BAB IV
PEMBAHASAN

4.1 Analisis Kebutuhan dan Konsep Awal Aplikasi

Konsep penelitian ini muncul dari analisis mendalam terhadap permasalahan meningkatnya angka pengangguran di Indonesia, khususnya di kalangan lulusan perguruan tinggi yang kurang memiliki pengalaman. Salah satu hambatan utamanya adalah kurangnya keahlian dalam hal kolaborasi bersama rekan tim. Dengan memahami tantangan tersebut, maka dibuatlah aplikasi Collabolio sebagai solusi yang dapat memfasilitasi para lulusan dan profesional muda untuk berkolaborasi dalam suatu *project*.

Nama "Collabolio" diambil dari gabungan kata "*collaboration*" dan "*portfolio*". Hal ini mencerminkan tujuan utama dari aplikasi tersebut, yaitu memfasilitasi kolaborasi antara individu dalam berbagai proyek serta memungkinkan mereka untuk membangun dan memamerkan portofolio profesional mereka. Dengan demikian, nama "Collabolio" tidak hanya mencerminkan esensi dan fokus utama aplikasi, tetapi juga agar mudah diingat dan memiliki daya tarik yang kuat.

Dari Konsep awal tersebut, kemudian dilakukan analisis fitur yang tersedia dalam *dating apps* dengan mengambil dari referensi aplikasi Tinder atau Bumble dan juga aplikasi LinkedIn sehingga dapat di implementasikan pada aplikasi Collabolio. Berikut adalah beberapa fiturnya.

Tabel 4.1 Analisis Fitur Aplikasi

No	Fitur	Deskripsi	Tingkat Kesulitan	Implementasi yang Digunakan
1	Sistem Pencocokan (Matching System)	Menggunakan algoritma untuk mencocokkan pengguna berdasarkan keahlian, minat, dan profil proyek.	Tinggi	Integrasi Algoritma *Machine Learning*, Android Retrofit
2	Profil Pengguna Terperinci	Memungkinkan pengguna untuk membuat dan mengedit profil dengan informasi	Sedang	Firebase Firestore (CRUD), Android MVVM

		terperinci seperti pendidikan, pengalaman, dan keahlian.		
3	Fitur Swipe	Memungkinkan pengguna untuk mengevaluasi profil rekan potensial dengan cepat dengan menggeser ke kanan (minat) atau ke kiri (tidak cocok).	Sedang	Android RecyclerView dan MVVM
4	Pesan Langsung	Memungkinkan pengguna untuk berkomunikasi secara langsung dengan rekan kolaborasi potensial.	Tinggi	Firebase Cloud Messaging
5	Notifikasi Aktivitas	Memberi tahu pengguna tentang aktivitas baru seperti match baru, pesan masuk, atau pembaruan proyek yang relevan.	Sedang	Firebase Cloud Messaging
6	Verifikasi Profil	Memungkinkan pengguna untuk memverifikasi identitas mereka untuk meningkatkan keamanan dan kepercayaan.	Sedang	Firebase Authentication
7	Pengaturan Privasi	Memungkinkan pengguna untuk mengatur preferensi privasi mereka, termasuk siapa yang dapat melihat profil dan mengirim pesan.	Sedang	Android SharedPreferences API

Berdasarkan tabel 4.1 diatas, diambil beberapa fitur yang dapat diimplementasikan dalam aplikasi Collabolio yaitu sistem pencocokan (Matching System) dengan algoritma *Machine Learning*, Fitur profil pengguna yang terperinci, Fitur Swipe, serta fitur verifikasi profil dan edit profil. Beberapa fitur lain seperti Notifikasi Aktivitas, Pengaturan Privasi, dan Pesan Langsung tidak akan diimplementasikan karena beberapa alasan yaitu tingkat kesulitan yang tinggi

serta ketersinambungan antara ketiganya sehingga menambah kompleksitas dari aplikasi. Sebagai alternatif, percakapan pesan akan dialihkan ke aplikasi lain seperti ke aplikasi pesan atau ke aplikasi email menggunakan *Implicit Intent* Android. *Implicit Intent* sendiri merupakan sebuah mekanisme yang dapat memungkinkan aplikasi untuk berpindah dan mengirim data ke aplikasi lain. Dengan demikian, pengguna masih dapat berkomunikasi dengan rekan kolaborasi mereka dengan mudah melalui aplikasi pesan yang sudah tersedia.

4.2 Desain Logo Aplikasi

Logo merupakan elemen penting dalam perancangan aplikasi karena merupakan representasi visual dari merek dan identitas aplikasi tersebut. Dalam konteks aplikasi Collabolio, logo akan menjadi elemen yang membedakan aplikasi ini dari yang lain, mencerminkan nilai-nilai seperti kolaborasi, inovasi, dan profesionalisme. Desain logo yang kuat dan menarik akan membantu memperkuat citra merek Collabolio di mata pengguna dan membangun koneksi emosional dengan mereka. Oleh karena itu, desain logo perlu memperhatikan aspek-aspek seperti kesan visual, keterbacaan, daya ingat, serta kesesuaian dengan tema dan tujuan aplikasi.

Gambar 4.1 Logo Aplikasi Collabolio

Gambar 4.1 merupakan tampilan desain logo dari aplikasi Collabolio. Bentuk dari logo tersebut merupakan gabungan dari beberapa potongan huruf yang

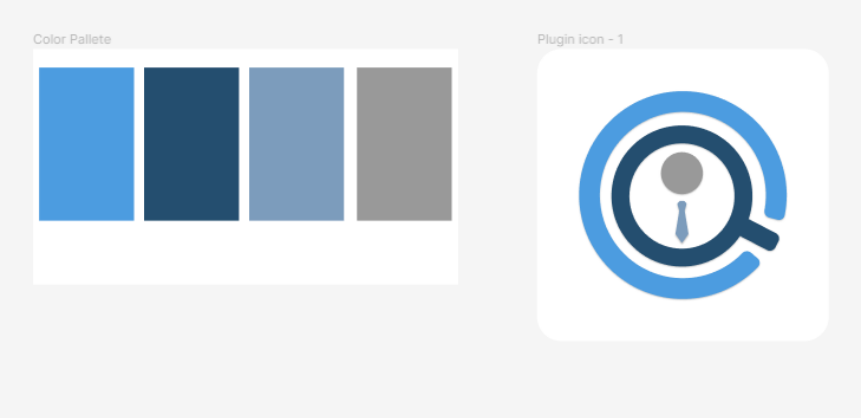
terdapat pada nama Collabolio. Berikut penjelasan makna dari setiap elemen yang terdapat pada logo.

Pertama, Lingkaran terluar dari logo yang membentuk huruf C memiliki makna kesatuan dan inklusivitas. Hal ini melambangkan sifat kolaboratif Collabolio, di mana individu-individu berkumpul dari berbagai latar belakang dan keahlian untuk berkolaborasi. bentuk lingkaran juga menunjukkan bentuk komunitas dan keterhubungan, menyoroti pentingnya kerja tim dan dukungan bersama dalam mencapai tujuan yang sama. Secara keseluruhan, lingkaran luar dalam logo memperkuat nilai-nilai inti kolaborasi dan kerjasama yang menandai Collabolio.

Kedua, Lingkaran yang membentuk gambar kaca pembesar pada logo memiliki makna pencarian, eksplorasi, dan penemuan. Hal tersebut mencerminkan semangat inovasi dan keingintahuan yang akan menjadi bagian dari pengalaman pengguna aplikasi Collabolio. Kaca pembesar sering kali diasosiasikan dengan pencarian informasi, pemahaman yang lebih dalam, dan penemuan hal-hal baru. Dalam konteks Collabolio, kaca pembesar melambangkan upaya pengguna untuk mengeksplorasi peluang kolaborasi, memperluas jaringan profesional, dan menemukan rekan *project* yang sesuai dengan minat dan keahlian mereka.

Terakhir, Lingkaran dan dasi yang terdapat di dalam kaca pembesar membentuk huruf I dan O, atau terlihat membentuk figur manusia berdasi, memiliki makna formalitas dan profesionalisme. Bentuk tersebut juga merupakan representasi dari portofolio pengguna aplikasi. Dengan demikian, simbol tersebut mencerminkan pentingnya portofolio dalam platform Collabolio sebagai cara untuk mempresentasikan keterampilan, proyek-proyek sebelumnya, dan pencapaian profesional pengguna kepada orang lain.

Penggunaan warna pada logo juga sangat berpengaruh terhadap elemen aplikasi. Warna yang dipilih tidak hanya mencerminkan identitas merek Collabolio tetapi juga dapat mempengaruhi persepsi pengguna terhadap aplikasi. Pemilihan warna yang tepat dapat membantu memperkuat pesan merek, menarik perhatian pengguna, dan menciptakan atmosfer yang sesuai dengan tujuan dan nilai-nilai aplikasi. Berikut palet warna yang digunakan pada logo.

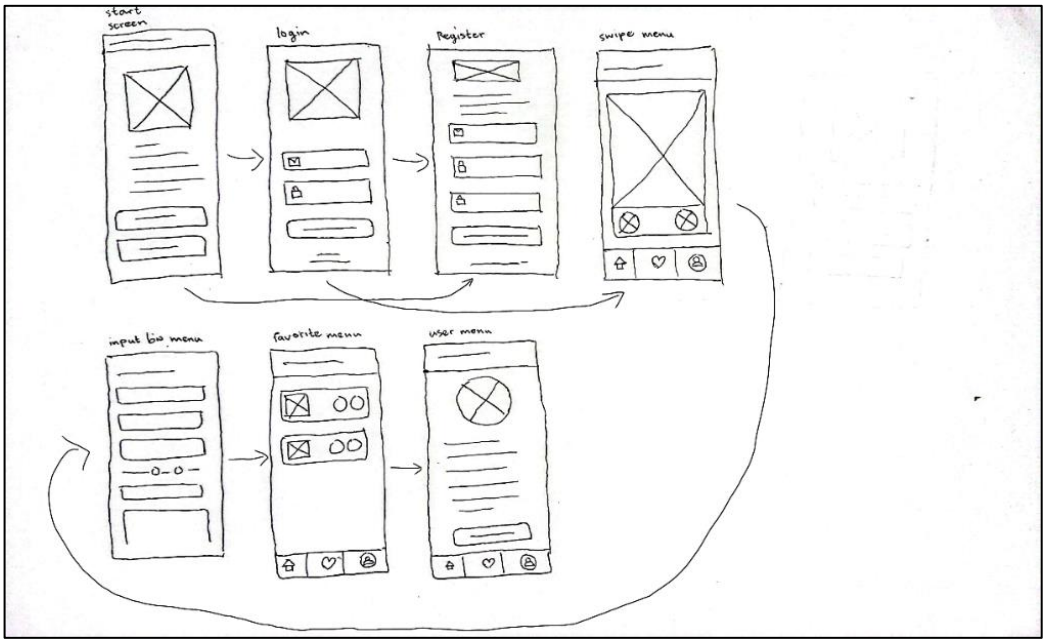


Gambar 4.2 Palet Warna Pada Logo

Penggunaan warna utama untuk aplikasi Collabolio adalah dari rentang warna biru cerah sampai biru gelap dengan tambahan warna abu-abu. Pemilihan warna ini tidak hanya mencerminkan unsur kepercayaan dan profesionalisme, tetapi juga menimbulkan kesan yang tenang dan stabil bagi pengguna. Warna biru cerah dapat memberikan kesan yang segar dan inovatif, sementara warna biru gelap menambahkan sentuhan elegan dan serius. Kombinasi dari kedua warna ini menciptakan harmoni visual yang dapat memperkuat identitas aplikasi dan menciptakan pengalaman pengguna yang konsisten dan menarik di seluruh aplikasi. Dengan demikian, penggunaan warna tersebut dalam desain aplikasi Collabolio diharapkan dapat menciptakan kesan yang positif dan menarik bagi pengguna serta memperkuat citra merek secara keseluruhan.

4.3 Desain Low Fidelity dan High Fidelity

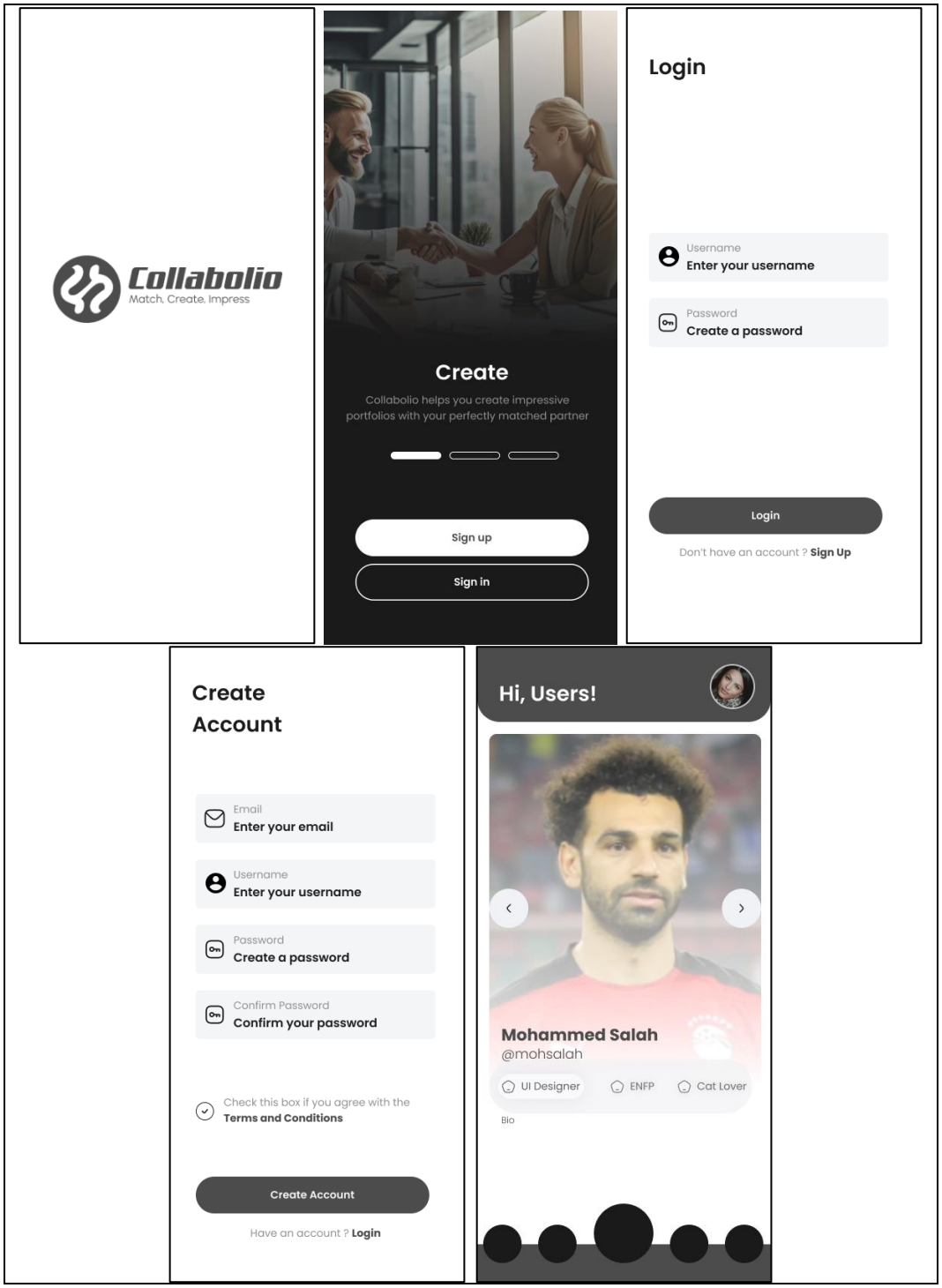
Untuk mendukung pengembangan aplikasi Collabolio, dibuat juga desain *low fidelity* (*Wireframe*) dan *high fidelity* (*Mockup*). Desain *low fidelity* akan memberikan gambaran awal tentang tata letak dan fungsi dasar aplikasi tanpa detail visual yang mendalam. Sementara itu, desain *high fidelity* akan menampilkan detail visual yang lebih lengkap, termasuk warna, tipografi, dan grafis yang lebih mendetail. Berikut untuk desain low fidelity yang dibuat menggunakan sketsa sederhana.



Gambar 4.3 Desain *Wireframe*

Pada gambar 4.3 merupakan desain *wireframe* aplikasi collabolio yang terdapat beberapa halaman yaitu *start/welcome screen* yang berfungsi sebagai tampilan awal atau sambutan kepada pengguna, halaman *login* Dimana pengguna diminta untuk memasukkan email dan kata sandi yang sudah terdaftar sebagai syarat masuk aplikasi, halaman *register* jika pengguna belum memiliki akun yang terdaftar ke dalam aplikasi, halaman menu utama yang memiliki 3 tab menu yaitu swipe menu sebagai menu utama aplikasi, favourite menu yaitu menu yang menampilkan rekan yang disukai atau diminati, dan *user* menu yang berisi biodata dari pengguna.

Setelah gambaran kasar dari tampilan aplikasi sudah didapatkan dan disetujui oleh tim pengembang, langkah berikutnya adalah mentransformasikan gambar tersebut menjadi desain *mockup* menggunakan *software* Figma. Desain *mockup* ini akan memperinci tampilan antarmuka pengguna (UI) dari aplikasi Collabolio, termasuk aspek-aspek seperti layout, warna, tipografi, dan ikon. Desain mockup ini akan menjadi dasar bagi tahap pengembangan selanjutnya, memastikan konsistensi desain dan kesesuaian dengan kebutuhan dan harapan pengguna. Berikut adalah hasilnya.



Gambar 4.4 Desain *Mockup*

Gambar 4.4 merupakan hasil desain mockup aplikasi menggunakan software Figma. Pada gambar tersebut, sudah terlihat kemiripan dengan tampilan aplikasi yang asli dengan penambahan warna sementara dan penempatan posisi komponen aplikasi yang disempurnakan, seperti tombol yang lebih terstruktur. Hal

ini akan mempermudah dalam implementasi desain UI pada aplikasi. Selain itu, dengan desain *mockup* ini juga mengurangi kemungkinan kesalahan atau perbedaan antara desain awal dan hasil akhir yang akan dilihat oleh pengguna. Desain mockup yang akurat juga memungkinkan tim untuk lebih fokus pada pengembangan fitur dan fungsionalitas aplikasi tanpa terlalu banyak membuang waktu untuk menyesuaikan desain yang tidak sesuai.

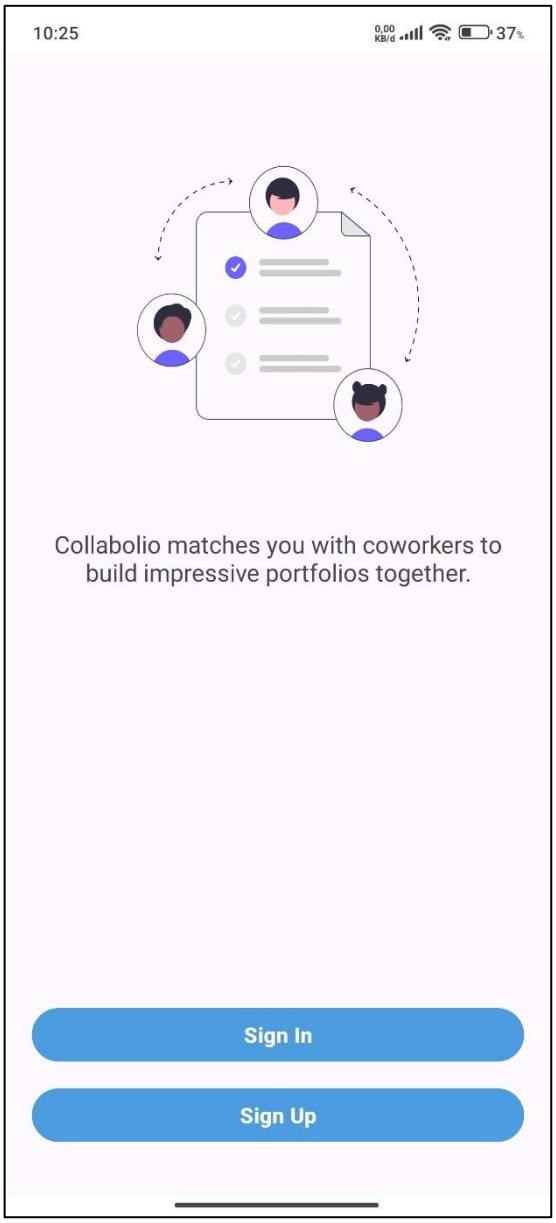
4.4 Implementasi Desain ke Dalam Aplikasi

Langkah selanjutnya adalah mengimplementasikan desain *mockup* ke aplikasi Collabolio menggunakan Software Android Studio dalam bentuk file XML. XML sendiri merupakan bahasa markup yang digunakan untuk mendefinisikan struktur dan tata letak elemen-elemen antarmuka pengguna dalam pengembangan aplikasi Android. Dalam XML, UI didefinisikan dengan menggunakan elemen-elemen seperti *TextView, Button, ImageView, LinearLayout, RelativeLayout,* dan sebagainya. Pengaturan dan tata letak elemen-elemen UI juga ditentukan oleh atribut-atribut XML seperti *layout_width, layout_height, padding, margin,* dan atribut lainnya.

Implementasi tampilan aplikasi Collabolio terdiri dari tampilan *splash screen, start/welcome screen, login screen, register screen, input bio screen,* dan 3 tab menu utama yaitu *swipe menu, favorite menu,* dan *user profile menu*. Semua tampilan menggunakan *constraint layout* yang mana lebih fleksibel untuk digunakan. *Constraint layout* sendiri merupakan jenis tata letak (*layout*) yang digunakan dalam pengembangan aplikasi Android. Tata letak ini memungkinkan pengembang untuk menentukan posisi dan hubungan antara elemen-elemen *user interface* menggunakan batasan (*constraints*). Dalam *constraint layout*, elemen-elemen UI ditempatkan relatif terhadap elemen-elemen lainnya dan terhadap batasan-batasan yang ditetapkan, seperti batasan atas, bawah, kiri, dan kanan.

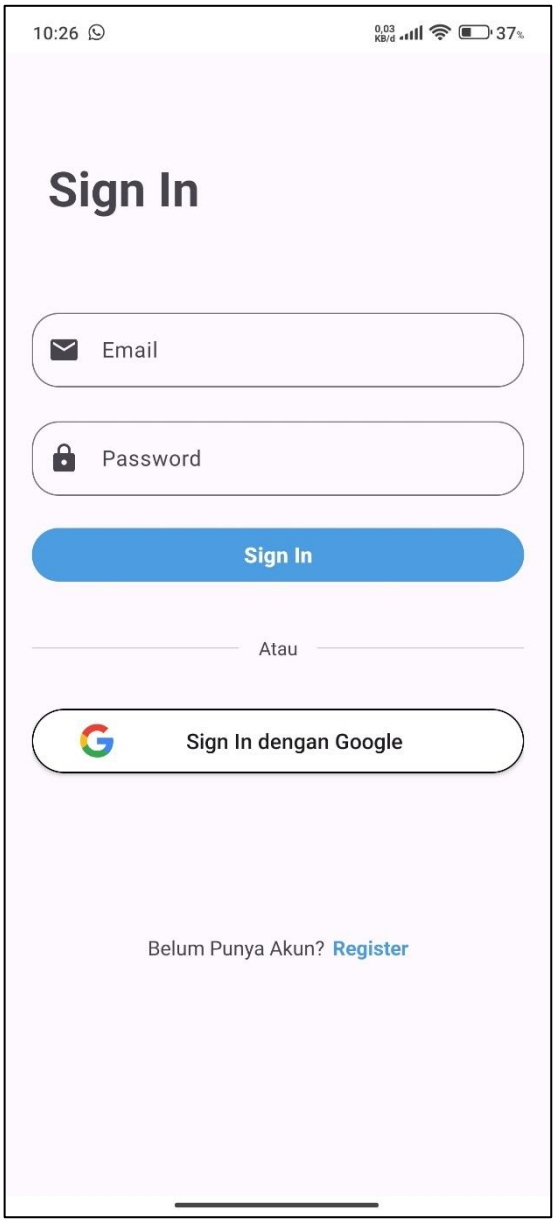
Gambar 4.5 Desain *Splash Screen Activity*

Gambar 4.5 merupakan tampilan *Splash screen activity* yang hanya berisi logo Collabolio. *Splash screen* berfungsi untuk memberikan pengalaman awal yang menarik kepada pengguna saat mereka membuka aplikasi. Selain itu juga, splash screen berfungsi untuk melakukan persiapan awal, seperti inisialisasi aplikasi, memuat data, atau melakukan tugas-tugas lain yang diperlukan sebelum aplikasi benar-benar dimuat. Hal tersebut akan memberikan pengalaman yang lebih responsif dan mengurangi kebingungan pengguna.



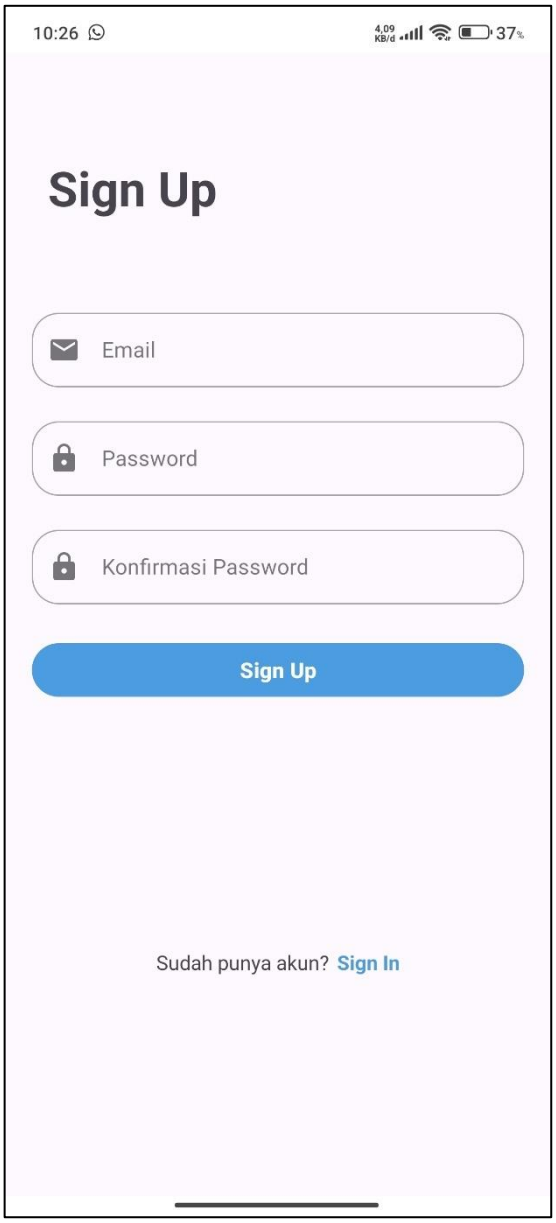
Gambar 4.6 Desain *Welcome Activity*

Gambar 4.6 Merupakan tampilan *Start/welcome activity* yang berfungsi sebagai tampilan pembuka dan sambutan kepada pengguna. Pada *start activity* ini di sematkan beberapa elemen seperti *text view* pembuka dan pengenalan aplikasi. Selain itu, terdapat juga gambar yang menarik dan sesuai dengan tema aplikasi. Pada *start screen* ini juga terdapat dua tombol yaitu tombol *sign in* dan *sign up.*



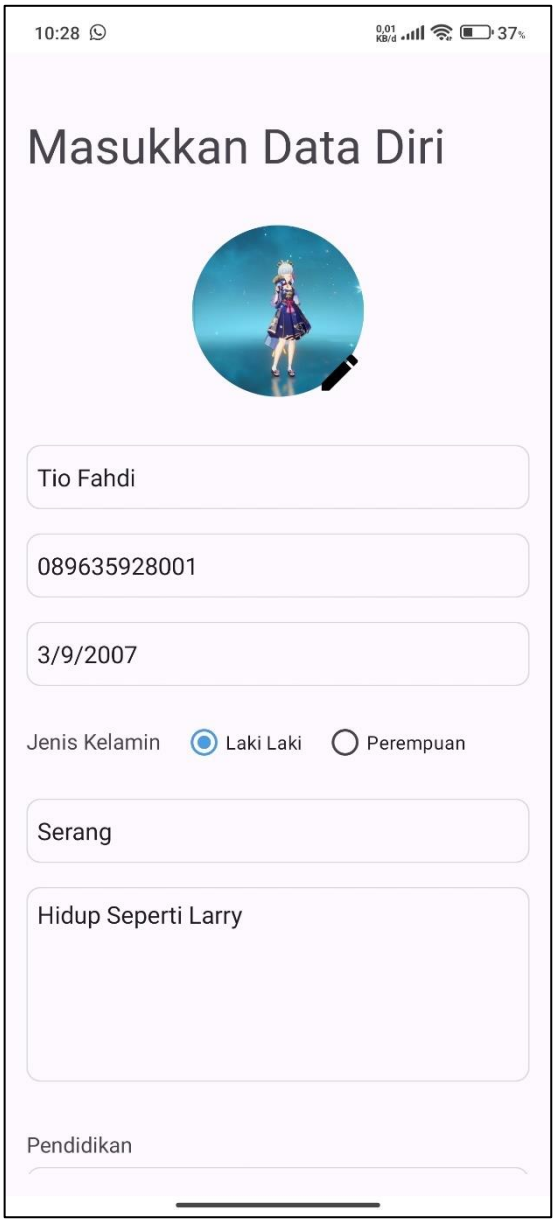
Gambar 4.7 Desain *Login Activity*

Gambar 4.7 merupakan tampilan *login activity* Dimana pengguna diminta untuk memasukkan alamat email dan *password* agar bisa masuk ke dalam menu utama aplikasi. Elemen yang digunakan pada tampilan ini yaitu *edit text view* untuk input email dan *password*, dan juga dua tombol *sign in*. Tombol *sign in* pertama digunakan jika memasukkan alamat email dan kata sandi secara manual. Dan tombol *sign in* kedua digunakan jika ingin masuk secara otomatis dengan memasukkan akun google. Selain itu, terdapat *clickable text* yaitu "belum punya akun? Sign Up".



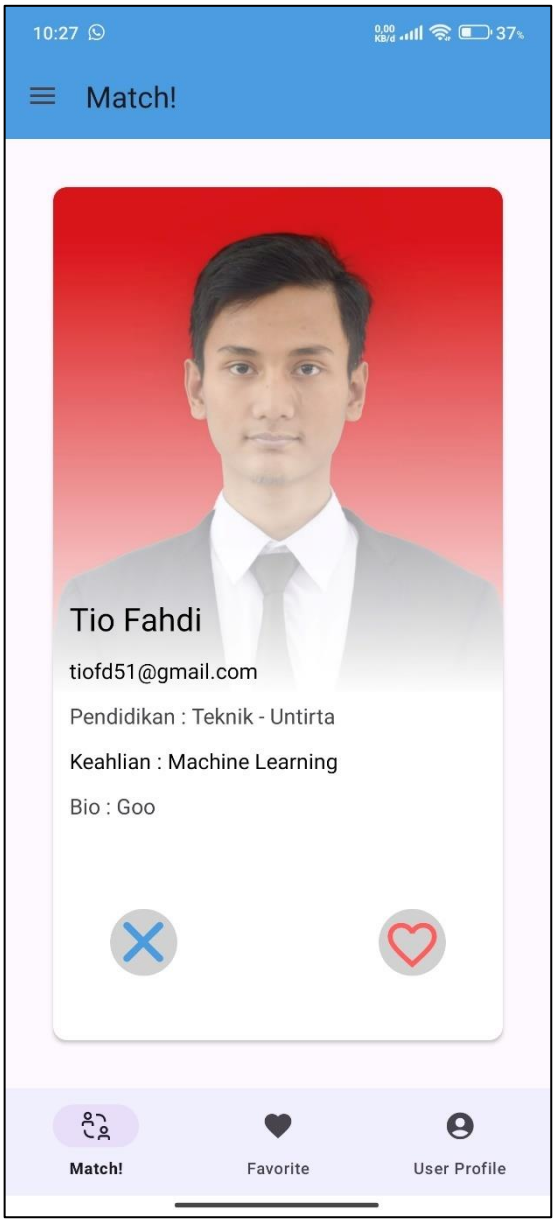
Gambar 4.8 Desain *Register Activity*

Gambar 4.8 merupakan tampilan *register activity* dimana pengguna diberi kesempatan untuk membuat akun baru dalam aplikasi. Pada tampilan ini, terdapat beberapa elemen yang sama seperti *login activity* namun dengan tambahan yaitu input untuk konfirmasi ulang *password.* Setelah pengguna berhasil membuat akun baru maka pengguna akan langsung diarahkan ke tampilan *InputBio Activity*.



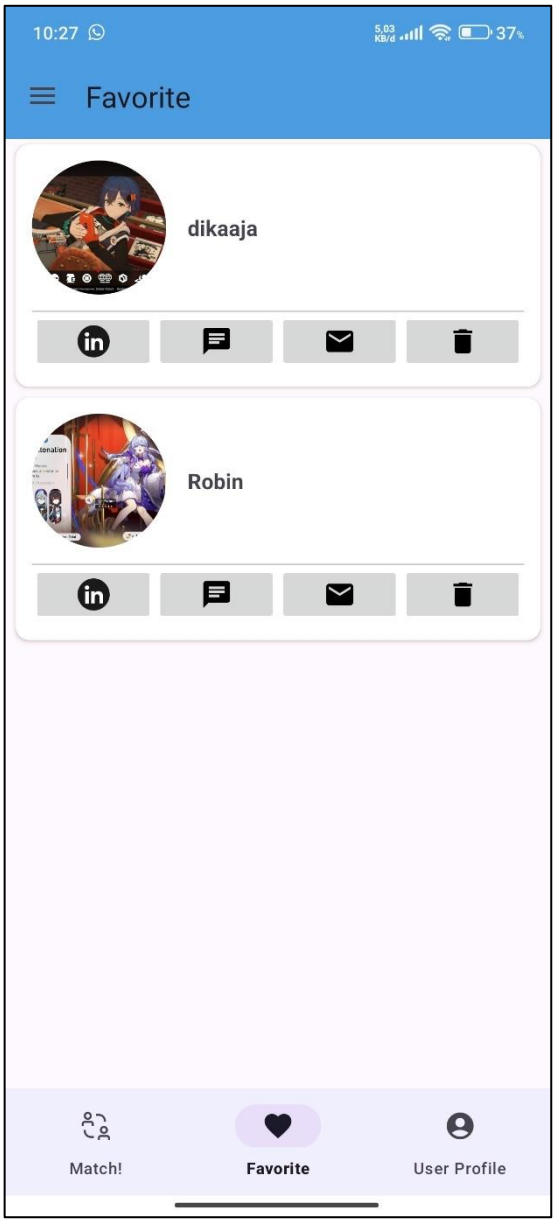
Gambar 4.9 Desain *InputBio Activity*

Gambar 4.9 merupakan tampilan *inputbio activity* yaitu tampilan untuk memasukkan biodata diri pengguna. Proses input pada tampilan ini menggunakan beberapa elemen input seperti *edit text view* untuk input nama, nomor telepon, alamat, pendidikan, dan deskripsi singkat. Selain itu terdapat juga elemen *radio button* untuk input jenis kelamin dan juga *dropdown* menu untuk pemilihan *skill* dan *interest*. Pengguna akan diminta untuk mengisi formulir secara lengkap dan sesuai dengan input yang tersedia.



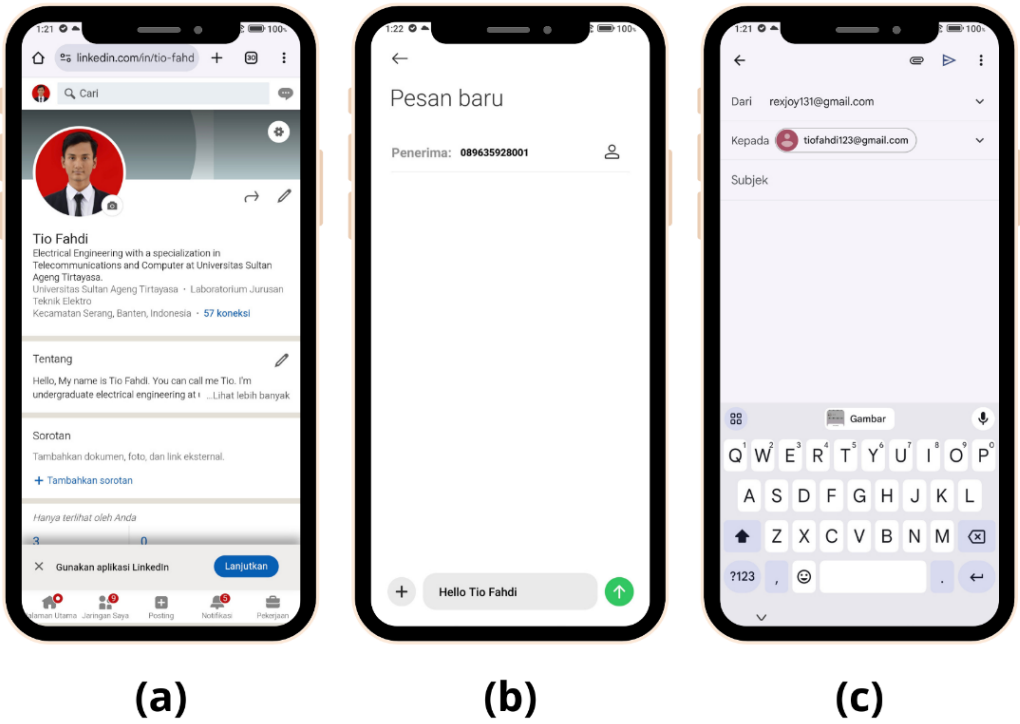
Gambar 4.10 Desain *Swipe Fragment*

Gambar 4.10 merupakan tampilan *tab swipe fragment* yang merupakan fitur utama aplikasi. Pada menu swipe ini mengimplementasikan elemen *recycler view* berupa *card stack view*. Setiap kartu mewakili profil pengguna dan memuat informasi yang relevan, seperti foto profil dan informasi singkat tentang pengguna tersebut. Pengguna dapat melihat profil pengguna lain dengan cara menggeser kartu ke kanan (jika tertarik) atau ke kiri (jika tidak tertarik). Hasil swipe kanan akan disimpan pada tab *favorite* sebagai pengguna yang disukai.



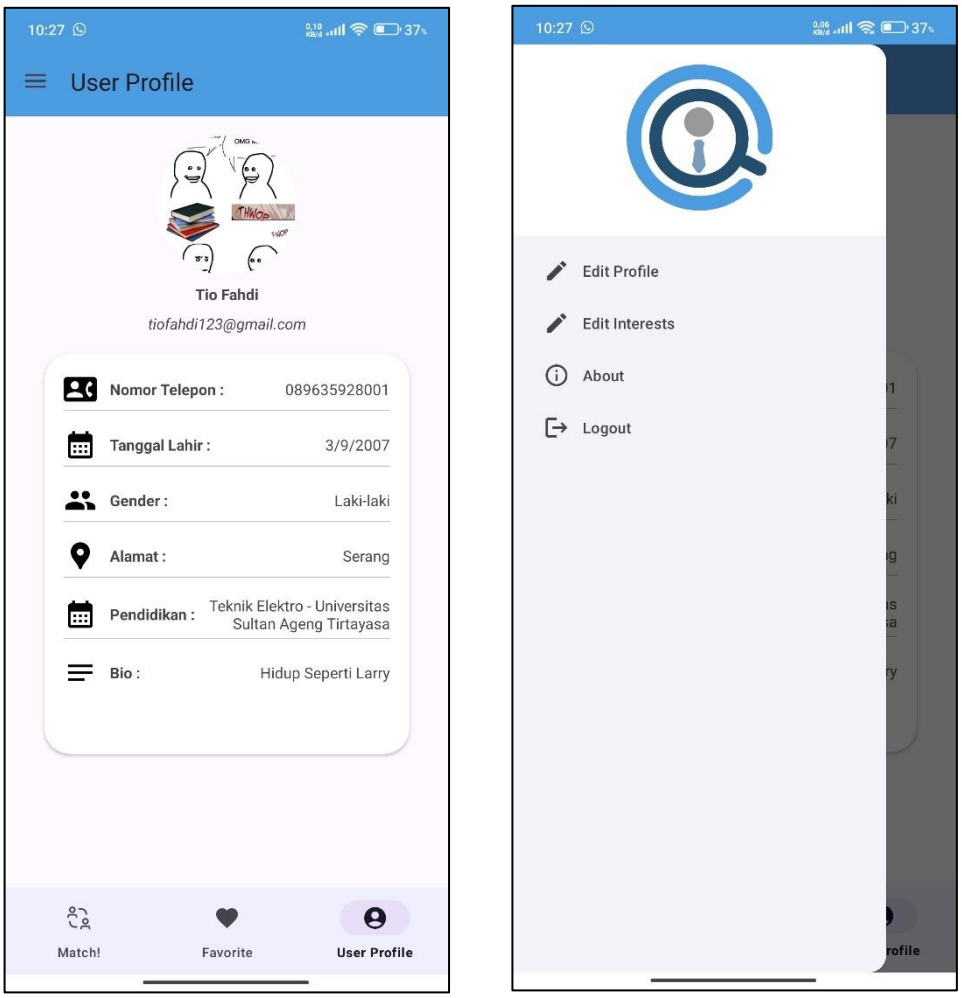
Gambar 4.11 Desain *Favorite Fragment*

Gambar 4.11 merupakan tab *favorite*, yang merupakan tampilan fragment untuk menyimpan hasil dari *swipe* kanan pada menu *swipe*. Elemen yang digunakan yaitu *Recycler view* untuk menampilkan daftar pengguna lain yang telah disukai. Di dalam *recycler view* ini, setiap item menampilkan informasi singkat tentang profil pengguna lain yang disukai, seperti foto profil, dan nama pengguna. Terdapat juga tombol-tombol aksi yang akan mengarah ke aplikasi lain seperti untuk membuka profile linked in pengguna yang disukai, mengirim pesan, mengirim email, dan terakhir yaitu menghapus pengguna dari daftar favorite.



Gambar 4.12 Tampilan ke Aplikasi Lain.

Pada gambar 4.12 bagian (a), ketika pengguna menekan tombol linkedin maka akan diarahkan ke web linkedin pengguna. Untuk gambar 4.12 bagian (b), ketika pengguna menekan tombol pesan maka pengguna akan diarahkan untuk mengirim pesan lewat whatsapp ataupun lewat aplikasi pesan seluler. Untuk gambar 4.12 bagian (c) ketika pengguna menekan tombol email maka pengguna akan diarahkan ke aplikasi gmail.



Gambar 4.13 Desain *Profile Fragment* dan Navigation Drawer

Gambar 4.12 merupakan tab user profile fragment yang menampilkan profil dari pengguna saat ini, yang memuat informasi seperti nama, nomor telepon, dan informasi lainnya. Kemudian pada navigation drawer, terdapat beberapa tombol yaitu tombol edit profil yang berfungsi untuk mengedit data profil, kemuudian tombol edit interest, tombol about, dan terakhir tombol *logout* yang memungkinkan pengguna untuk keluar dari sesi saat ini. Fitur logout ini penting karena memungkinkan pengguna untuk mengakhiri akses mereka ke aplikasi dan memastikan keamanan data serta privasi mereka

4.5 Perancangan Program Aplikasi

Pada proses ini, dibutuhkan pengetahuan mendalam tentang komponen-komponen yang terdapat pada android studio. Hal tersebut disebabkan karena pada

android studio terdapat banyak sekali komponen *class* dan *function* yang bisa digunakan. Setiap bagian dari kode *class* dan *function* dapat disesuaikan dengan kebutuhan aplikasi yang sedang dikembangkan sehingga aplikasi bisa berjalan dengan lancar dan tidak membebani kinerja perangkat.

4.5.1 Program pada tampilan *Splash Screen*

```
override fun onStart() {
        super.onStart()

        val currentUser = auth.currentUser

        Handler(Looper.getMainLooper()).postDelayed({
            if (currentUser != null) {
                goToNextActivity()
            } else {
                val intent = Intent(this,
WelcomeActivity::class.java)
                startActivity(intent)
                finish()
            }
        }, 3000)
    }

    private fun goToNextActivity() {
        val user = auth.currentUser
        val firestore = FirebaseFirestore.getInstance()
        user?.let { currentUser ->
            val userId = currentUser.uid
            val userRef =
firestore.collection("users").document(userId)
            userRef.get()
                .addOnSuccessListener { documentSnapshot ->
                    val phoneNumber =
documentSnapshot.getString("phoneNumber")
                    if (phoneNumber.isNullOrEmpty()) {
                        val intent = Intent(this,
InputBioActivity::class.java)
                        startActivity(intent)
                        finish()
                    } else {
                        val intent = Intent(this,
MainActivity::class.java)
                        startActivity(intent)
                        finish()
                    }
                    finish()
                }
                .addOnFailureListener { exception ->
                    Toast.makeText(this, "Error:
${exception.message}", Toast.LENGTH_SHORT).show()
                }
        }
    }
```

Pada kode program diatas, Ketika proses `onStart()` pada *Splash screen*, tampilan yang akan muncul setelahnya diatur dengan menggunakan *handler* dan pengecekan kondisi dengan percabangan *if else*. Jika pengguna baru menginstal aplikasi dan belum pernah *login* ke dalam aplikasi, maka akan diarahkan ke *welcome activity* yang kemudian pengguna akan diminta untuk melakukan *login* atau register. Selain itu, jika pengguna sudah pernah *login*, maka data *login* tersebut akan disimpan sehingga ketika pengguna masuk ke aplikasi akan langsung diarahkan ke menu utama yaitu menu *swipe*.

4.5.2 Authentikasi Login dan Register Menggunakan Firebase Auth

Dengan menggunakan Firebase Auth, pengguna dapat melakukan login ke aplikasi menggunakan alamat email dan *password*, atau melalui mekanisme autentikasi lainnya yang disediakan, seperti login menggunakan akun google atau akun facebook. Namun untuk implementasi ke aplikasi Collabolio ini hanya menggunakan dua cara saja yaitu login menggunakan email dan password atau login menggunakan akun Google. Untuk autentikasi menggunakan email dan *password* dapat dilihat pada program berikut.

```
private fun inputValidity(){
        binding.etLoginEmail.addTextChangedListener(object :
TextWatcher {
            override fun afterTextChanged(s: Editable?) {
                checkInputValidity()
            }

            override fun beforeTextChanged(s: CharSequence?,
start: Int, count: Int, after: Int) {}

            override fun onTextChanged(s: CharSequence?, start:
Int, before: Int, count: Int) {}
        })

        binding.etLoginPassword.addTextChangedListener(object :
TextWatcher {
            override fun afterTextChanged(s: Editable?) {
                checkInputValidity()
            }

            override fun beforeTextChanged(s: CharSequence?,
start: Int, count: Int, after: Int) {}

            override fun onTextChanged(s: CharSequence?, start:
Int, before: Int, count: Int) {}
        })
    }
```

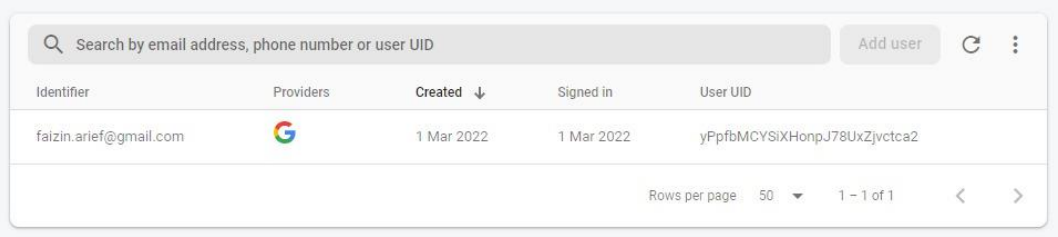
```
    private fun checkInputValidity() {
        val email = binding.etLoginEmail.text.toString()
        val password = binding.etLoginPassword.text.toString()
        val emailIsValid =
Patterns.EMAIL_ADDRESS.matcher(email).matches()
        val passwordIsValid = password.length >= 8
        binding.btnLogin.isEnabled = emailIsValid &&
passwordIsValid
    }

    private fun signIn(email: String, password: String) {
        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    goToNextActivity()
                } else {
                    Toast.makeText(this,
getString(R.string.error_login), Toast.LENGTH_SHORT).show()
                }
            }
    }
    private fun actionClick(){
        binding.toRegister.setOnClickListener {
            val intent = Intent(this,
RegisterActivity::class.java)
            startActivity(intent)
        }
        binding.btnGoogle.setOnClickListener {
            signInGoogle()
        }
        binding.btnLogin.setOnClickListener {
            val email = binding.etLoginEmail.text.toString()
            val password =
binding.etLoginPassword.text.toString()
            signIn(email, password)
        }
    }
```

Pengguna akan diminta untuk memasukkan alamat email dan password yang valid sesuai dengan input yang tersedia. Terdapat juga pengecekan input dari email dan *password* dengan syarat format email harus sesuai dan *password* harus berisi minimal 8 karakter. Jika syarat tidak terpenuhi maka tombol *sign in* tidak akan bisa ditekan.

Pada kode program juga, terdapat fungsi `signInGoogle()` dimana terdapat `ActivityResultLauncher` dengan parameter `signInIntent` untuk menampilkan pop up berisi pilihan akun Google. Pengguna akan diminta untuk memilih akun google yang tersedia pada perangkat. Apabila sudah dipilih, pengguna akan mendapatkan `idToken` dari `GoogleSignInAccount`. Untuk pengguna yang login menggunakan akun google ini, tidak perlu melakukan register

karena akunnya akan otomatis terdaftar ke firebase *auth*. Hasil dari *login* pengguna akan tercatat pada firebase auth dan dapat dilihat pada gambar berikut.



Gambar 4.14 Hasil Autentikasi Pada Firebase

Pada login activity juga terdapat proses pengecekan pada program yang dimana ketika biodata pengguna sudah tersedia pada *cloud firestore*, maka pengguna akan langsung diarahkan ke tampilan menu utama. Sedangkan ketika biodata pengguna belum tersedia, maka pengguna akan diarahkan ke *inputbio activity*.

Register activity digunakan jika pengguna ingin mendaftarkan akun baru ke aplikasi dengan menggunakan email dan *password*. untuk implementasinya mirip seperti *login activity* yaitu terdapat pengecekan untuk input dari email dan *password* dengan *tambahan input* berupa konfirmasi ulang *password*. Setelah pengguna menekan tombol *sign up* maka akan muncul *popup* permberitahuan "Registrasi Berhasil" dan pengguna akan langsung diarahkan ke *inputbio activity.*

4.5.3 Mengirim Dan Menyimpan Data Pengguna ke Firestore

Proses mengirim dan menyimpan data pengguna terjadi pada *inputbio activity* dengan menggunakan firebase firestore *cloud*. Berikut adalah kode programnya.

```
private fun startGallery() {

launcherGallery.launch(PickVisualMediaRequest(ActivityResultCont
racts.PickVisualMedia.ImageOnly))
    }

    private val launcherGallery = registerForActivityResult(
        ActivityResultContracts.PickVisualMedia()
    ) { uri: Uri? ->
        if (uri != null) {
            currentImageUri = uri
            showImage()
        } else {
            Log.d("Photo Picker", "No media selected")
```

```
        }
    }

    private fun showImage() {
        currentImageUri?.let {
            Log.d("Image URI", "showImage: $it")
            binding.ivProfile.setImageURI(it)
        }
    }

    private fun saveProfile() {
        progressBar.visibility = View.VISIBLE
        val email = firebaseAuth.currentUser?.email ?: ""
        val uid = firebaseAuth.currentUser?.uid ?: ""
        val displayName = binding.inputName.text.toString()
        val phoneNumber = binding.inputNumber.text.toString()
        val birthDate = binding.inputDate.text.toString()
        val male = binding.radioButtonMale.isChecked
        val location = binding.inputAddress.text.toString()
        val degree = binding.inputDegree.text.toString()
        val school = binding.inputSchool.text.toString()
        val bio = binding.inputBio.text.toString()
        var userSkills = binding.mactvSkills.text.toString()
        userSkills = userSkills.trimEnd(',', ' ')
        val arraySkills = userSkills.split(",").map{it.trim()}
        val education = Education(degree, school)
        val linkedinURL = binding.inputLinkedin.text.toString()

        val skillsList = mutableListOf<Skills>()
        for (skillName in arraySkills) {
            val skill = Skills(skillName)
            skillsList.add(skill)
        }

        val connectionList = mutableListOf<Connections>()

        val currentUser = firebaseAuth.currentUser
        currentUser?.let {
            val userId = it.uid
            val                    userRef                    =
firestore.collection("users").document(userId)
            when {
                displayName.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputName.error = "Masukkan Nama"
                    Toast.makeText(this,    "Masukkan    Nama",
Toast.LENGTH_SHORT).show()

                }
                phoneNumber.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputNumber.error = "Masukkan Nomor
Telepon"
                    Toast.makeText(this,    "Masukkan    Nomor
Telepon", Toast.LENGTH_SHORT).show()
                }
                birthDate.isEmpty() -> {
                    progressBar.visibility = View.GONE
```

```
                    binding.inputNumber.error     =     "Masukkan
Tanggal Lahir"
                    Toast.makeText(this,    "Masukkan    Tanggal
Lahir", Toast.LENGTH_SHORT).show()
                }

                userSkills.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.mactvSkills.error     =     "Masukkan
Keahlian"
                    Toast.makeText(this,   "Masukkan   Keahlian",
Toast.LENGTH_SHORT).show()
                }

                location.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputAddress.error    =    "Masukkan
Alamat"
                    Toast.makeText(this,    "Masukkan    Alamat",
Toast.LENGTH_SHORT).show()
                }
                degree.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputDegree.error = "Masukkan Gelar
dan Jurusan"
                    Toast.makeText(this,  "Masukkan  Gelar  dan
Jurusan", Toast.LENGTH_SHORT).show()
                }
                school.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputAddress.error    =    "Masukkan
Sekolah atau Universitas"
                    Toast.makeText(this, "Masukkan Sekolah atau
Universitas", Toast.LENGTH_SHORT).show()
                }
                bio.isEmpty() -> {
                    progressBar.visibility = View.GONE
                    binding.inputAddress.error = "Masukkan Bio"
                    Toast.makeText(this,     "Masukkan     Bio",
Toast.LENGTH_SHORT).show()
                }
                linkedinURL.isNotEmpty()                        &&
!linkedinURL.startsWith("http") -> {
                    progressBar.visibility = View.GONE
                    binding.inputLinkedin.error = "Masukkan link
URL dengan sesuai"
                    Toast.makeText(this,  "Link  tidak  valid",
Toast.LENGTH_SHORT).show()
                }
                currentImageUri == null -> {
                    progressBar.visibility = View.GONE
                    Toast.makeText(this, "Masukkan Foto Profil
Anda", Toast.LENGTH_SHORT).show()
                }
                else -> {
                    val               storageRef               =
storage.getReference("users").child(userId).child("profile.jpg")

storageRef.putFile(currentImageUri!!).addOnSuccessListener {
```

```
storageRef.downloadUrl.addOnSuccessListener { documents ->
                            val              users              =
UserProfileResponse(email,   uid,   displayName,   phoneNumber,
birthDate,   education,   male,   bio,   location,   photoURL   =
documents.toString(), skillsList, connectionList, linkedinURL =
linkedinURL)
                            userRef.set(users,
SetOptions.merge())
                                .addOnSuccessListener {
                                    progressBar.visibility     =
View.GONE
                                    Toast.makeText(this, "Profil
berhasil diperbarui", Toast.LENGTH_SHORT)
                                        .show()
                                    goToNextActivity()
                                }
                                .addOnFailureListener {
                                    Toast.makeText(this, "Profil
gagal diperbarui", Toast.LENGTH_SHORT)
                                        .show()
                                }
                        }.addOnFailureListener {
                            Toast.makeText(this,         "Terjadi
Masalah", Toast.LENGTH_SHORT).show()
                        }
                    }.addOnFailureListener{
                        Toast.makeText(this,  "Terjadi  Masalah",
Toast.LENGTH_SHORT).show()
                    }

                }
            }
        }
    }
    private fun fetchSkills() {
        val skillsCollection = firestore.collection("Skills")

        skillsCollection.get()
            .addOnSuccessListener { querySnapshot ->
                val skillsList = mutableListOf<String>()

                for (document in querySnapshot.documents) {
                    val skill = document.getString("name")
                    skill?.let { skillsList.add(it) }
                }

                val      adapter      =      ArrayAdapter(this,
R.layout.simple_dropdown_item_1line, skillsList)
                binding.mactvSkills.setAdapter(adapter)

binding.mactvSkills.setTokenizer(MultiAutoCompleteTextView.Comma
Tokenizer())
            }
            .addOnFailureListener {
            }
    }

    private fun goToNextActivity() {
```

```
        val user = firebaseAuth.currentUser
        val firestore = FirebaseFirestore.getInstance()
        user?.let { currentUser ->
            val userId = currentUser.uid
            val                    userRef                    =
firestore.collection("users").document(userId)
            userRef.get()
                .addOnSuccessListener { documentSnapshot ->
                    val               interests               =
documentSnapshot.getString("interests")
                    if (interests.isNullOrEmpty()) {
                        val      intent      =      Intent(this,
InterestActivity::class.java)
                        startActivity(intent)
                        finish()
                    } else {
                        val      intent      =      Intent(this,
MainActivity::class.java)
                        startActivity(intent)
                        finish()
                    }
                    finish()
                }
                .addOnFailureListener { exception ->
                    Toast.makeText(this,                  "Error:
${exception.message}", Toast.LENGTH_SHORT).show()
                }
        }
    }

    private fun showDatePicker(editText: EditText){
        val year = calendar.get(Calendar.YEAR)
        val month = calendar.get(Calendar.MONTH)
        val day = calendar.get(Calendar.DAY_OF_MONTH)

        datePickerDialog  =  DatePickerDialog(this,  {_,  year,
month, day ->
            val selectedDate = "$day/${month+1}/$year"
            editText.setText(selectedDate)
        }, year, month, day)
        datePickerDialog.show()
    }
```

Pada listing kode, pengguna akan diminta untuk memasukkan data diri pengguna sesuai dengan *input* yang tersedia seperti nama lengkap, nomor telepon, jenis kelamin, Pendidikan, keahlian (*skill*), dan minat keahlian rekan yang dicari (*interest*). Pada *inputbio activity* terdapat pengecekan agar tidak ada kolom *input* yang kosong atau *null* data. Ketika pengguna selesai mengisi data kemudian menekan tombol “lanjut” maka data pengguna akan dikirim dan disimpan pada firebase firestore *cloud* dan pengguna akan diarahkan ke tampilan menu utama yaitu menu *swipe*.

4.5.4 Implementasi *machine learning* pada aplikasi

Implementasi *machine learning* pada aplikasi Collabolio adalah langkah penting karena hal tersebut merupakan fitur utama dalam aplikasi. Dengan memanfaatkan teknologi *machine learning*, aplikasi dapat memberikan rekomendasi rekan kerja yang sesuai dengan keterampilan dan minat pengguna. Untuk program algoritma *machine learning* sendiri sudah disiapkan oleh tim pengembang *machine learning* dan disimpan ke dalam sebuah API. Proses implementasi pada aplikasi dilakukan dengan cara melakukan panggilan API *machine learning* tersebut untuk melakukan *training* model dan hasilnya akan di tampilkan pada aplikasi. Berikut adalah kode program pemanggilan API *machine learning* dengan menggunakan okhttp.

```
class ApiRequest : OkHttpClient() {

    private val firebaseAuth = FirebaseAuth.getInstance()

    fun getUsers(): Array<UserApi>? {
        val uid = firebaseAuth.currentUser?.uid

        if (uid == null) {
            return arrayOf()
        }

        val request = Request.Builder()
            .url("https://model-endpoint-onhqnm5xvq-
uc.a.run.app/api/users/$uid/30")
            .build()

        val response = newCall(request).execute()

        val user = response.body?.string()?.let {
Gson().fromJson(it, Array<UserApi>::class.java)}

        if (user != null) {
            return user
        } else {
            return arrayOf()
        }
    }
}
```

Pada listing kode, okhttp akan melakukan panggilan api dengan endpoint yang sudah ditentukan. Setelah itu, akan terjadi training model pada cloud yang menghasilkan sebuah *list* rekomendasi untuk pengguna. List tersebut akan

ditangkap oleh okhttp dan kemudian list data tersebut disimpan dan diolah sesuai dengan tampilan yang dibuat.

4.5.5 Menampilkan Hasil Rekomendasi

Setelah mendapatkan hasil *training model* pada *machine learning* berupa *list* UID (User ID), langkah berikutnya adalah menampilkan daftar tersebut pada swipe menu. Swipe menu ini menggunakan RecyclerView khusus yang telah disesuaikan dengan bentuk *CardStackView*. Pada tampilan *CardStackView* ini, pengguna dapat melakukan swipe ke kiri (tidak minat) atau swipe kanan (minat). Berikut untuk program pada swipe menu.

```
private suspend fun apiReqForUids(): MutableList<String> {
        return withContext(Dispatchers.IO) {
            val apiRequest = ApiRequest()
            val uidsResult = mutableListOf<String>()
            val users = apiRequest.getUsers()
            for (user in users!!) {
                uidsResult.add(user.uid.toString())
            }
            uidsResult
        }
    }

    private fun fillingTheItemsByApi() {
        CoroutineScope(Dispatchers.Main).launch {
            val uids = apiReqForUids()
            val cardsQuery =
db.collection("users").whereIn("uid", uids)
            try {
                val documents = withContext(Dispatchers.IO) {
cardsQuery.get().await() }
                for (document in documents) {
                    val user =
document.toObject(UserSwipe::class.java)
                    if (!rowItems.contains(user)) {
                        rowItems.add(user)
                        cardsAdapter?.notifyDataSetChanged()
                    }
                }
            } catch (exception: Exception) {
                Log.d("KARTU", "get failed with : ", exception)
            }
        }
    }
```

Pertama diambil list rekomendasi pengguna berupa UID menggunakan fungsi `apiReqForUids()`, kemudian hasil list UID tersebut di masukkan ke dalam fungsi `fillingTheItemsByApi()` untuk proses pencarian list UID

tersebut pada database di firestore. Setelah didapatkan list pengguna berdasarkan UID tersebut pada firestore, maka *list* data pengguna akan diambil dan kemudian ditampilkan pada tampilan swipe menu.

Setelah pengguna melakukan swipe ke kanan pada swipe menu, rekan kerja yang dipilih akan disimpan ke dalam *list favourite* menu. Implementasi penyimpanan ini juga menggunakan firestore. Setiap kali pengguna melakukan swipe ke kanan pada kartu rekomendasi, data rekan kerja tersebut akan disimpan ke list connection pengguna. Informasi yang disimpan berupa data UID. Untuk menampilkannya pada favorite menu, list uid dari connection tersebut diambil untuk dijadikan parameter filter pada firestore sehingga didapatkan profile pengguna yang di favoritkan. Berikut adalah kode program pada menu favorite.

```
fun fetchConnections() {
        if (currentUserUid != null) {
            firestore.collection("users")
                .document(currentUserUid)
                .get()
                .addOnSuccessListener { document ->
                    val userConnections =
document.toObject(UserProfileResponse::class.java)?.connections
?: listOf()
                    if (userConnections.isNotEmpty()) {
                        fetchConnectionProfiles(userConnections)
                    } else {
                        _connections.value = listOf()
                    }
                }
                .addOnFailureListener { exception ->
                    Log.d("FavoriteViewModel", "Error getting
user document: ", exception)
                }
        } else {
            Log.d("FavoriteViewModel", "Current user UID is
null")
        }
    }

    private fun fetchConnectionProfiles(connectionUids:
List<Connections>) {
        val profilesList = mutableListOf<UserProfileResponse>()
        firestore.collection("users")
            .whereIn("uid", connectionUids.map { it.uid })
            .get()
            .addOnSuccessListener { result ->
                for (document in result) {
                    val profile =
document.toObject(UserProfileResponse::class.java)
                    profilesList.add(profile)
                }
                _connections.value = profilesList
            }
```

```
            .addOnFailureListener { exception ->
                Log.d("FavoriteViewModel", "Error getting
connection profiles: ", exception)
            }
    }

    fun removeConnection(userProfile: UserProfileResponse) {
        if (currentUserUid != null) {
            firestore.collection("users")
                .document(currentUserUid)
                .update("connections",
FieldValue.arrayRemove(Connections(userProfile.uid)))
                .addOnSuccessListener {
                    fetchConnections()
                }
                .addOnFailureListener { exception ->
                    Log.d("FavoriteViewModel", "Error removing
connection: ", exception)
                }
        }
    }
```

Pengguna dapat melihat daftar rekan favorite dalam bentuk list *RecyclerView*. Selain itu, pengguna dapat melakukan tindakan seperti menghapus rekan kerja dari daftar favorit, mengirim pesan, ataupun mengirim email ke pengguna yang dipilih.

4.5.6 Menampilkan dan Mengedit User Profile

Pada *user profile* menu, aplikasi akan menampilkan informasi yang telah mereka berikan saat mendaftar, seperti nama, nomor telepon, alamat email dan lain-lain. Profile menu juga mengimplementasikan database firestore yang berfungsi untuk mengambil data pengguna dari database firestore. Berikut adalah kode program pada *user profile* menu.

```
private fun displayUserProfile(userProfile:
UserProfileResponse?) {
        if (userProfile != null) {
            binding.tvEmail.text = userProfile.email
            binding.tvName.text = userProfile.displayName
            binding.tvPhoneNumber.text = userProfile.phoneNumber
            binding.tvBirthDate.text = userProfile.birthDate
            binding.tvLocation.text = userProfile.location
            binding.tvBio.text = userProfile.bio
            binding.tvEducation.text =
"${userProfile.educations.degree} -
${userProfile.educations.school}"
            val genderText = if (userProfile.male) {
                "Laki-laki"
            } else {
                "Perempuan"
```

```
            }
            binding.tvIsMale.text = genderText
            Glide.with(requireContext())
                .load(userProfile.photoURL)
                .into(binding.userImageView)
        }
    }
```

Pada listing code, pertama dilakukan pengambilan data pengguna dari firestore, kemudian dilakukan proses pengecekan data null pada `userProfile` agar terhindar dari error *null pointer exception* yang menyebabkan aplikasi *crash* atau *force close*. Setelah itu, data akan dapat ditampilkan pada tampilan *user profile* menu. Selain menampilkan informasi profil, pengguna juga diberikan opsi untuk mengedit profil. Pada tombol edit profil, diberikan perintah yang akan melakukan `Intent` ke *inputbio activity*. Pengguna dapat mengubah atau menambahkan informasi baru sesuai kebutuhan. Dan terakhir terdapat opsi *logout* yang berfungsi untuk keluar dari sesi saat ini dan kembali ke tampilan login activity. Opsi *logout* berguna untuk keamanan agar data pengguna terjaga ketika pengguna sedang tidak menggunakan aplikasi.

4.5.7 Navigation Drawer

Untuk navigation drawer sendiri merupakan menu tambahan yang membantu dalam melakukan navigasi ke tampilan lain. Berikut adalah kodenya.

```
navViewDrawer.setNavigationItemSelectedListener { menuItem ->
            when (menuItem.itemId) {
                R.id.nav_edit_profile -> {
                    val       intent       =       Intent(this,
InputBioActivity::class.java)
                    startActivity(intent)
                    true
                }
                R.id.nav_edit_interests -> {
                    val       intent       =       Intent(this,
InterestActivity::class.java)
                    startActivity(intent)
                    true
                }
                R.id.nav_about -> {
                    val       intent       =       Intent(this,
AboutActivity::class.java)
                    startActivity(intent)
                    true
                }
                R.id.nav_logout -> {
                    signOut()
```

```
                    true
                }
                else -> false
            }
        }
```

Pada menu navigasi terdapat beberapa pilihan untuk pindah ke tampilan lain dengan menggunakan `intent` seperti edit profile, edit interest, tampilan about, dan melakukan logout.

4.6 Pengujian Aplikasi

Tahap terakhir dalam proses perancangan aplikasi adalah pengujian aplikasi. Pengujian dilakukan dengan metode *System Usability Scale* (SUS). Pengujian *usability* berfungsi untuk mengetahui tingkat kelayakan aplikasi yang dinilai dari segi efektivitas, efisiensi, dan kepuasan pengguna.

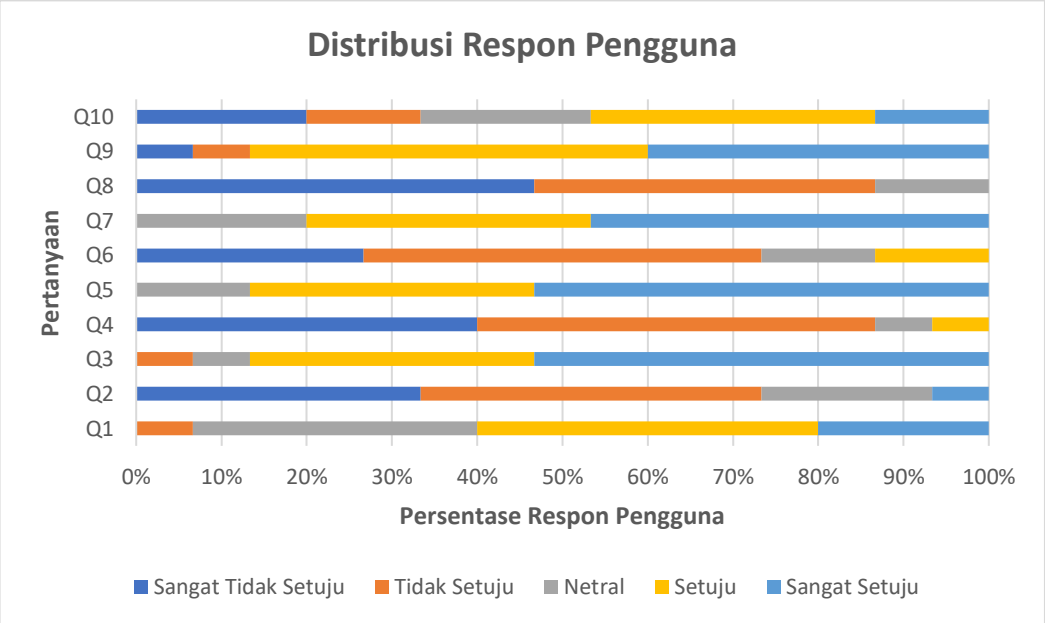
Pengujian melibatkan 15 responden mahasiswa yang berasal dari berbagai bidang keahlian. Responden terbagi atas 13 orang laki-laki (86,7%) dan 2 orang Perempuan (13,3%). Hasil Penilaian Responden tersebut dapat dilihat pada tabel berikut.

Tabel 4.2 Hasil Penilaian Responden

Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
R1	5	1	5	1	5	1	5	1	5	1
R2	4	2	5	1	4	3	4	2	4	3
R3	3	1	5	3	5	3	4	2	4	5
R4	4	2	5	2	5	1	5	1	5	2
R5	4	1	5	2	4	2	5	1	4	2
R6	5	3	4	2	5	2	3	3	5	3
R7	5	1	5	1	5	1	5	1	1	1
R8	4	3	3	2	4	2	3	2	4	5
R9	3	2	4	2	4	4	4	3	2	4
R10	2	2	4	2	3	4	4	2	4	3
R11	3	1	5	1	4	2	5	1	5	4
R12	4	2	4	4	5	2	5	2	4	4
R13	3	3	4	1	3	2	3	1	4	4
R14	4	5	5	1	5	1	5	1	5	1
R15	3	2	2	2	5	2	4	2	5	4

Keterangan:
R1-R15 = Jumlah Responden
Q1-Q10 = Jumlah Pertanyaan

Berdasarkan data tabel 4.3 diatas, dapat dibuat sebuah grafik data distribusi respon pengguna pada kuisioner SUS. Untuk grafik distribusi respon pengguna dapat dilihat pada gambar berikut.



Gambar 4.15 Distribusi Respon Pengguna

Setelah didapatkan data penilaian responden terhadap aplikasi, langkah selanjutnya adalah proses perhitungan data skor SUS. Hasil perhitungan data skor SUS dapat dilihat pada tabel berikut.

Tabel 4.3 Hasil Rata-rata Skor SUS

Responden	Jumlah	Skor SUS
R1	40	100
R2	30	75
R3	27	67.5
R4	36	90
R5	34	85
R6	29	72.5
R7	36	90
R8	24	60
R9	22	55
R10	24	60
R11	33	82.5
R12	28	70
R13	26	65
R14	35	87.5
R15	27	67.5

Rata-rata	**75.16666667**
Keterangan: **R1-R15 = Jumlah Responden**	

Berdasarkan perhitungan tabel diatas, didapatkan nilai rata-rata skor SUS sebesar 75.16666667. Jika dilakukan interpretasi skor SUS berdasarkan skala pada gambar 3.4, maka dapat diambil kesimpulan bahwa aplikasi termasuk ke dalam kategori *Good* (bagus) dalam skala *adjective* (peringkat sifat), sudah memenuhi kriteria *acceptable* dalam skala tingkat penerimaan, namun mendapat tingkat loyalitas pengguna passive dalam skala NPS (*Net Promotore Score*) yang berarti pengguna tidak dalam kondisi menolak maupun menyukai aplikasi Collabolio ini.