BAB 4
HASIL PENELITIAN

4.1 PERENCANAAN (*PLANNING*)

Perencanaan sistem dilakukan dengan melakukan pengumpulan data untuk menentukan garis besar aplikasi dan fitur-fitur di dalamnya yang akan dibuat. Proses penentuan tersebut dilakukan dengan 2 cara, yaitu analisis kompetitor dengan aplikasi-aplikasi serupa yang telah ada sebelumnya dan dengan melakukan wawancara kepada responden yang sesuai kriteria penelitian.

4.1.1 Analisis Kompetitor

Analisis kompetitor dilakukan dengan membandingkan fitur-fitur pada aplikasi analisis warna yang sudah populer yaitu Dressika dan Wardah *Personal Color* dengan aplikasi yang dikembangkan dalam penelitian ini. Dressika merupakan aplikasi *mobile personal color* dengan teknologi AI dan konsultan gaya pribadi yang menggunakan teori warna 12 musim. Wardah adalah merek produk kecantikan yang diproduksi oleh salah satu perusahaan kosmetik terbesar di Indonesia yaitu PT Paragon *Technology and Innovation*. Wardah memiliki situs web layanan konsultasi *personal color* yang bernama Wardah *Personal Color*.

Tabel 4.1 Hasil analisis kompetitor

No.	Fitur	Dressika	Wardah Personal Color	TrueMe
1.	*Personal color analysis*	✓	✓	✓
2.	Rekomendasi gaya berpakaian			✓
3.	Rekomendasi gaya pakaian harian			✓
4.	Rekomendasi *shade makeup*	✓	✓	
5.	Fitur *make up try-on*	✓		

Hasil analisis kompetitor menunjukkan bahwa *personal color analysis* telah diterapkan pada berbagai platform termasuk aplikasi *mobile* dan situs web. Penerapan AI juga telah dilakukan pada aplikasi Dressika, sehingga aplikasi ini

dapat menjadi referensi dalam pengembangan aplikasi pada penelitian ini. Kedua aplikasi kompetitor memiliki tampilan aplikasi yang mudah dipahami yang menjadi poin penting dalam pengembangan aplikasi pada penelitian ini. Dua aplikasi kompetitor memiliki fitur rekomendasi *shade makeup*, namun fitur tersebut tidak akan dikembangkan pada aplikasi yang dibuat karena fokus penelitian ini adalah untuk memberikan rekomendasi gaya berpakaian. Faktor utama pembeda pada aplikasi yang dikembangkan adalah dalam personalisasi berdasarkan warna kulit Indonesia yang belum dilakukan pada 2 aplikasi kompetitor. Aplikasi pada penelitian ini juga menyediakan rekomendasi gaya berpakaian harian untuk memberikan pengalaman yang lebih relevan dan kontekstual.

4.1.2 Wawancara

Wawancara dilakukan pada 10 responden dengan kriteria termasuk dalam generasi Z atau dengan rentang usia 17 hingga 27 tahun. Pertanyaan-pertanyaan yang diajukan berkaitan dengan preferensi penampilan & gaya berpakaian, pengalaman dengan aplikasi *fashion* atau *personal color*, kebutuhan & ekspektasi, serta harapan terhadap aplikasi yang dikembangkan. Daftar pertanyaan wawancara ditunjukkan pada tabel 4.2.

Tabel 4.2 Daftar pertanyaan wawancara

No.	Pertanyaan	Kategori
1.	Seberapa besar ketertarikan Anda terhadap dunia fashion atau gaya berpakaian?	Preferensi Penampilan & Gaya Berpakaian
2.	Apakah Anda merasa kesulitan dalam memilih warna pakaian yang cocok untuk kulit Anda?	Preferensi Penampilan & Gaya Berpakaian
3.	Bagaimana Anda biasanya memilih warna pakaian? (misalnya: berdasarkan selera, tren, saran teman, dll)	Preferensi Penampilan & Gaya Berpakaian
4.	Apakah Anda pernah mendengar tentang *personal color analysis* sebelumnya? Jika ya, dari mana Anda mengetahuinya? (contoh: TikTok, YouTube, dll)	Pengalaman Pengguna
5.	Apakah Anda pernah mencoba *personal color test*, baik secara *online* maupun *offline*?	Pengalaman Pengguna
6.	Bagaimana dengan aplikasi *fashion* atau *personal color test*? Apakah Anda pernah mengetahui atau menggunakannya?	Pengalaman Pengguna
7.	Jika pernah, aplikasi apa yang Anda gunakan dan bagaimana pengalaman Anda menggunakannya?	Pengalaman Pengguna

No.	Pertanyaan	Kategori
8.	Apa fitur yang menurut Anda paling berguna dari aplikasi tersebut?	Pengalaman Pengguna
9.	Apa kekurangan yang Anda rasakan dari aplikasi yang pernah Anda coba?	Pengalaman Pengguna
10.	Menurut Anda, fitur apa yang paling penting dalam aplikasi yang membantu memilih warna pakaian berdasarkan warna kulit?	Kebutuhan & Ekspektasi
11.	Seberapa penting antarmuka aplikasi yang menarik dan mudah digunakan bagi Anda?	Kebutuhan & Ekspektasi
12.	Apakah Anda tertarik menggunakan aplikasi yang bisa memberi saran gaya berpakaian harian berdasarkan *personal color* Anda?	Kebutuhan & Ekspektasi
13.	Jika tersedia, apakah Anda bersedia menggunakan aplikasi yang bisa menganalisis warna kulit dan merekomendasikan gaya pakaian?	Harapan Terhadap Aplikasi
14.	Apakah ada saran atau harapan pribadi Anda terhadap aplikasi semacam ini?	Harapan Terhadap Aplikasi

Wawancara dilakukan pada waktu yang berbeda dalam rentang 5 Mei sampai dengan 11 Mei 2025. Wawancara dilakukan kepada 10 responden dengan identitas responden diperlihatkan pada Tabel 4.3.

Tabel 4.3 Daftar Responden Wawancara

No.	Nama	Usia	Jenis Kelamin	Profesi
1.	Agil Pramana Putra	22	Laki-laki	Mahasiswa
2.	Dian Shesylia Putri	23	Perempuan	Mahasiswa
3.	Salsabila Hasna Fitria	21	Perempuan	Mahasiswa
4.	Fina Fatwasari	21	Perempuan	Mahasiswa
5.	Alda Kurnia Putri	22	Perempuan	Mahasiswa
6.	Sevira Setiyaningsih	21	Perempuan	Mahasiswa
7.	Muhammad Rizqi Akbar	22	Laki-laki	Mahasiswa
8.	Firna Rizka Sabila	21	Perempuan	Mahasiswa
9.	Dea Cahyaningsih	23	Perempuan	Pekerja
10.	Akhmad Romli	21	Laki-laki	Mahasiswa

Dari hasil wawancara yang telah dilakukan, diperoleh kesimpulan bahwa sebagian besar responden menginginkan tampilan aplikasi yang sederhana dan mudah dimengerti. Rekomendasi gaya pakaian yang sesuai dengan kepribadian

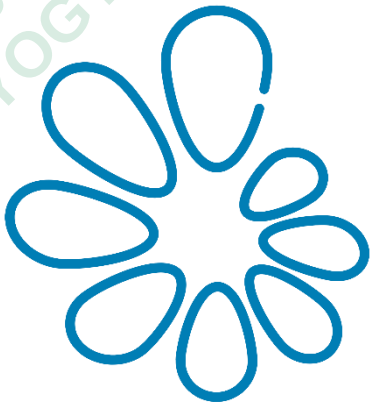
juga menjadi harapan pada beberapa responden. Dalam sistem analisis warna personal, responden menginginkan hasil yang akurat dan proses analisis yang cepat sehingga aplikasi dapat dipercaya sebagai salah satu alat referensi dalam berpakaian. Responden juga mengharapkan rekomendasi gaya pakaian harian.

4.1.3 Hasil Analisis Kebutuhan Sistem

Berdasarkan hasil analisis kompetitor dan wawancara yang telah dijabarkan pada sub bab sebelumnya, diperoleh hasil analisis kebutuhan sistem sebagai berikut:

1. Tampilan antarmuka aplikasi yang sederhana dan mudah dipahami.
2. Sistem rekomendasi gaya berpakaian yang terpersonalisasi.
3. Analisis warna personal yang cepat dan akurat.
4. Sistem rekomendasi gaya berpakaian harian.

Setelah mendapatkan hasil analisis kebutuhan sistem dan fitur-fitur yang akan dibuat, aplikasi yang dikembangkan penulis diberi nama “TrueMe”. Logo aplikasi TrueMe ditunjukkan pada Gambar 4.1. Logo tersebut mengambil referensi dari bentuk palet warna dan bunga.



Gambar 4.1 Logo TrueMe

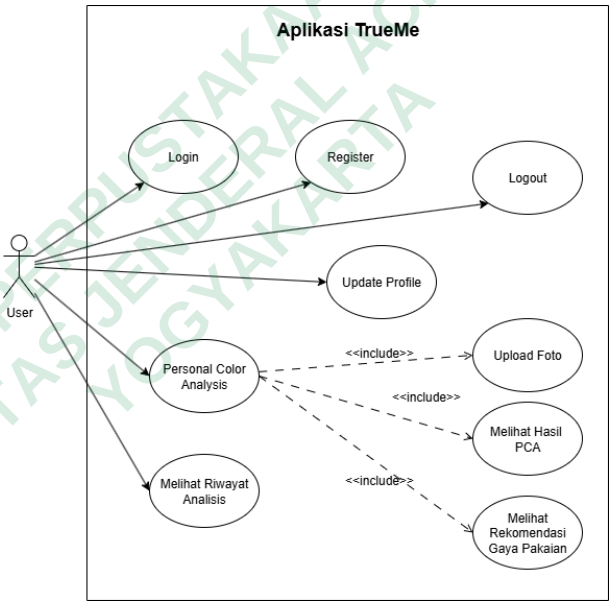
Tahap selanjutnya mulai dari perancangan hingga pengujian dilakukan dalam iterasi yang melibatkan responden awal untuk mendapatkan umpan balik. Pengumpulan umpan balik dilakukan pada tahap perancangan antarmuka sistem dan pada tahap pengujian *usability*.

4.2 PERANCANGAN (*DESIGN*)

Hasil dari analisis kebutuhan sistem pada tahap perencanaan sebelumnya dijadikan dasar dalam proses perancangan sistem. Tahap perancangan bertujuan untuk menghasilkan gambaran teknis terkait struktur dan alur kerja sistem sebelum nantinya diimplementasikan. Proses perancangan dalam penelitian ini meliputi pembuatan diagram UML, perancangan *database*, dan perancangan antarmuka pengguna (UI).

4.2.1 *Use Case Diagram*

Use case diagram dibuat untuk merancang tiap aktor yang terlibat dalam sistem dan relasinya dengan sistem. *Use case diagram* ditunjukkan pada Gambar 4.2.

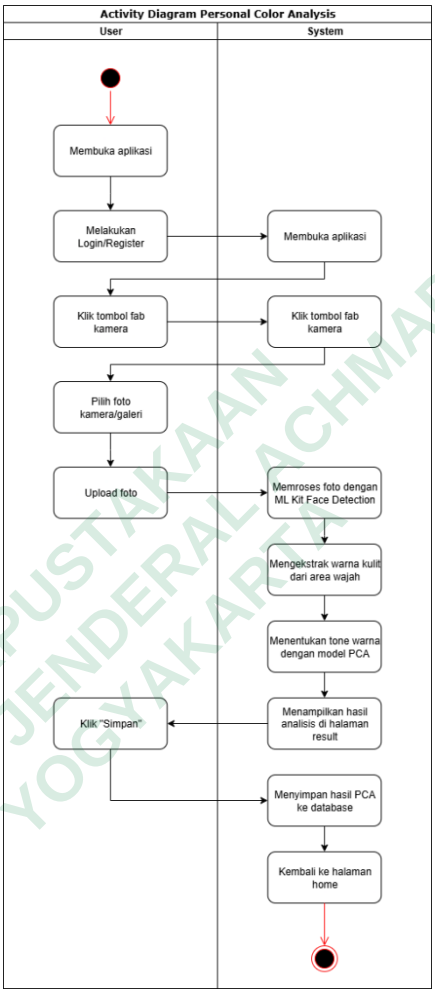


Gambar 4.2 *Use case diagram*

Dalam *use case diagram*, terdapat satu aktor yaitu *User*. *User* dapat melakukan berbagai aksi dalam aplikasi, di antaranya membuat akun (*register*), masuk ke dalam sistem (*login*), memperbarui profil (*update profile*), mengunggah foto, melakukan analisis warna kulit (*personal color analysis*), melihat hasil *personal color*, melihat riwayat hasil analisis warna, melihat hasil rekomendasi gaya berpakaian, dan keluar dari aplikasi (*logout*).

4.2.2 *Activity Diagram*

Activity diagram menggambarkan alur aktivitas pengguna saat menggunakan fitur utama aplikasi, *yaitu personal color analysis*. *Activity diagram* secara detail dapat dilihat pada Gambar 4.3.

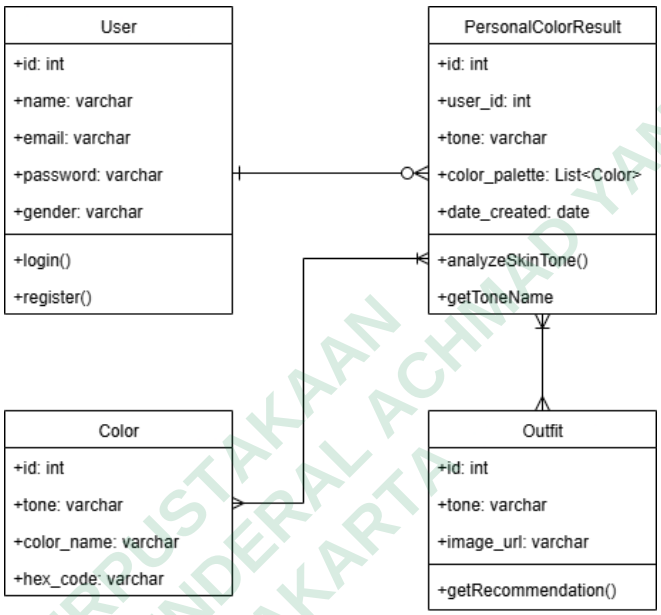


Gambar 4.3 *Activity diagram*

Alur dimulai dari pengguna membuka aplikasi, lalu melakukan *login* atau *register*. Selanjutnya pengguna menekan tombol dengan ikon kamera untuk mengunggah atau mengambil foto wajah. Setelah foto diunggah, sistem mendeteksi wajah menggunakan ML Kit. Setelah itu, sistem mengekstrak warna kulit dari area wajah dan mengklasifikasikan *tone* warna (*Spring*, *Summer*, *Autumn*, atau *Winter*) menggunakan model *machine learning*. Langkah terakhir, sistem menampilkan hasil *tone* dan rekomendasi gaya pakaian yang sesuai.

4.2.3 *Class Diagram*

Class Diagram menggambarkan struktur objek dalam sistem dan relasinya. Terdapat empat kelas utama, yaitu *User, PersonalColorResult, Color,* dan *OutfitRecommendation*. Gambaran lebih jelas terkait *class diagram* terdapat pada Gambar 4.4



Gambar 4.4 *Class diagram*

Kelas *User* menyimpan informasi pengguna seperti nama, *email*, *password*, dan *gender*. Setiap pengguna dapat memiliki nol atau lebih hasil analisis warna kulit yang disimpan di kelas *PersonalColorResult* (*one-to-many*). Setiap hasil analisis berelasi dengan beberapa warna dari kelas *Color* (*many-to-many*), yang menyimpan data warna berupa nama dan kode HEX. Selain itu, sistem juga memiliki koleksi rekomendasi gaya pakaian yang disimpan dalam kelas *Outfit*, yang dihubungkan berdasarkan *tone* warna. Relasi antara kelas *PersonalColorResult* dengan kelas *Outfit* adalah *one-to-many* yang berarti bahwa satu hasil personal color dapat merekomendasikan banyak gaya pakaian, namun satu gaya pakaian direkomendasikan oleh satu hasil *personal color* saja.

4.2.4 Struktur *Database*

Perancangan *database* dilakukan untuk mendukung proses penyimpanan dan pengelolaan data pada aplikasi. Struktur *database* dirancang berdasarkan hasil *class diagram* yang telah dijelaskan sebelumnya. Setiap entitas direpresentasikan dalam bentuk tabel yang saling berelasi.

1. Struktur Tabel *User*

Tabel *User* berfungsi sebagai penyimpanan data pengguna aplikasi. Tabel User memiliki 5 kolom. Struktur tabel *User* ditunjukkan pada Tabel 4.4.

Tabel 4.4 Struktur tabel *User*

Kolom	Tipe Data	Keterangan
id	Int(10)	*Primary key*, AI
name	Varchar(100)	
email	Varchar(100)	
password	Varchar(100)	
gender	Varchar(100)	

2. Struktur Tabel *PersonalColorResult*

Tabel *PersonalColorResult* digunakan untuk menyimpan data hasil analisis warna personal. Tabel ini memiliki relasi *one-to-many* dengan tabel *User*. Artinya, 1 pengguna dapat memiliki lebih dari satu hasil analisis *personal color*. Relasi ini direpresentasikan melalui kolom user_id pada tabel *PersonalColorResult* yang menjadi *foreign key* yang merujuk ke id pada tabel *User*. Struktur tabel *PersonalColorResult* terdiri dari 4 kolom, ditunjukkan pada Tabel 4.5.

Tabel 4.5 Struktur tabel *PersonalColorResult*

Kolom	Tipe Data	Keterangan
id	Int(10)	*Primary key*, AI
user_id	Int(10)	*Foreign key*
tone	Varchar(100)	
date_created	Datetime	

3. Struktur Tabel *Color*

Tabel *Color* digunakan untuk menyimpan informasi palet warna yang digunakan dalam hasil analisis *personal color*. Setiap warna memiliki nama, kode HEX, dan kategorisasi *tone* warna. Palet warna dan kategori *tone* berpedoman pada artikel dari "*The Concept Wardrobe*", sebuah *platform* konsultan gaya personal dan *digital retailer* yang berfokus pada penciptaan gaya busana pribadi. Struktur tabel *Color* terdapat pada Tabel 4.6.

Tabel 4.6 Struktur tabel *Color*

Kolom	Tipe Data	Keterangan
id	Int(10)	*Primary key*, AI
tone	Varchar(100)	
color_name	Varchar(100)	
hex_code	Varchar(10)	

4. Struktur Tabel *Outfit*

Tabel *Outfit* merupakan tabel untuk menyimpan data pakaian yang digunakan untuk sistem rekomendasi berdasarkan hasil *tone* warna pengguna. Gaya pakaian diperoleh dari *platform Pinterest* yang disesuaikan dengan palet warna yang telah diperoleh sebelumnya. Struktur tabel *Outfit* ditunjukkan pada Tabel 4.7.

Tabel 4.7 Struktur tabel *Outfit*

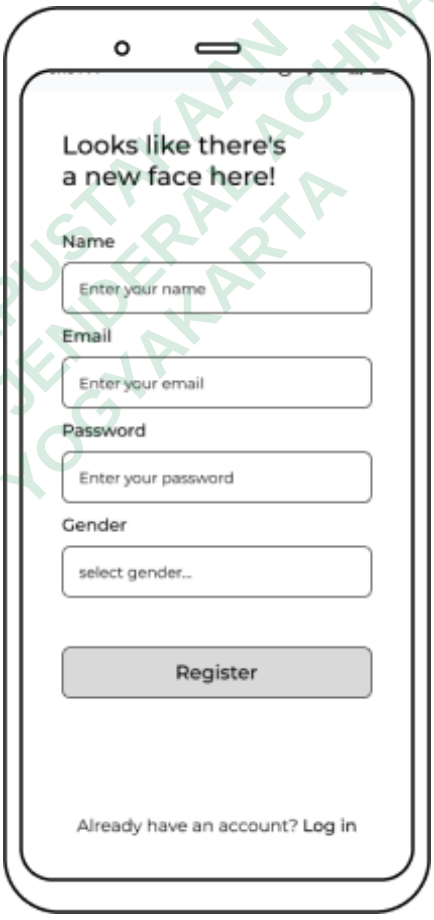
Kolom	Tipe Data	Keterangan
id	Int(10)	*Primary key*, AI
tone	Varchar(100)	
image_url	Varchar(100)	

4.2.5 Rancangan *Wireframe*

Wireframe merupakan gambaran visual sederhana dari tampilan antarmuka dalam bentuk sketsa. *Wireframe* berguna untuk menyusun tata letak elemen-elemen dalam halaman seperti teks, tombol, bidang input, dan lain sebagainya. Penjabaran rancangan *wireframe* untuk tiap-tiap halaman dalam aplikasi adalah sebagai berikut.

1. *Wireframe* Halaman *Register*

Halaman *register* adalah halaman yang digunakan oleh pengguna baru untuk membuat akun. *Wireframe* untuk halaman *register* ditunjukkan pada Gambar 4.5.



Gambar 4.5 *Wireframe* halaman *register*

Halaman ini berbentuk sebuah *form* dengan beberapa input yang harus diisi pengguna, antara lain nama, *email*, *password*, dan jenis kelamin.

2. *Wireframe* Halaman *Login*

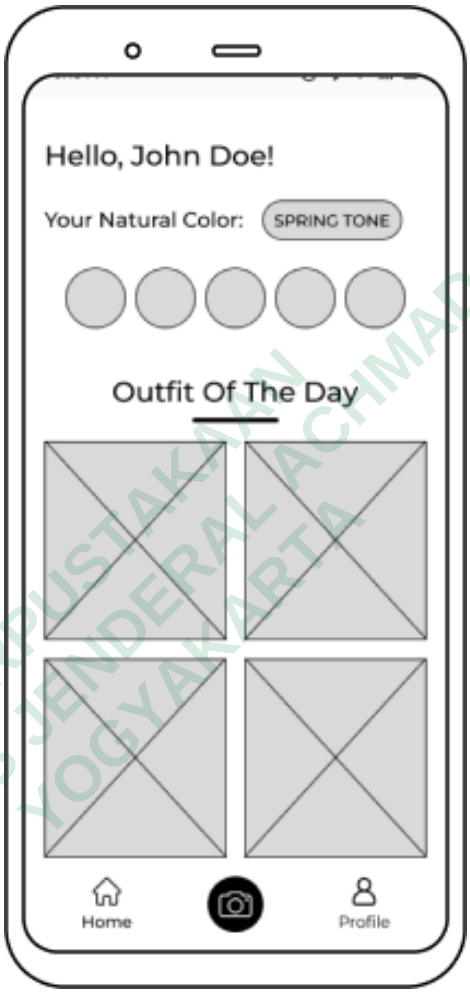
Halaman *login* digunakan sebagai gerbang masuk pengguna sebelum dapat mengakses tampilan utama dalam aplikasi. Tampilan *wireframe* halaman *login* terdapat pada Gambar 4.6.

Gambar 4.6 *Wireframe* halaman *login*

Halaman *login* juga berbentuk sebuah *form*. Pengguna harus memasukkan *email* dan *password* yang terdaftar dalam aplikasi. Data yang dimasukkan harus tersedia dan sesuai dengan yang ada pada *database*.

3. *Wireframe* Halaman *Home*

Halaman *home* merupakan tampilan utama dalam aplikasi. Halaman ini menyajikan hasil analisis warna yang terakhir disimpan dan rekomendasi gaya berpakaian. *Wireframe* halaman login terdapat pada Gambar 4.7.

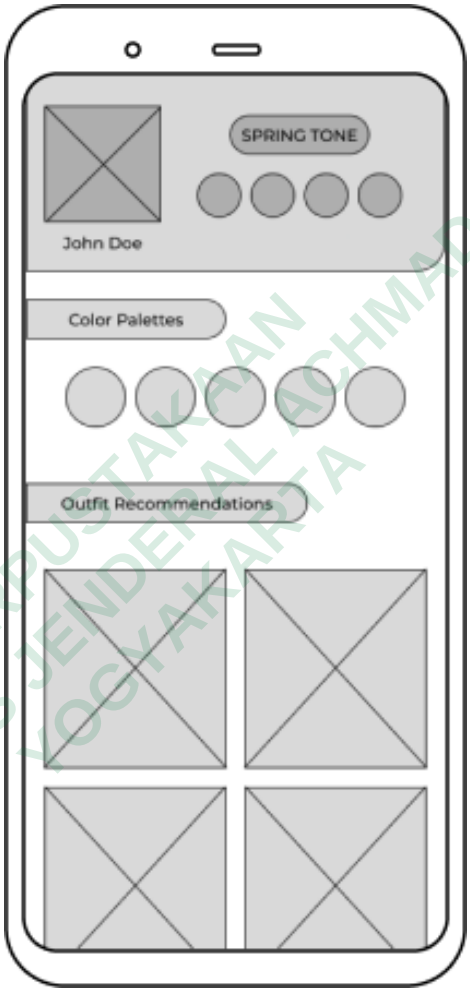


Gambar 4.7 *Wireframe* halaman *home*

Halaman *login* menampilkan kategori warna pengguna, palet warna, dan rekomendasi gaya berpakaian yang sesuai dengan hasil analisis warna personal.

4. *Wireframe* Halaman *Result*

Halaman *result* merupakan halaman yang menyajikan hasil analisis warna personal. Berbagai informasi dimuat pada halaman ini setelah sistem selesai melakukan analisis warna. Rancangan *wireframe* halaman *result* seperti ditunjukkan pada Gambar 4.8.



Gambar 4.8 *Wireframe* halaman *result*

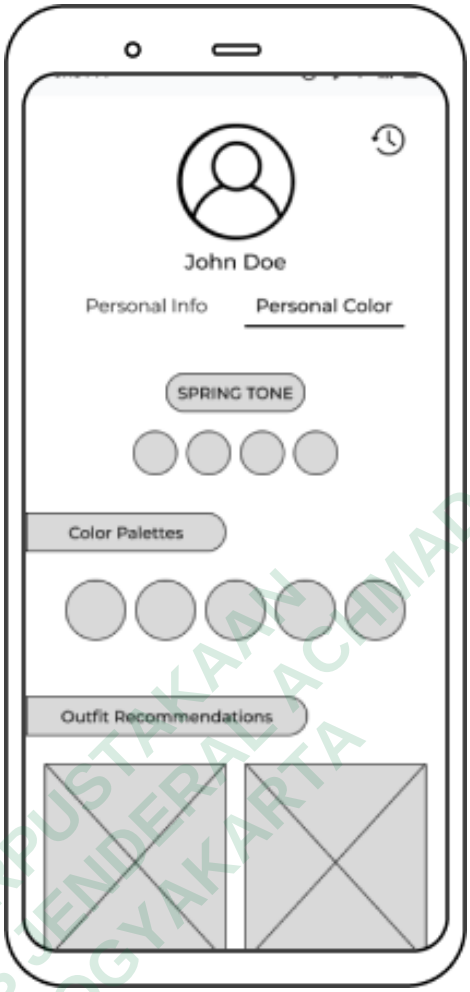
Informasi yang disajikan pada halaman *result* meliputi kategori warna pengguna, palet warna yang cocok dengan kategori warna, dan rekomendasi gaya pakaian yang telah dipersonalisasi.

5. *Wireframe* Halaman *Profile*

Halaman *profile* merupakan halaman untuk menyajikan informasi dari pengguna. Informasi pada halaman ini dibagi menjadi 2 tab, yaitu tab "*Personal Info*" dan tab "*Personal Color*". *Wireframe* halaman *profile* bagian tab "*Personal Info*" terdapat pada Gambar 4.9.

Gambar 4.9 *Wireframe profile* tab *personal info*

Tab "*Personal Info*" mencakup informasi pribadi pengguna, antara lain nama, *email*, *password*, dan jenis kelamin. Pengguna juga dapat melakukan *log out* pada tab ini. Sedangkan tab "*Personal Color*" terdapat pada Gambar 4.10.



Gambar 4.10 *Wireframe profile* tab *personal color*

Tab "*Personal Color*" berisi hasil analisis warna terakhir yang disimpan pengguna. Sama seperti pada halaman *result*, konten pada halaman ini berupa kategori warna pengguna, palet warna, dan rekomendasi gaya pakaian.

4.3 PENGKODEAN (*CODING*)

4.3.1 Pengembangan Model *Personal Color Analysis*

Model analisis warna dikembangkan menggunakan metode *Convolutional Neutral Networks* (CNN). Dataset yang digunakan berisi 200 data gambar wajah orang Indonesia yang diberi label berdasarkan *tone* warna musim (*Spring*, *Summer*, *Autumn*, atau *Winter*). Pembuatan program dilakukan menggunakan bahasa pemrograman *Python* dengan *tools Google Colaboratory*. Tahap pertama dalam pembuatan model adalah mengambil *dataset* yang telah disimpan dalam *Google Drive* lalu melakukan *pre-processing data*. Kode untuk melakukan *pre-processing data* dapat dilihat di bawah ini.

Kode Program 4.1 *Preprocessing data*

```
datagen = ImageDataGenerator(rescale=1./255)

train_gen = datagen.flow_from_dataframe(
    dataframe=train_df,
    x_col='filepath',
    y_col='label',
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical')

val_gen = datagen.flow_from_dataframe(
    dataframe=val_df,
    x_col='filepath',
    y_col='label',
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical')
```

Pre-processing data pada Kode Program 4.1 memanfaatkan *ImageDataGenerator* dari *library TensorFlow Keras*. *Library* ini digunakan untuk membuat *data generator* yang berguna dalam melakukan tahap *training* dan *validation*. *Generator* akan memuat *dataset* gambar lalu melakukan *pre-processing data*. Tahap selanjutnya setelah *pre-processing* selesai dilakukan adalah melakukan klasifikasi gambar menggunakan CNN.

Kode Program 4.2 Klasifikasi data

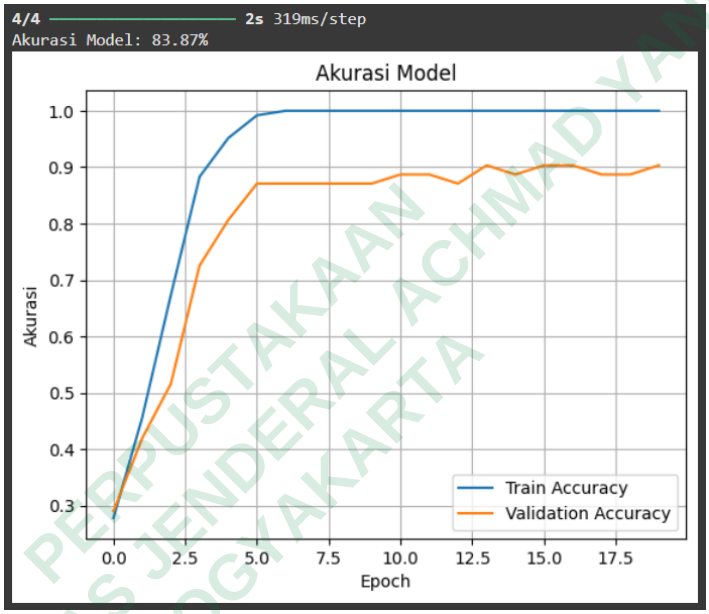
```
1 | model = Sequential([
2 |     Conv2D(32, (3,3), activation='relu', input_shape=(224, 224,
3)),
3 |     MaxPooling2D(2, 2),
4 |
5 |     Conv2D(64, (3,3), activation='relu'),
6 |     MaxPooling2D(2, 2),
7 |
8 |     Flatten(),
9 |     Dropout(0.3),
10|     Dense(128, activation='relu'),
11|     Dense(4, activation='softmax')])
```

Proses klasifikasi yang ditunjukkan pada Kode Program 4.2 memanfaatkan *Keras Sequential API* yang berada pada baris 1. *Conv2D* yang berada pada baris 2 dan baris 5 digunakan untuk mengekstraksi fitur wajah dari gambar. Kode program pada baris 3 dan 6 yaitu *MaxPooling2D* digunakan untuk melakukan operasi *max pooling* yang bertujuan untuk mengurangi kompleksitas dalam komputasi gambar. *Dropout* yang terdapat pada baris 9 berfungsi untuk menetapkan sebagian data secara acak ke 0 pada saat *data training* sehingga dapat mencegah *overfitting*. Tahap selanjutnya adalah melatih data yang ditunjukkan pada Kode Program 4.3.

Kode Program 4.3 *Data training*

```
1 | model.compile(
2 |     optimizer='adam',
3 |     loss='categorical_crossentropy',
4 |     metrics=['accuracy'])
5 |
6 | history = model.fit(
7 |     train_gen,
8 |     validation_data=val_gen,
9 |     epochs=20)
```

Model CNN sebelumnya dilakukan *compile* sebelum melatih data. Data kemudian dilatih secara iterasi dalam 20 sesi yang ditunjukkan pada baris 9. Hal ini bertujuan untuk mengevaluasi model dalam tiap sesi pelatihan. Tahap terakhir setelah *data training* selesai dilakukan adalah mengevaluasi model yang telah dilatih. Evaluasi divisualisasikan dalam grafik yang mengukur akurasi serta performa *data training* dan *validation* selama pelatihan. Visualisasi grafik untuk evaluasi model terdapat pada Gambar 4.11.



Gambar 4.11 Grafik evaluasi model CNN

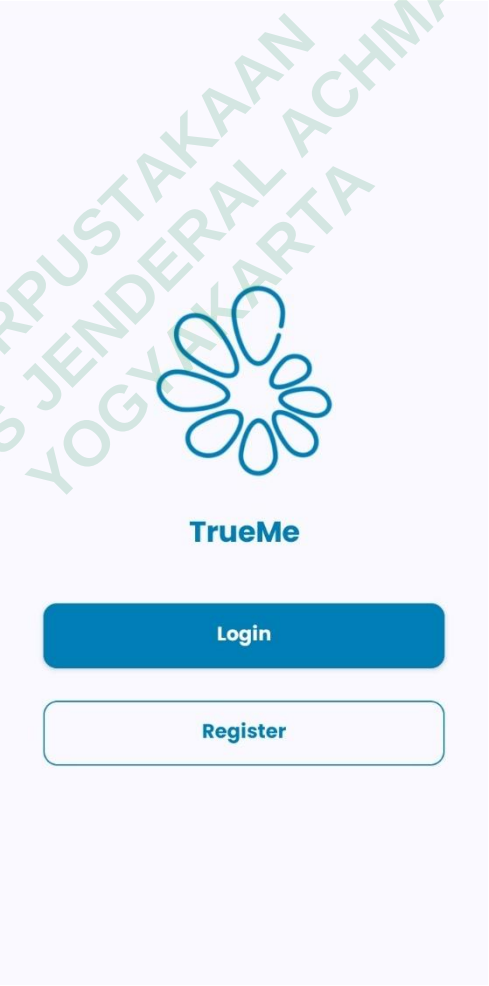
Hasil evaluasi model CNN menunjukkan tingkat akurasi model sebesar 83,87%. Walau akurasi modelnya yang terbilang bagus, dari grafik yang dihasilkan menunjukkan model mengalami *overfitting*. *Overfitting* merupakan sebuah kondisi ketika model terlalu menghafal data pelatihan dan kurang melakukan generalisasi pada data baru (validasi). Hal ini dapat dilihat pada grafik yang menunjukkan akurasi terhadap data pelatihan terlalu tinggi (hampir 100%), namun akurasi terhadap data validasi stagnan pada 90% dengan sedikit fluktuasi. Hal ini menyebabkan model memiliki performa yang buruk pada data yang belum pernah dilihat atau *data testing*. Selain itu, hal ini juga dapat menyebabkan model mengalami kesalahan dalam klasifikasi.

4.3.2 Implementasi Kode Sistem

Tahap selanjutnya setelah model *personal color analysis* berhasil dibuat adalah mengimplementasikan desain antarmuka pada tahap perancangan ke dalam kode program. Sistem dikembangkan dengan bahasa pemrograman Kotlin. Implementasi kode pada tiap halaman dan fitur akan dijabarkan sebagai berikut.

1. Halaman *Onboarding*

Halaman *onboarding* merupakan halaman awal yang diperlihatkan kepada pengguna ketika pertama kali memasang dan membuka aplikasi. Halaman *onboarding* berisi dua tombol yang diperuntukkan sebagai navigasi ke halaman *register* atau *login*. Halaman onboarding dapat dilihat pada Gambar 4.12.



Gambar 4.12 Halaman *onboarding*

Berikut pada Kode Program 4.4 merupakan kode sumber dari halaman *onboarding*.

Kode Program 4.4 Kode *activity* halaman *onboarding*

```
1 | private fun setupView() {
2 |     @Suppress("DEPRECATION")
3 |     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
4 |         window.insetsController?.hide(WindowInsets.Type.statusBars
())
5 |     } else {
6 |         window.setFlags(
7 |             WindowManager.LayoutParams.FLAG_FULLSCREEN,
8 |             WindowManager.LayoutParams.FLAG_FULLSCREEN
9 |         )
10|     }
11|     supportActionBar?.hide()
12| }
13|
14| private fun setupAction() {
15|     binding.btnLogin.setOnClickListener {
16|         startActivity(Intent(this, LoginActivity::class.java))
17|     }
18|
19|     binding.btnRegister.setOnClickListener {
20|         startActivity(Intent(this, RegisterActivity::class.java))
21|     }
22| }
```

Tampilan halaman *onboarding* ditulis dalam fungsi *setupView()* yang berada pada baris 1 sampai 12. Aksi tombol seperti berpindah ke halaman *login* dan *register* diatur dalam fungsi *setupAction()* yang terdapat pada baris 14 sampai 22.

2. Halaman *Register*

Halaman *register* adalah halaman yang digunakan pengguna untuk membuat akun dalam aplikasi. Pengguna diharuskan mengisi data nama, *email*, *password*, dan jenis kelamin sebagai persyaratan dalam pendaftaran akun. Tampilan halaman register dapat dilihat pada Gambar 4.13.

Gambar 4.13 Halaman *register*

Kode program untuk tampilan halaman *register* tertera pada Kode Program 4.5. Kode program tersebut merupakan kode untuk tampilan antarmuka dan pengaturan logika penanganan aksi.

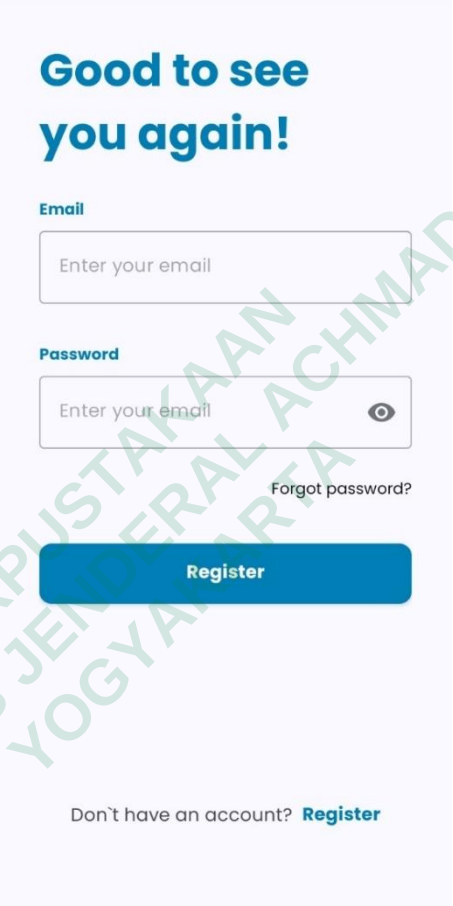
Kode Program 4.5 Kode *activity* halaman *register*

```
1 | binding.btnRegister.setOnClickListener {
2 |       val name = binding.etName.toString()
3 |       val email = binding.etEmail.toString()
4 |       val password = binding.etPassword.toString()
5 |       val gender = binding.spinnerGender.selectedItem.toString()
6 |
7 |       viewModel.register(name, email, password, gender)
8 | }
9 |
10| viewModel.registrationSuccess.observe(this) { success ->
11|     if (success) {
12|         Toast.makeText(this, "Registration Successful",
Toast.LENGTH_SHORT).show()
13|         startActivity(Intent(this, MainActivity::class.java))
14|         finish()
15|     } else {
16|         Toast.makeText(this, "Registration Failed",
Toast.LENGTH_SHORT).show()
17|     }
18| }
```

Kode Program 4.5 merupakan potongan kode dari *activity register*. Kode tersebut merupakan pemanggilan *binding* untuk menambahkan aksi pada tombol register. Pemanggilan *binding* dilakukan dengan cara seperti yang ditunjukkan pada baris 1. Fungsi tombol yang menunjukkan status berhasil tidaknya proses *register* ditulis pada baris 10 sampai 18.

3. Halaman *Login*

Halaman *login* merupakan gerbang sebelum pengguna dapat mengakses fitur-fitur dalam aplikasi. Pengguna harus memasukkan *email* dan *password* yang terdaftar dan sesuai dengan yang tersimpan dalam *database* sistem. Pengguna dapat masuk ke halaman utama aplikasi jika akun tervalidasi oleh sistem.



Gambar 4.14 Halaman *login*

Gambar 4.14 merupakan tampilan dari halaman *login*. Selain melakukan *login*, terdapat fitur bagi pengguna yang lupa dengan *password*-nya. Kode Program 4.6 digunakan untuk membuat halaman *login*.

Kode Program 4.6 Kode *activity* halaman *login*

```
1 | binding.btnLogin.setOnClickListener {
2 |     val email = binding.etEmail.toString()
3 |     val password = binding.etPassword.toString()
4 |     viewModel.login(email, password)
5 | }
6 |
7 | viewModel.loginSuccess.observe(this) { success ->
8 |     if (success) {
9 |         Toast.makeText(this, "Login successful",
Toast.LENGTH_SHORT).show()
10|                  startActivity(Intent(this,
MainActivity::class.java))
11|                 finish()
12|             } else {
13|                 Toast.makeText(this, "Login failed",
Toast.LENGTH_SHORT).show()
14|             }
15|         }
```

Kode Program 4.6 juga merupakan potongan kode dari *activity login*. Sama seperti pada *register*, halaman *login* juga menggunakan *binding* untuk mengatur tampilan dan aksi dalam halaman. Fungsi tombol diatur dalam *viewModel* yang terdapat pada baris 7 sampai dengan baris 15.

4. Halaman *Home*

Halaman *home* merupakan halaman utama dalam aplikasi. Halaman ini terbuka setelah pengguna berhasil melakukan *login*. Halaman *home* berisi kategori warna pengguna dan palet warnanya, serta terdapat rekomendasi gaya berpakaian harian berdasarkan hasil analisis terakhir. Tampilan halaman *home* secara lebih jelas diperlihatkan pada Gambar 4.15.

Gambar 4.15 Halaman *home*

Kode program untuk halaman *home* secara lebih lengkap dijabarkan pada Kode Program 4.7.

Kode Program 4.7 Kode *activity* halaman *home*

```
1 | viewModel.userName.observe(viewLifecycleOwner) {
2 |     binding.tvGreeting.text = "Hello, $it!"
3 | }
4 |
5 | viewModel.naturalColor.observe(viewLifecycleOwner) {
6 |     binding.tvTone.text = it
7 | }
8 |
9 | val adapter = OutfitAdapter()
10|     binding.rvOutfits.layoutManager =
GridLayoutManager(requireContext(), 2)
11|     binding.rvOutfits.adapter = adapter
12|
13| viewModel.outfitList.observe(viewLifecycleOwner) {
14|     adapter.submitList(it)}
```

Berbeda dengan halaman lain yang menggunakan *activity*, halaman ini menggunakan *fragment*. Hal ini dikarenakan halaman *home* merupakan salah satu halaman yang berada pada *bottom navigation bar*. Palet warna diambil dari *database* dan disajikan pada halaman menggunakan kode program baris 5 sampai 7. Rekomendasi gaya berpakaian harian dibuat memanfaatkan adapter *OutfitAdapter()* yang dibuat dalam *grid layout* dengan kode program pada baris 9 sampai 14.

5. Halaman *Result*

Halaman *result* merupakan halaman yang mempresentasikan hasil analisis warna. Halaman ini memberikan informasi jenis *tone* warna pengguna, warna pakaian yang cocok dikenakan, dan rekomendasi gaya berpakaiannya. Halaman ini dapat dilihat pada Gambar 4.16.

Gambar 4.16 Halaman *result*

Halaman *result* terbuka setelah sistem selesai melakukan analisis warna kulit. Alur analisis dimulai dengan pengguna menekan tombol kamera yang posisinya berada di bagian tengah pada *bottom navigation bar*. Setelah itu, *bottom sheet* terbuka dan memberikan opsi untuk mengunggah gambar dari galeri atau mengambil gambar dari kamera. Ketika gambar berhasil diunggah, sistem akan memroses gambar mulai dari mengekstraksi area wajah menggunakan *Google ML Kit Face Detection*, lalu melakukan analisis warna kulit dengan model yang dibuat

sebelumnya. Tahap terakhir adalah mengklasifikasikannya sesuai kategori *tone* warna dan menampilkan palet warna yang sesuai dan rekomendasi gaya berpakaiannya.

Daftar palet warna yang digunakan dalam aplikasi ini mengambil rujukan dari artikel yang berasal dari *platform* "*The Concept Wardrobe*", yang merupakan konsultan gaya personal yang berfokus pada penciptaan gaya busana pribadi (The Concept Wardrobe, 2023). 5 warna dipilih pada tiap kategori *tone* warna untuk digunakan sebagai palet warna pada halaman *result*. Rekomendasi gaya pakaian dikumpulkan melalui platform *Pinterest* dengan mengacu pada warna yang telah diperoleh sebelumnya.

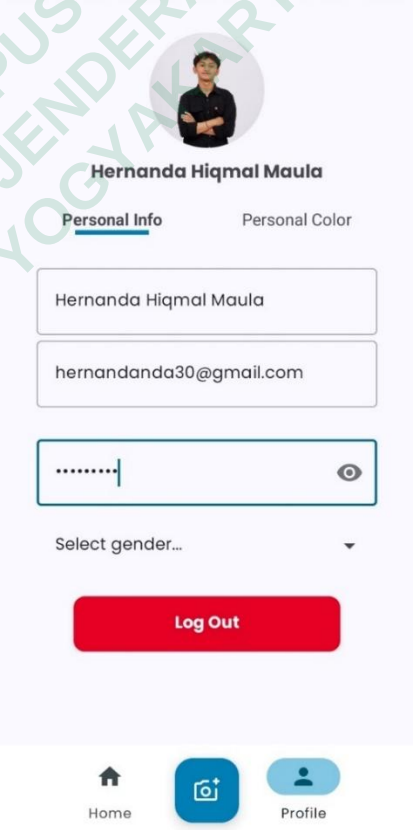
Kode Program 4.8 Kode *activity* halaman *result*

```
1 | private fun analyze(bitmap: Bitmap) {
2 |     detector.process(image)
3 |         .addOnSuccessListener { faces ->
4 |             if (faces.isNotEmpty()) {
5 |                 val face = faces[0]
6 |                 val centerX = face.boundingBox.centerX()
7 |                 val centerY = face.boundingBox.centerY()
8 |                 val pixel = bitmap.getPixel(centerX, centerY)
9 |
10|                 val input = preprocessColor(pixel)
11|                 val output = Array(1) { FloatArray(4) }
12|                 tflite.run(input, output)
13|
14|                 val labelIndex = output[0].indices.maxByOrNull
{ output[0][it] } ?: -1
15|                 val tone = labels[labelIndex]
16|
17|                 viewModel.setTone(tone)
18|                 viewModel.loadOutfits(tone)
19|             } else {
20|                 viewModel.setTone("Tidak terdeteksi")
21|             }
22|         }
23| }
```

Kode Program 4.8 merupakan kode untuk melakukan proses analisis *personal color*. Gambar yang diunggah diproses menggunakan *library ML Kit Face Detection* melalui *detector.process(image)* untuk mendeteksi keberadaan wajah dengan kode program pada baris 2. Jika wajah berhasil ditemukan, sistem akan menentukan titik tengah dari *bounding box* wajah tersebut. Warna kulit di titik tengah diambil menggunakan fungsi *getPixel()* seperti ditunjukkan pada baris 8. Nilai warna yang diperoleh lalu diolah melalui fungsi *preprocessColor()* agar sesuai dengan bentuk *input* yang dibutuhkan oleh model *machine learning*. Model kemudian dijalankan menggunakan *tflite.run()* yang ditunjukkan pada baris 12 untuk memprediksi *tone* warna kulit yang sesuai.

6. Halaman *Profile*

Halaman *profile* memuat informasi pribadi dan informasi hasil analisis pengguna. Halaman ini terbagi menjadi 2 tab, yaitu tab "*Personal Info*" dan tab "*Personal Color*". Halaman personal info ditunjukkan pada Gambar 4.17.



Gambar 4.17 Halaman *profile* tab *Personal Info*

Tab *personal info* memuat informasi pribadi pengguna yang tersimpan dalam aplikasi seperti nama, *email*, dan *password*. Sedangkan tab *personal color* memuat informasi yang sama seperti pada halaman *result*. Hasil analisis yang dimuat pada tab *personal color* merupakan hasil analisis terakhir. Tab *personal color* terdapat pada Gambar 4.18.

Gambar 4.18 Halaman *profile* tab *personal color*

Sama seperti *home*, halaman *profile* juga diakses melalui *bottom navigation bar* sehingga halamannya menggunakan *fragment*. Setiap tab yang terdapat pada halaman *profile* dimuat dalam satu kode program yang sama, dengan pengaturan visibilitas masing-masing elemen agar tidak saling menumpuk saat berpindah tab. Kode program halaman *profile* tertera pada Kode Program 4.9.

Kode Program 4.9 Kode *fragment* halaman *profile*

```
1 | viewModel.activeTab.observe(viewLifecycleOwner) { tab ->
2 |     if (tab == "color") {
3 |         binding.layoutPersonalColor.visibility = View.VISIBLE
4 |         binding.layoutPersonalInfo.visibility = View.GONE
5 |         binding.tvPersonalColor.setTypeface(null, Typeface.BOLD)
6 |         binding.tvPersonalInfo.setTypeface(null, Typeface.NORMAL)
7 |     } else {
8 |         binding.layoutPersonalColor.visibility = View.GONE
9 |         binding.layoutPersonalInfo.visibility = View.VISIBLE
10|         binding.tvPersonalColor.setTypeface(null, Typeface.NORMAL)
11|         binding.tvPersonalInfo.setTypeface(null, Typeface.BOLD)
12|     }
13| }
```

Praktik terbaik dalam membuat halaman tab adalah dengan menggunakan *fragment*. Namun, karena halaman *profile* sudah menggunakan *fragment*, maka hal tersebut tidak dapat dilakukan. Alternatif yang digunakan adalah dengan mengatur visibilitas antar tab. Visibilitas diatur dengan mengamati perubahan nilai tab aktif dari *ViewModel*, lalu mengubah tampilan antarmuka secara dinamis menggunakan *View.VISIBLE* dan *View.GONE* pada masing-masing *layout* yang mewakili tab. Kode program untuk fungsi ini berada pada baris 1 sampai dengan 12.

4.4 PENGUJIAN (*TESTING*)

Pengujian dilakukan pada 10 responden awal menggunakan metode *System Usability Scale* (SUS). Responden menjalankan semua fitur dalam aplikasi mulai dari *login*, analisis warna personal, hingga *logout*. Metode SUS digunakan pada penelitian ini dikarenakan beberapa alasan, di antaranya:

1. Skala pengujian mudah dimengerti responden
2. Cocok untuk pengujian dengan sampel kecil, namun tetap memberikan hasil yang dapat diandalkan
3. Sistem pengukuran yang efektif dapat membedakan sistem yang mampu digunakan atau tidak.

Kuesioner SUS terdiri dari 10 pertanyaan dengan 5 pilihan jawaban berdasarkan skala likert. Tiap pilihan jawaban memiliki skor masing-masing, dimulai dari sangat tidak setuju (1) sampai sangat setuju (5). Daftar pertanyaan pada kuesioner SUS dapat dilihat pada Tabel 4.8.

Tabel 4.8 Daftar pertanyaan SUS

No.	Pertanyaan
1.	Saya berpikir akan menggunakan sistem ini lagi.
2.	Saya merasa sistem ini rumit untuk digunakan.
3.	Saya merasa sistem ini mudah digunakan.
4.	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini.
5.	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya.
6.	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini).
7.	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat.
8.	Saya merasa sistem ini membingungkan.
9.	Saya merasa tidak ada hambatan dalam menggunakan sistem ini.
10.	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini.

Seperti yang telah disebutkan sebelumnya, SUS memiliki 5 pilihan jawaban dengan skor yang berbeda-beda. Penjabaran tiap pilihan jawaban dan skor yang dimiliki terdapat pada Tabel 4.9.

Tabel 4.9 Pilihan jawaban dan skor pada SUS

Jawaban	Skor
Sangat Tidak Setuju (STS)	1
Tidak Setuju (TS)	2
Ragu-ragu (RG)	3
Setuju (S)	4
Sangat Setuju (SS)	5

Setelah pengumpulan data dari responden selesai dilakukan, langkah selanjutnya adalah melakukan perhitungan nilai. Perhitungan skor SUS mengikuti beberapa aturan berikut:

1. Setiap pertanyaan dengan nomor ganjil, skor dikurangi 1 (*x-1*).
2. Setiap pertanyaan dengan nomor genap, skor dihitung dengan rumus 5 dikurangi skor yang diperoleh (*5-x*).
3. Total skor SUS diperoleh dari penjumlahan seluruh skor yang telah disesuaikan, kemudian dikalikan dengan 2,5.

Selanjutnya, skor SUS dari masing-masing responden dihitung menggunakan rumus perhitungan standar SUS yang telah ditetapkan, guna memperoleh nilai rata-rata sebagai indikator tingkat kegunaan sistem secara keseluruhan. Berikut merupakan rumus untuk perhitungan skor SUS.

$$\underline{x} = \frac{\sum x}{n}$$

$$\underline{x = \text{Skor rata} - \text{rata}}$$

$$\sum x = \text{Jumlah skor SUS}$$

$$n = \text{Jumlah responden}$$

Skor akhir yang diperoleh diklasifikasikan ke dalam lima kategori dan diinterpretasikan dalam bentuk nilai huruf, mulai dari A hingga F. Pedoman umum mengenai interpretasi skor SUS disajikan pada Tabel 4.10.

Tabel 4.10 Interpretasi skor SUS

SUS Score	***Grade***	***Adjective Rating***
>80.3	A	*Excellent*
68-80.3	B	*Good*
68	C	*Acceptable*
51-68	D	*Poor*
<51	F	*Awful*

4.4.1 Hasil Pengujian SUS

Pengujian aplikasi dilakukan berdasarkan kuesioner yang sesuai dengan ketentuan SUS dan melibatkan 10 responden. Rekapitulasi jawaban responden dan hasil perhitungan skor SUS tertera pada Tabel 4.11 sebagai berikut.

Tabel 4.11 Hasil Perhitungan SUS

Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Skor SUS
R1	4	1	5	1	5	2	5	1	4	2	90
R2	2	3	3	1	3	4	4	3	1	1	53
R3	4	1	5	1	4	2	5	1	4	1	90
R4	3	3	4	2	3	3	4	2	4	2	65
R5	5	2	4	1	4	3	5	2	5	2	83
R6	4	2	5	1	4	1	5	2	3	1	85
R7	3	2	4	2	5	3	3	3	4	2	68
R8	1	2	4	2	1	4	3	4	1	3	38
R9	3	2	3	2	3	3	4	3	3	2	60
R10	2	3	2	3	2	4	3	3	2	2	40
Skor Rata-rata (Hasil Akhir)											67

Setelah menyelesaikan tahap pengisian kuesioner, dilakukan perhitungan skor rata-rata dan diperoleh nilai akhir 67 pada pengujian aplikasi. Berdasarkan pedoman yang terdapat pada Tabel 4.10, maka nilai tersebut termasuk dalam kategori "*Acceptable*" atau "C" *Grade.* Hal ini menunjukkan bahwa tingkat *usability* aplikasi berada pada tingkat sedang (*marginally acceptable*). Dengan begitu, meski aplikasi telah memenuhi standar minimum usabilitas, masih terdapat banyak ruang untuk dilakukan perbaikan dan peningkatan.

4.5 PEMBAHASAN

Berdasarkan hasil analisis, rancangan, implementasi, serta pengujian aplikasi pada penelitian ini, menghasilkan sebuah Informasi bahwa pengembangan aplikasi *personal color analysis* menggunakan metode XP terbilang cukup bagus dan cocok digunakan. Hal ini terbukti dari proses pengembangan aplikasi yang selesai pada akhir bulan Juni, sesuai dengan jadwal penelitian yang telah ditentukan seperti terlampir pada Lampiran 2. Dengan iterasi yang dilakukan, penulis mendapatkan umpan balik target pengguna secara langsung sehingga proses perbaikan sistem dapat dilakukan secepat mungkin.

Penerapan teknologi *Google ML Kit face detection* juga efektif digunakan untuk melakukan operasi pengekstrakan gambar wajah untuk *personal color analysis*. Dengan hasil deteksi yang konsisten dan akurat memberikan koordinat titik area wajah, membuat teknologi ini memenuhi tujuan dalam membantu proses analisis warna personal. Masalah timbul pada model yang dibangun karena mengalami *overfitting* sehingga hasil analisis kurang konsisten dan tidak akurat pada data baru. Hal ini terlihat dari hasil evaluasi model yang menunjukkan akurasi terhadap *data validation* yang stagnan di angka 90% dan sedikit fluktuatif, tidak sebanding dengan akurasi terhadap *data training* yang stabil di angka 100%, meski model CNN mendapatkan tingkat akurasi sebesar 83,87%.

Karena masalah model yang tidak akurat tersebut, pengguna mengalami ketidakpuasan dalam menggunakan aplikasi. Muncul juga keraguan pada pengguna akan hasil analisis warna yang mereka dapatkan. Hal ini dibuktikan melalui skor rata-rata yang diperoleh saat pengujian SUS, yaitu 67. Hal tersebut dapat menjadi perhatian sebagai bahan evaluasi dalam pengembangan sistem serupa ke depannya.