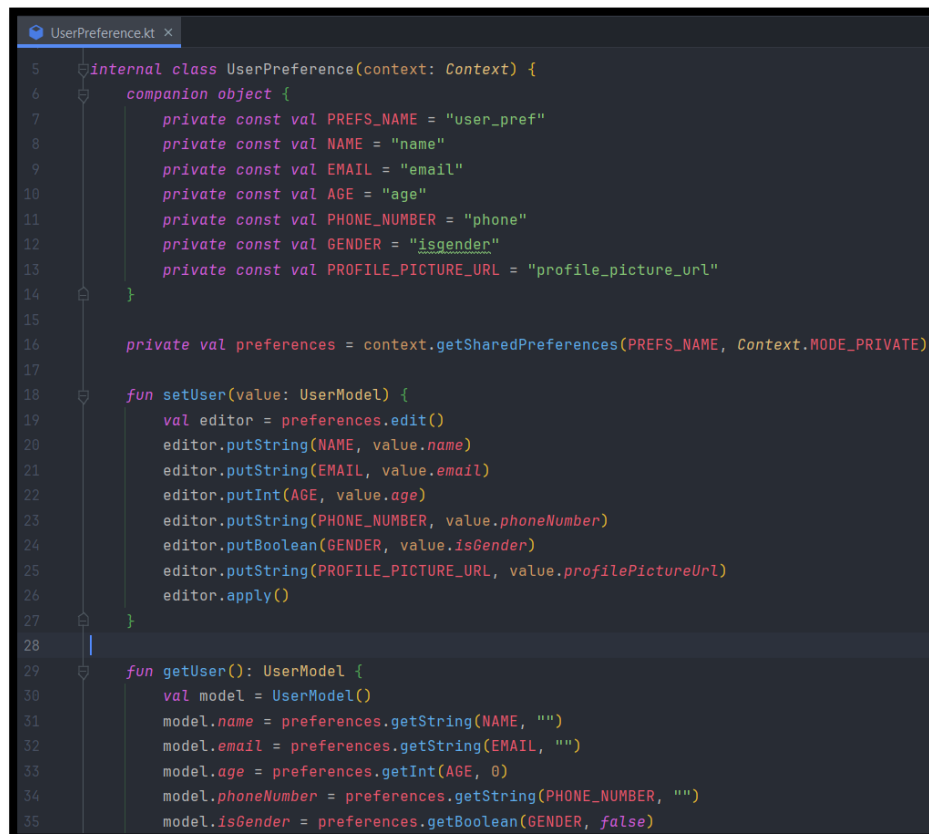


Gambar 7 UserModel.kt

Kode di atas adalah definisi kelas data Kotlin **UserModel** yang menyimpan informasi pengguna seperti nama, email, usia, nomor telepon, gender, dan URL gambar profil.

Dengan mengimplementasikan **Parcelable**, kelas ini dapat digunakan untuk mentransfer objek antar komponen aplikasi. Meskipun tidak langsung terkait dengan **SharedPreferences**, **UserModel** dapat bekerja dengan **SharedPreferences** dengan mengonversinya menjadi string JSON untuk disimpan dan dipulihkan dari **SharedPreferences**, memungkinkan pengelolaan data pengguna yang efisien dan pemeliharaan keadaan antara sesi aplikasi.

- **UserPreference.kt**



```
5 internal class UserPreference(context: Context) {
6     companion object {
7         private const val PREFS_NAME = "user_pref"
8         private const val NAME = "name"
9         private const val EMAIL = "email"
10        private const val AGE = "age"
11        private const val PHONE_NUMBER = "phone"
12        private const val GENDER = "isgender"
13        private const val PROFILE_PICTURE_URL = "profile_picture_url"
14    }
15
16    private val preferences = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
17
18    fun setUser(value: UserModel) {
19        val editor = preferences.edit()
20        editor.putString(NAME, value.name)
21        editor.putString(EMAIL, value.email)
22        editor.putInt(AGE, value.age)
23        editor.putString(PHONE_NUMBER, value.phoneNumber)
24        editor.putBoolean(GENDER, value.isGender)
25        editor.putString(PROFILE_PICTURE_URL, value.profilePictureUrl)
26        editor.apply()
27    }
28
29    fun getUser(): UserModel {
30        val model = UserModel()
31        model.name = preferences.getString(NAME, "")
32        model.email = preferences.getString(EMAIL, "")
33        model.age = preferences.getInt(AGE, 0)
34        model.phoneNumber = preferences.getString(PHONE_NUMBER, "")
35        model.isGender = preferences.getBoolean(GENDER, false)
```

Gambar 8 UserPreference.kt

Kode di atas adalah kelas **UserPreference** yang mengelola **SharedPreferences** untuk data pengguna. Metode **setUser()** digunakan untuk menyimpan data pengguna ke **SharedPreferences**, sedangkan metode **getUser()** digunakan untuk mengambilnya kembali. Properti yang disimpan dan diambil termasuk nama, email, usia, nomor telepon, gender, dan URL gambar profil. Dengan demikian, kelas ini menyediakan mekanisme yang efisien untuk menyimpan dan mengambil data pengguna dalam aplikasi menggunakan **SharedPreferences**.

▪ SharedActivity.kt

```
SharedActivity.kt x
13 @Suppress(...)
14 class SharedActivity : AppCompatActivity(), View.OnClickListener {
15
16     private lateinit var mUserPreference: UserPreference
17     private lateinit var binding: ActivitySharedBinding
18
19     private var isPreferenceEmpty = false
20     private lateinit var userModel: UserModel
21
22     private val resultLauncher = registerForActivityResult(
23         ActivityResultContracts.StartActivityForResult()
24     ) { result: ActivityResult ->
25         if (result.data != null && result.resultCode == FormUserPreferenceActivity.RESULT_CODE) {
26             val tempUserModel = result.data?.getParcelableExtra<UserModel>(FormUserPreferenceActivi
27             if (tempUserModel != null) {
28                 userModel = tempUserModel
29                 populateView(userModel)
30                 checkForm(userModel)
31             }
32         }
33     }
34
35     override fun onCreate(savedInstanceState: Bundle?) {
36         super.onCreate(savedInstanceState)
37         binding = ActivitySharedBinding.inflate(layoutInflater)
38         setContentView(binding.root)
39
40         window.statusBarColor = getColor(android.R.color.white)
41         val windowInsetsController = WindowCompat.getInsetsController(window, window.decorView)
42         windowInsetsController.isAppearanceLightStatusBars = true
```

```
SharedActivity.kt x
44 supportActionBar?.title = "My User Preference"
45
46 mUserPreference = UserPreference(context this)
47
48 showExistingPreference()
49
50 binding.btnSave.setOnClickListener(this)
51
52
53 private fun showExistingPreference() {
54     userModel = mUserPreference.getUser()
55     populateView(userModel)
56     checkForm(userModel)
57 }
58
59 private fun populateView(userModel: UserModel) {
60     binding.tvName.text = if (userModel.name.isNullOrEmpty()) "Tidak Ada" else userModel.name
61     binding.tvAge.text = userModel.age.toString()
62     binding.tvGender.text = if (userModel.isGender) "Male" else "Female"
63     binding.tvEmail.text = if (userModel.email.isNullOrEmpty()) "Tidak Ada" else userModel.emai
64     binding.tvPhone.text = if (userModel.phoneNumber.isNullOrEmpty()) "Tidak Ada" else userMode
65
66     val profilePictureUrl = userModel.profilePictureUrl
67     if (!profilePictureUrl.isNullOrEmpty()) {
68         Glide.with(activity this) RequestManager
69             .load(profilePictureUrl) RequestBuilder<Drawable!>
70             .into(binding.ivProfilePicture)
71     } else {
72         binding.ivProfilePicture.setImageResource(R.drawable.hacker)
73     }
74 }
```



```

76     private fun checkForm(userModel: UserModel) {
77         if (userModel.name.isNullOrEmpty()) {
78             binding.btnSave.text = "Simpan"
79             isPreferenceEmpty = true
80         } else {
81             binding.btnSave.text = "Ubah"
82             isPreferenceEmpty = false
83         }
84     }
85
86     override fun onClick(view: View) {
87         if (view.id == R.id.btnSave) {
88             val intent = Intent(packageContext, this@SharedActivity, FormUserPreferenceActivity::class)
89             if (isPreferenceEmpty) {
90                 intent.putExtra(FormUserPreferenceActivity.EXTRA_TYPE_FORM, FormUserPreferenceActiv
91             } else {
92                 intent.putExtra(FormUserPreferenceActivity.EXTRA_TYPE_FORM, FormUserPreferenceActiv
93             }
94             intent.putExtra(name: "USER", userModel)
95             resultLauncher.launch(intent)
96         }
97     }
98 }

```

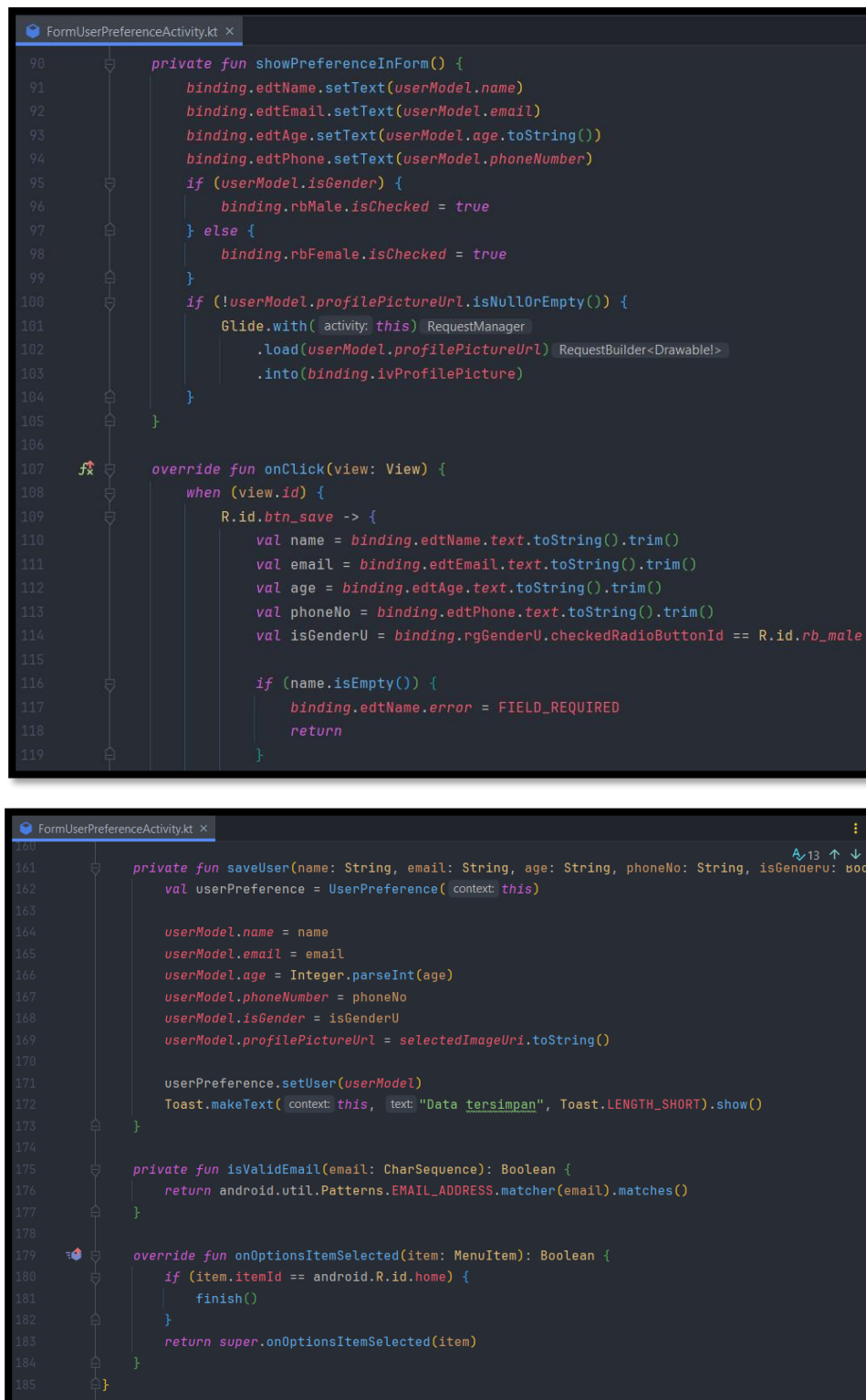
Gambar 9 SharedActivity.kt

Kode di atas adalah sebuah aktivitas **SharedActivity** yang bertanggung jawab untuk menampilkan data pengguna yang disimpan dalam **SharedPreferences**, memungkinkan pengguna untuk mengubah atau mengisi data tersebut, dan kemudian menampilkan kembali hasilnya. Ketika aktivitas dibuat, metode **onCreate()** dipanggil untuk menyiapkan tampilan dan memuat data pengguna dari **SharedPreferences** menggunakan kelas **UserPreference**. Selain itu, aktivitas ini juga mengimplementasikan fungsi untuk menangani hasil dari aktivitas tambahan untuk mengubah data pengguna. Data pengguna ditampilkan dalam tampilan, dan pengguna dapat mengubahnya dengan menekan tombol "Save", yang akan membuka aktivitas tambahan yaitu **FormUserPreferenceActivity**. Setelah pengguna selesai mengubah atau mengisi data, hasilnya akan diperbarui dalam **SharedPreferences** dan ditampilkan kembali dalam **SharedActivity**. Dengan cara ini, aktivitas ini memberikan antarmuka pengguna untuk mengelola data pengguna dengan cara mengubah atau mengisi data.

▪ FormUserPreferenceActivity.kt

```
FormUserPreferenceActivity.kt x
19 @Suppress(...)
20 class FormUserPreferenceActivity : AppCompatActivity(), View.OnClickListener {
21     |
22     private lateinit var binding: ActivityFormUserPreferenceBinding
23
24     companion object {
25         const val EXTRA_TYPE_FORM = "extra_type_form"
26         const val EXTRA_RESULT = "extra_result"
27         const val RESULT_CODE = 101
28
29         const val TYPE_ADD = 1
30         const val TYPE_EDIT = 2
31
32         private const val FIELD_REQUIRED = "Field tidak boleh kosong"
33         private const val FIELD_DIGIT_ONLY = "Hanya boleh terisi numerik"
34         private const val FIELD_IS_NOT_VALID = "Email tidak valid"
35     }
36
37     private lateinit var userModel: UserModel
38     private var selectedImageUri: Uri? = null
39
40     private val imagePickerLauncher = registerForActivityResult(
41         ActivityResultContracts.StartActivityForResult()
42     ) { result: ActivityResult ->
43         if (result.resultCode == Activity.RESULT_OK) {
44             val data = result.data
45             if (data != null && data.data != null) {
46                 selectedImageUri = data.data
47                 Glide.with(activity: this) RequestManager
48                     .load(selectedImageUri) RequestBuilder<Drawable>
```

```
FormUserPreferenceActivity.kt x
54 override fun onCreate(savedInstanceState: Bundle?) {
55     super.onCreate(savedInstanceState)
56     binding = ActivityFormUserPreferenceBinding.inflate(layoutInflater)
57     setContentView(binding.root)
58
59     window.statusBarColor = getColor(android.R.color.white)
60     val windowInsetsController = WindowCompat.getInsetsController(window, window.decorView)
61     windowInsetsController.isAppearanceLightStatusBars = true
62
63     binding.btnSave.setOnClickListener(this)
64     binding.btnChoosePicture.setOnClickListener(this)
65
66     userModel = intent.getParcelableExtra<UserModel>(name: "USER") as UserModel
67     val formType = intent.getIntExtra(EXTRA_TYPE_FORM, defaultValue: 0)
68
69     var actionBarTitle = ""
70     var btnTitle = ""
71
72     when (formType) {
73         TYPE_ADD -> {
74             actionBarTitle = "Tambah Baru"
75             btnTitle = "Simpan"
76         }
77         TYPE_EDIT -> {
78             actionBarTitle = "Ubah"
79             btnTitle = "Update"
80             showPreferenceInForm()
81         }
82     }
83 }
```



```

FormUserPreferenceActivity.kt
90     private fun showPreferenceInForm() {
91         binding.edtName.setText(userModel.name)
92         binding.edtEmail.setText(userModel.email)
93         binding.edtAge.setText(userModel.age.toString())
94         binding.edtPhone.setText(userModel.phoneNumber)
95         if (userModel.isGender) {
96             binding.rbMale.isChecked = true
97         } else {
98             binding.rbFemale.isChecked = true
99         }
100         if (!userModel.profilePictureUrl.isNullOrEmpty()) {
101             Glide.with( activity: this) RequestManager
102                 .load(userModel.profilePictureUrl) RequestBuilder<Drawable>
103                 .into(binding.ivProfilePicture)
104         }
105     }
106
107     override fun onClick(view: View) {
108         when (view.id) {
109             R.id.btn_save -> {
110                 val name = binding.edtName.text.toString().trim()
111                 val email = binding.edtEmail.text.toString().trim()
112                 val age = binding.edtAge.text.toString().trim()
113                 val phoneNo = binding.edtPhone.text.toString().trim()
114                 val isGenderU = binding.rgGenderU.checkedRadioButtonId == R.id.rb_male
115
116                 if (name.isEmpty()) {
117                     binding.edtName.error = FIELD_REQUIRED
118                     return
119                 }
120
121                 private fun saveUser(name: String, email: String, age: String, phoneNo: String, isGenderU: Boolean) {
122                     val userPreference = UserPreference( context: this)
123
124                     userModel.name = name
125                     userModel.email = email
126                     userModel.age = Integer.parseInt(age)
127                     userModel.phoneNumber = phoneNo
128                     userModel.isGender = isGenderU
129                     userModel.profilePictureUrl = selectedImageUri.toString()
130
131                     userPreference.setUser(userModel)
132                     Toast.makeText( context: this, text: "Data tersimpan", Toast.LENGTH_SHORT).show()
133                 }
134
135                 private fun isValidEmail(email: CharSequence): Boolean {
136                     return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
137                 }
138
139                 override fun onOptionsItemSelected(item: MenuItem): Boolean {
140                     if (item.itemId == android.R.id.home) {
141                         finish()
142                     }
143                     return super.onOptionsItemSelected(item)
144                 }
145             }
146         }
147     }

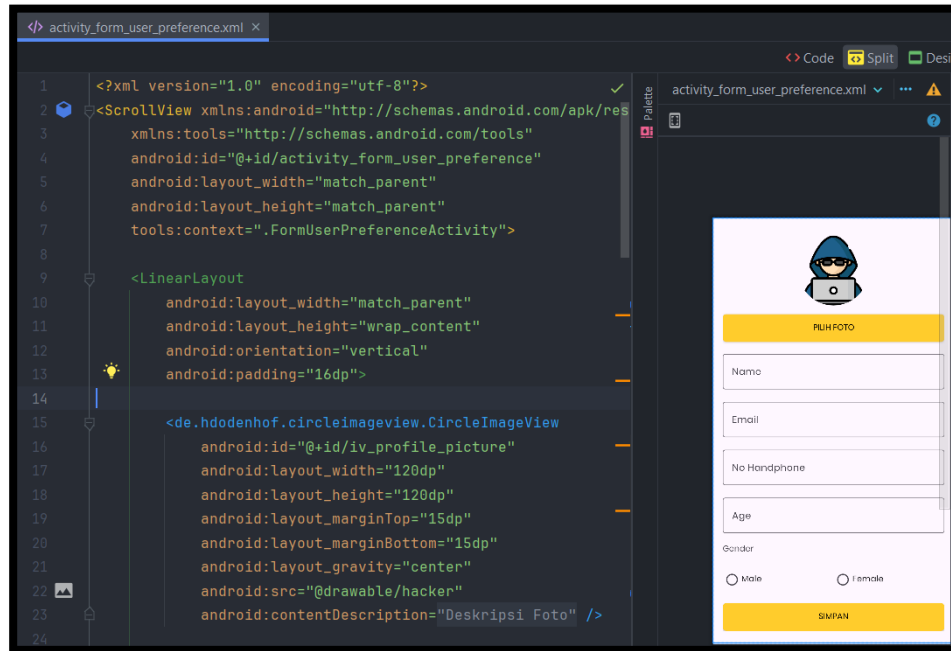
```

Gambar 10 FormUserPreferenceActivity.kt

Kode di atas adalah aktivitas **FormUserPreferenceActivity** yang bertanggung jawab untuk menampilkan formulir mengisi atau ubah data pengguna. Ketika aktivitas dibuat, formulir disesuaikan berdasarkan tipe yang diterima dari intent. Data pengguna dimuat jika aktivitas bertujuan untuk mengubah data yang ada. Pengguna dapat memilih gambar apapun menggunakan galeri, dan formulir akan memvalidasi input pengguna sebelum menyimpannya. Setelah disimpan, data pengguna diperbarui dalam

SharedPreferences menggunakan kelas **UserPreference**. Aktivitas ini berfungsi sebagai antarmuka pengguna untuk mengelola data pengguna, dengan **SharedPreferences** digunakan sebagai media penyimpanannya.

- **activity_form_user_preference.xml**



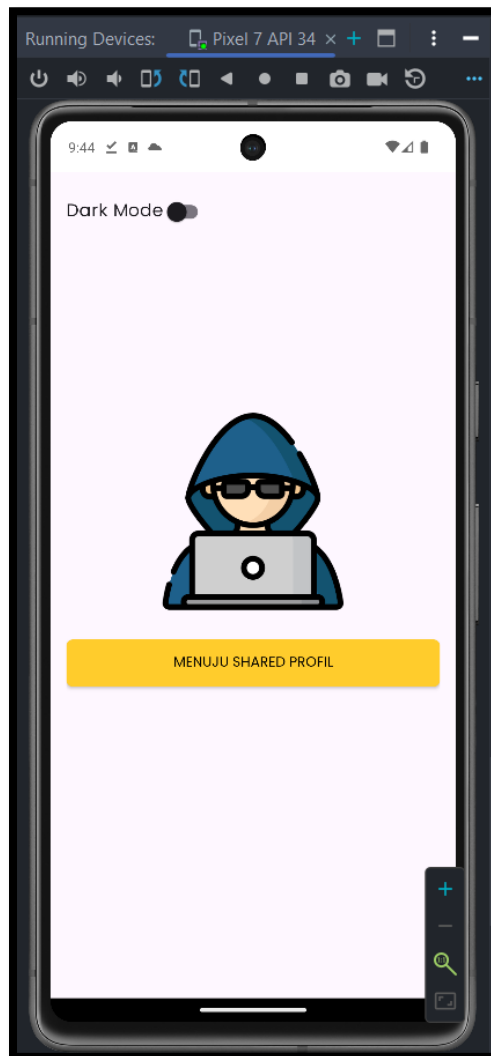
Gambar 11 activity_form_user_preference.xml

Kode XML di atas adalah tata letak untuk tampilan `FormUserPreferenceActivity.kt` dalam aplikasi Android. Dengan `ScrollView` untuk memungkinkan pengguna menggulir konten jika diperlukan, elemen-elemen seperti foto profil, input teks untuk nama, email, nomor telepon, dan usia, serta pilihan "male" atau "female" untuk pertanyaan gender, semuanya disusun dalam `LinearLayout`. Implementasi penyimpanan preferensi menggunakan `SharedPreferences` akan memungkinkan aplikasi menyimpan dan mengambil nilai-nilai ini secara persisten di perangkat pengguna, sehingga data dapat diakses kembali setiap kali aplikasi dibuka.

2. Screenshot Emulator

Berikut adalah hasil pada emulator ketika aplikasi dijalankan atau dieksekusi. Aplikasi ini menerapkan materi penyimpanan dengan mekanisme **SharedPreferences** untuk mekanisme penyimpanan foto dan data diri yang saya miliki dan **DataStore** untuk menerapkan light/dark mode pada aplikasi. Berikut merupakan hasil screenshot pada aplikasi.

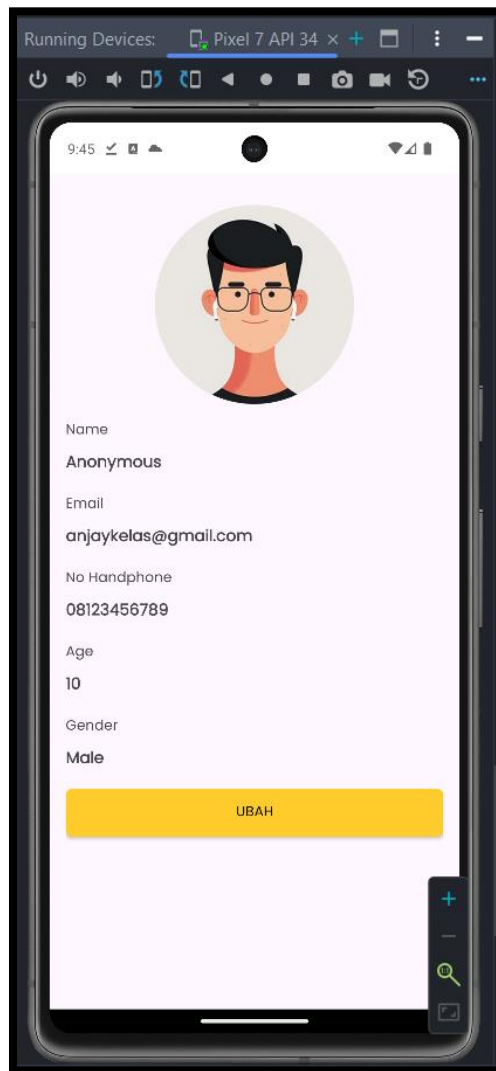
A. Tampilan Awal



Gambar 12 tampilan awal

Gambar diatas menampilkan hasil dari emulator setelah aplikasi dijalankan. Tampilan ini menampilkan tampilan awal aplikasi yang didalamnya terdapat tombol switch untuk mengubah tema light/dark mode dan terdapat tombol "Menuju Shared Profil". Pada tampilan ini menerapkan materi penyimpanan **DataStore** yang diterapkan pada switch tema light/dark mode. DataStore ini berguna untuk mengubah warna tema yang diterapkan dan akan terus menyimpan tema nya ketika user keluar dan masuk kembali aplikasi tersebut.

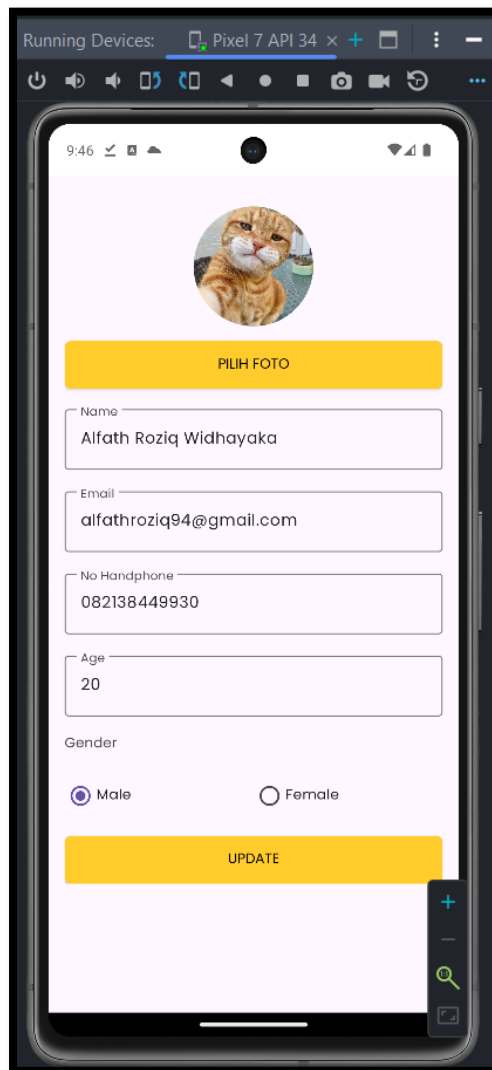
B. Tampilan Shared Activity (Tampilan sebelum data diisi atau diubah)



Gambar 13 tampilan shared activity sebelum diubah

Gambar diatas merupakan gambar emulator ketika user telah menekan tombol "Menuju Shared Profil". Pada tampilan ini (SharedActivity) merupakan tampilan untuk menampilkan data pengguna yang belum diubah/diisi oleh user.

C. Tampilan FormUserPreference (Pengisian Formulir untuk mengubah atau mengisi data)

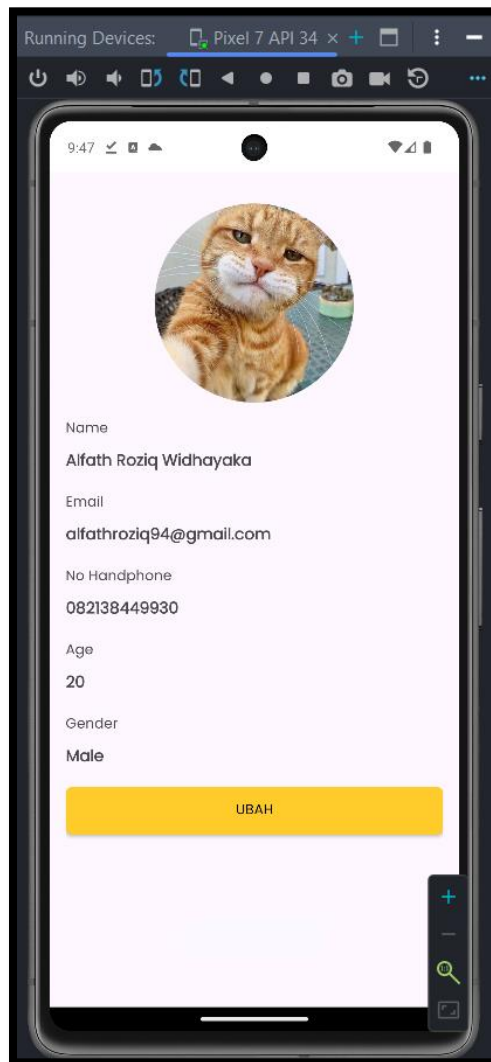


The screenshot shows a mobile application interface for updating user preferences. At the top, there is a status bar with the time 9:46 and various icons. Below the status bar is a header with a circular profile picture of a cat and a yellow button labeled "PILIH FOTO". The form contains several input fields: "Name" with the value "Alfath Roziq Widhayaka", "Email" with the value "alfathroziq94@gmail.com", "No Handphone" with the value "082138449930", and "Age" with the value "20". Below these fields is a "Gender" section with two radio buttons: "Male" (selected) and "Female". At the bottom of the form is a yellow button labeled "UPDATE". The entire form is displayed on a smartphone screen within an Android Studio environment, as indicated by the "Running Devices" header at the top of the image.

Gambar 14 tampilan mengisi atau mengubah data pada form

Gambar diatas merupakan gambar ketika user menekan tombol "Ubah" pada tampilan sebelumnya. Pada tampilan ini, user dapat mengubah atau mengisi data yang diinginkan user melalui form yang harus diisi. User juga dapat mengubah foto yang dipilih melalui galeri user. Jika sudah selesai tekan tombol "Update" untuk kembali ke tampilan sebelumnya. Materi yang diterapkan pada tampilan ini adalah mekanisme penyimpanan **SharedPreferences** dalam menjalankan formulir aplikasi yang dapat menyimpan data tersebut.

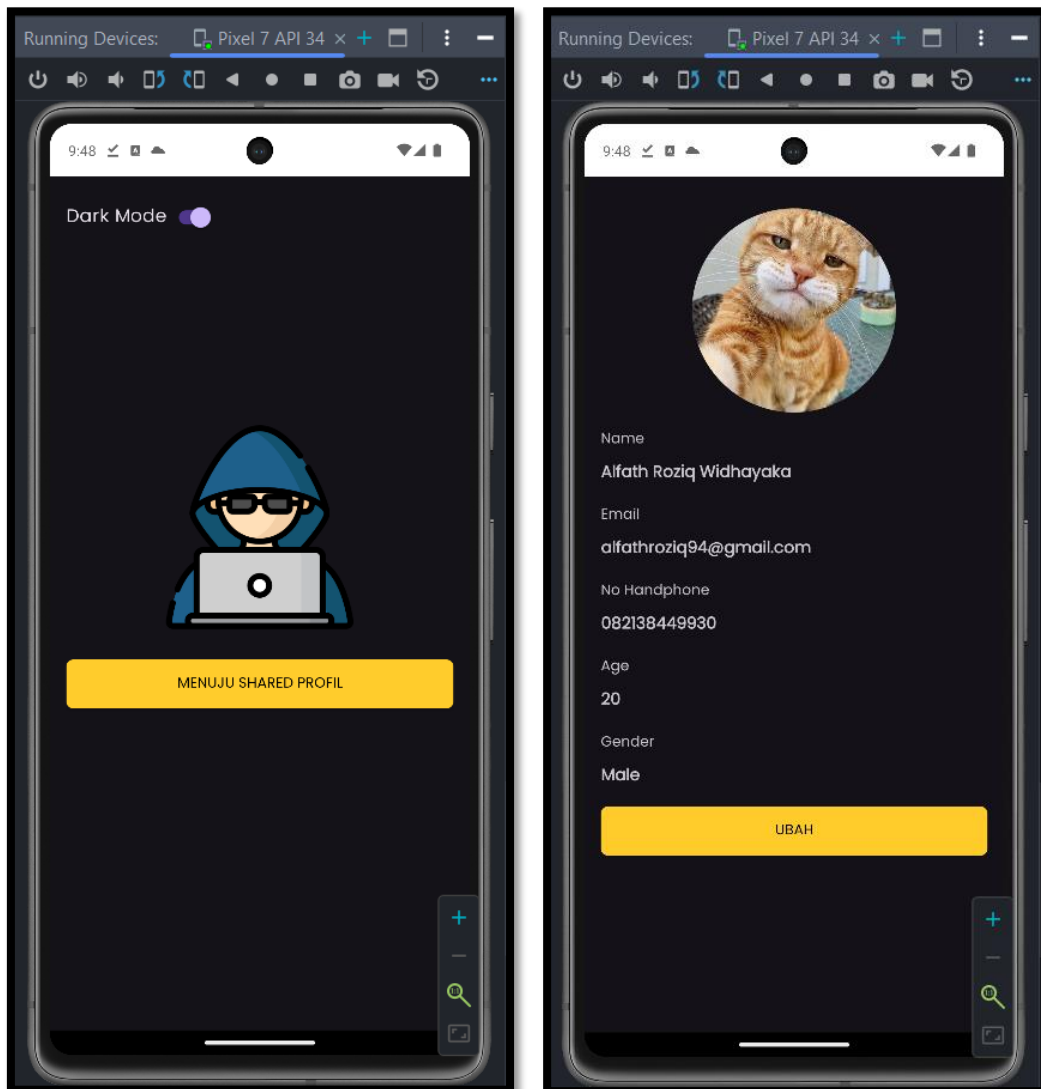
D. Tampilan Shared Activity (Tampilan setelah data diubah)



Gambar 15 tampilan shared activity ketika tampilan data sudah diubah

Gambar diatas merupakan tampilan emulator ketika data user telah diubah atau diisi oleh user sendiri. Disini user telah mengganti gambar dan mengganti data diri.

E. Tampilan setelah aplikasi diubah ke mode dark mode menggunakan switch



Gambar 15 tampilan aplikasi ketika diubah menjadi dark mode menggunakan switch

Gambar diatas merupakan tampilan ketika user kembali ke tampilan awal aplikasi dan user ingin mengubah nya menjadi tema dark mode. Ketika aplikasi diubah menjadi dark mode melalui switch yang ada di tampilan awal. Maka keseluruhan aplikasi di dalamnya ini akan berubah bertema gelap semua. Disini Aplikasi menerapkan **DataStore** dan **SharedPreferences** agar aplikasi ini menyimpan data user tema gelap ini dan data-data diri didalamnya, yaitu ketika user keluar dari aplikasi dan kembali masuk aplikasinya maka aplikasi akan tetap bertema gelap dan menyimpan data-data sebelumnya.

3. Kesimpulan

Pada praktikum ini, aplikasi yang saya buat mengimplementasikan mekanisme penyimpanan data menggunakan **SharedPreferences** dan **DataStore** dalam aplikasi Android. Penggunaan **SharedPreferences** memungkinkan penyimpanan data pengguna seperti nama, email, usia, nomor telepon, gender, dan URL gambar profil. Mekanisme ini memungkinkan data pengguna untuk tetap tersimpan dan dapat diakses kembali setelah aplikasi ditutup dan dibuka kembali. Dengan menggunakan kelas **UserPreference** dan **UserModel**, saya bisa mengelola data pengguna secara efisien dan menampilkan informasi tersebut pada antarmuka aplikasi melalui **SharedActivity** dan **FormUserPreferenceActivity**.

Selain itu, saya juga menerapkan **DataStore** untuk mengelola preferensi tema aplikasi (light/dark mode). Dengan menggunakan kelas **SettingPreferences** dan **MainViewModel**, aplikasi dapat menyimpan preferensi tema yang dipilih pengguna, sehingga ketika aplikasi ditutup dan dibuka kembali, tema yang dipilih akan tetap tersimpan dan diterapkan. Implementasi ini menunjukkan penggunaan arsitektur **ViewModel** untuk memisahkan logika bisnis dari antarmuka pengguna, serta memastikan bahwa preferensi pengguna disimpan dengan baik. Praktikum ini menunjukkan pentingnya pengelolaan data dan preferensi pengguna untuk meningkatkan pengalaman pengguna dalam aplikasi Android.