

Dokumentasi Project Mobile App Habit Tracker

NPM : 23312115

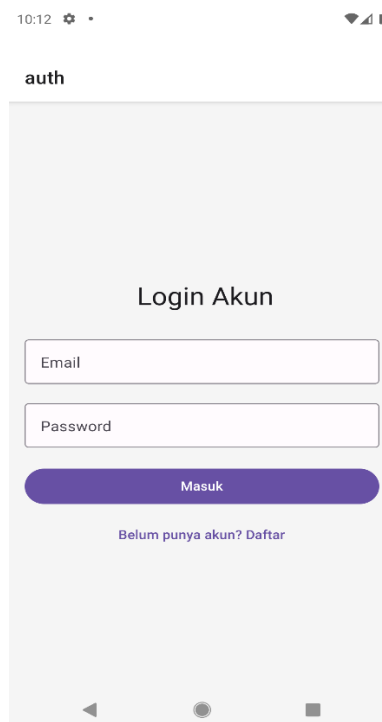
Nama : Rivaldo Azwir

Ringkasan Singkat

Habit Tracker merupakan aplikasi mobile yang dirancang untuk membantu pengguna membangun dan menjaga kebiasaan positif secara konsisten. Melalui aplikasi ini, pengguna dapat membuat daftar kebiasaan yang ingin dijalankan, baik bersifat harian, mingguan, maupun bulanan. Setiap kebiasaan dapat dilengkapi dengan judul dan deskripsi agar tujuan yang ingin dicapai lebih jelas. Antarmuka aplikasi dirancang sederhana dan mudah digunakan, sehingga pengguna dapat fokus pada aktivitas pembentukan kebiasaan tanpa hambatan teknis.

Selain menambahkan kebiasaan, Habit Tracker memungkinkan pengguna untuk melacak pelaksanaan kebiasaan tersebut setiap hari dengan menandai aktivitas yang telah selesai. Sistem secara otomatis mencatat progres pengguna dan menghitung streak atau rentetan hari keberhasilan sebagai indikator konsistensi. Seluruh data kebiasaan dan progres pengguna disimpan secara aman melalui API backend, sehingga informasi tetap sinkron dan dapat diakses dengan baik selama pengguna menggunakan aplikasi.

1. Halaman Login (Login Page / Auth – Login Screen)



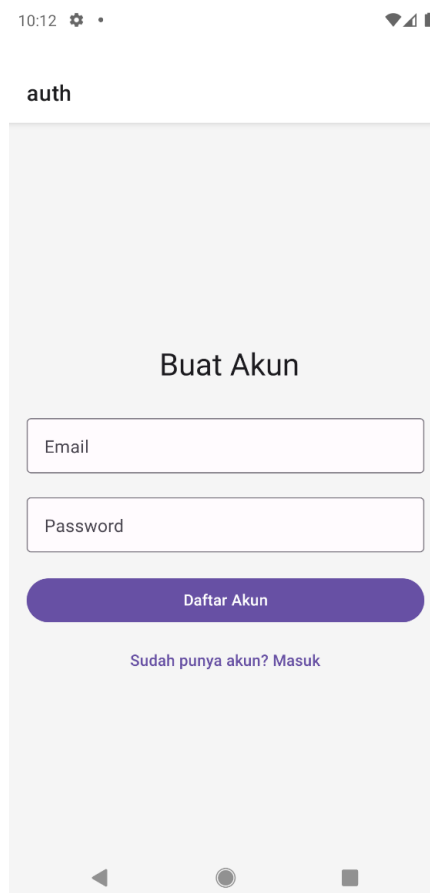
Sistem menampilkan Halaman Login sebagai bagian dari *AuthScreen*. Pada kondisi ini, nilai `isSignUp` diset **false**, sehingga judul halaman yang ditampilkan adalah “**Login Akun**” dan tombol utama bertuliskan “**Masuk**”.

- **Pengguna Mengisi Data Login**

Pengguna diminta untuk memasukkan **email** dan **password** pada field yang telah disediakan.

- **Menekan Tombol Masuk**
Setelah data diisi, pengguna menekan tombol **Masuk** untuk memulai proses autentikasi.
- **Proses Autentikasi oleh Sistem**
Sistem menjalankan fungsi `handleAuth()` yang bertugas sebagai fungsi utama autentikasi. Fungsi ini akan melakukan **validasi input** terlebih dahulu.
- **Pemanggilan Fungsi Login**
Jika input valid, `handleAuth()` akan memanggil fungsi `signIn(email, password)` dari `useAuth()` untuk melakukan proses login ke sistem.
- **Login Berhasil**
Apabila proses login berhasil, sistem akan menjalankan perintah `router.replace("/")` untuk **mengalihkan pengguna ke halaman utama aplikasi**.
- **Login Gagal**
Jika terjadi kesalahan saat proses login, sistem akan menyimpan pesan kesalahan ke dalam **state error** dan menampilkannya kepada pengguna.
- **Navigasi ke Halaman Pendaftaran**
Jika pengguna belum memiliki akun, mereka dapat menekan tombol **“Belum punya akun? Daftar”**. Tombol ini akan memanggil fungsi `handleSwitchMode()` untuk berpindah ke **mode pendaftaran (Sign Up)**.

2. Halaman Buat Akun (AuthScreen – Mode Register)

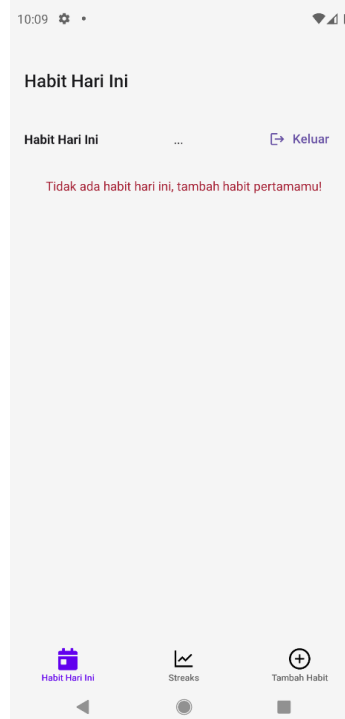


Halaman Buat Akun merupakan bagian dari **AuthScreen** yang digunakan untuk proses pendaftaran pengguna baru. Pada halaman ini, nilai `isSignUp` bernilai `true` sehingga

sistem menampilkan judul “**Buat Akun**” dan tombol “**Daftar Akun**”. Halaman ini berfungsi untuk menerima data email dan password pengguna yang kemudian diproses oleh fungsi `handleAuth()` sebagai pengendali utama proses registrasi.

Ketika tombol **Daftar Akun** ditekan, fungsi `handleAuth()` akan melakukan validasi input dan memanggil fungsi `signUp(email, password)` dari `useAuth()`. Jika terjadi kesalahan, pesan akan ditampilkan melalui state `error`. Apabila proses pendaftaran berhasil, akun pengguna akan dibuat dan halaman siap digunakan untuk proses login. Navigasi kembali ke halaman login disediakan melalui tombol “**Sudah punya akun? Masuk**” yang memanggil fungsi `handleSwitchMode()`.

3. Halaman Dashboard (Habit Hari Ini – Kondisi Kosong)



Halaman Dashboard merupakan halaman utama aplikasi yang ditampilkan setelah pengguna berhasil login. Halaman ini diimplementasikan pada komponen **Index** dan berfungsi untuk menampilkan daftar habit harian milik pengguna. Data habit diambil berdasarkan `user_id` melalui fungsi `fetchHabits()` dan disimpan ke dalam state `habits`. Pada kondisi ketika state `habits` bernilai kosong, sistem akan menampilkan tampilan *empty state* berupa pesan “**Tidak ada habit hari ini, tambah habit pertamamu!**” sebagai indikasi bahwa pengguna belum memiliki habit yang terdaftar untuk hari ini.

Pada halaman ini juga dijalankan proses sinkronisasi data secara real-time menggunakan `client.subscribe`, sehingga setiap perubahan data habit atau penyelesaian habit akan langsung memicu pemanggilan ulang fungsi `fetchHabits()` dan `fetchTodayCompletion()`. Selain itu, halaman Dashboard menyediakan tombol **Keluar** yang terhubung dengan fungsi `signOut()` dari `useAuth()` untuk mengakhiri sesi pengguna dan keluar dari aplikasi.

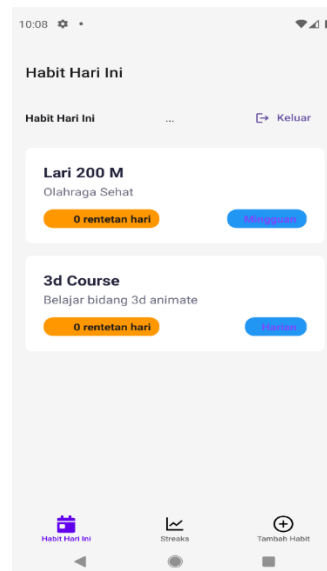
4. Halaman Tambah Habit (Tambah Habit Screen)

Halaman Tambah Habit digunakan untuk menambahkan data habit baru milik pengguna ke dalam sistem. Pada halaman ini, pengguna mengisi judul habit dan deskripsi yang masing-masing disimpan ke dalam state `title` dan `description`. Selain itu, pengguna dapat memilih frekuensi habit melalui opsi **Harian**, **Mingguan**, atau **Bulanan**.

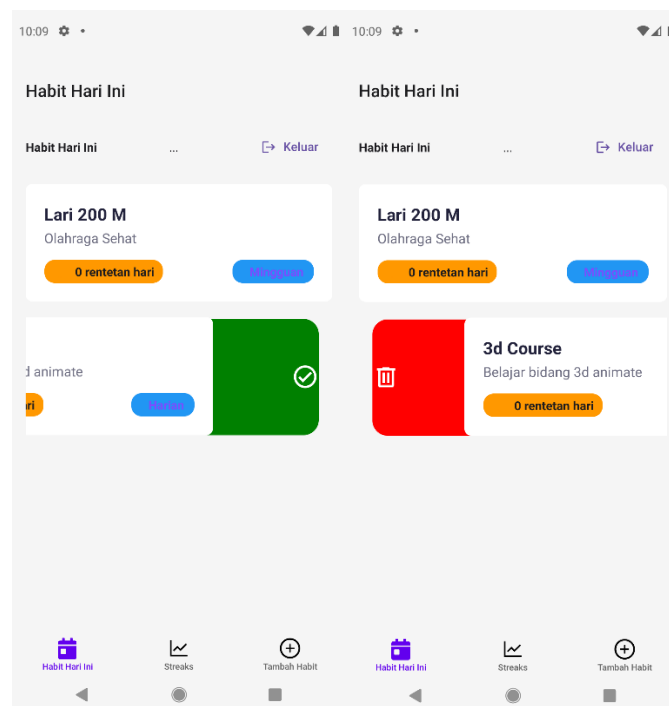
Berikut adalah langkah-langkah mekanisme ketika program berjalan:

- **Menekan Tombol Tambah Habit**
Pengguna menekan tombol **Tambah Habit**, lalu sistem menjalankan fungsi `handleSubmit()`.
- **Menyimpan Data Habit**
Fungsi `handleSubmit()` akan memproses dan menyimpan data habit baru yang terhubung dengan akun pengguna.
- **Berhasil Menambahkan Habit**
Jika penyimpanan berhasil, sistem akan menjalankan perintah `router.back()` untuk membawa pengguna kembali ke **halaman Dashboard**.
- **Terjadi Kesalahan**
Jika terjadi masalah saat proses penyimpanan, fungsi `handleSubmit()` akan menampilkan **pesan kesalahan** kepada pengguna.

5. Halaman Dashboard



Halaman Dashboard menampilkan daftar habit harian pengguna yang telah tersimpan di dalam sistem. Data habit diambil melalui fungsi `fetchHabits()` yang memanggil `databases.listDocuments` berdasarkan `user_id`, kemudian disimpan ke dalam state `habits`. Setiap habit ditampilkan dalam bentuk kartu yang memuat judul, deskripsi, jumlah *streak* (`streak_count`), serta frekuensi habit. Sistem juga memanggil fungsi `fetchTodayCompletion()` untuk mengetahui apakah suatu habit telah diselesaikan pada hari tersebut, yang kemudian digunakan oleh fungsi `isHabitCompleted()` untuk menandai status habit.

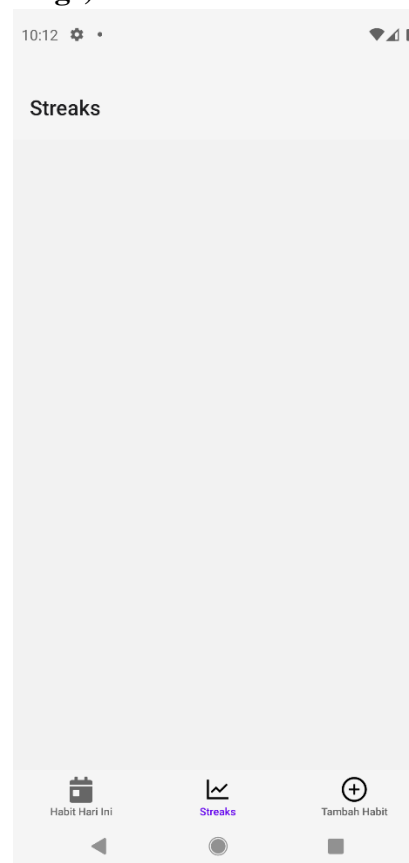


Berikut langkah-langkah yang akan terjadi Ketika user berinteraksi jadwal pada halaman dashboard:

- **Melakukan Gesture Swipe pada Habit**
Pengguna dapat berinteraksi dengan daftar habit menggunakan gesture **swipe** (geser).
- **Swipe ke Kiri akan Menyelesaikan Habit**
Jika pengguna melakukan **swipe ke kiri**, sistem akan menjalankan fungsi `handleCompleteHabit()` untuk menandai habit sebagai **selesai** dan memperbarui **streak**.
- **Swipe ke Kanan akan Menghapus Habit**
Jika pengguna melakukan **swipe ke kanan**, sistem akan menjalankan fungsi `handleDeleteHabit()` untuk **menghapus habit** dari sistem.
- **Pembaruan Data Secara Otomatis**
Setiap perubahan data akan langsung diperbarui secara **real-time** melalui mekanisme `client.subscribe`, sehingga daftar habit selalu menampilkan data terbaru.

Selain itu, halaman ini juga menyediakan tombol **Keluar** yang menjalankan fungsi `signOut()` untuk mengakhiri sesi pengguna.

6. Halaman Streak (Streaks Page)



Halaman Streak digunakan untuk menampilkan konsistensi pengguna dalam menjalankan habit yang telah dibuat. Pada halaman ini, sistem menampilkan informasi streak atau jumlah rentetan hari penyelesaian habit berdasarkan data yang tersimpan di dalam sistem. Data streak diperoleh dari nilai `streak_count` pada setiap habit yang diperbarui ketika pengguna menyelesaikan habit melalui fungsi

`handleCompleteHabit()`. Halaman ini membantu pengguna memantau progres dan menjaga motivasi dalam menjalankan kebiasaan secara berkelanjutan.

Halaman Streak bersifat informatif dan hanya menampilkan data hasil perhitungan dari aktivitas pengguna pada halaman Dashboard. Setiap perubahan penyelesaian habit akan berdampak langsung pada nilai streak, sehingga informasi yang ditampilkan selalu relevan dengan kondisi terbaru pengguna.