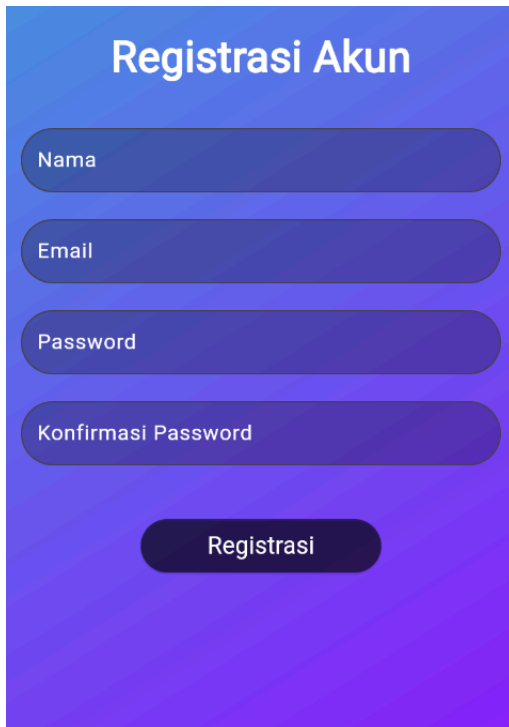


Nama : Yacobus Daeli

NIM : H1D022024

Responsi 1

1. Proses Registrasi

The image shows a mobile application interface for account registration. The background is a gradient of blue and purple. At the top, the title "Registrasi Akun" is displayed in white. Below the title, there are four rounded rectangular input fields, each with a placeholder text: "Nama", "Email", "Password", and "Konfirmasi Password". At the bottom of the form, there is a dark blue button with the text "Registrasi" in white.

Pengguna mengisi formulir pendaftaran dengan nama, email, dan kata sandi. Tiga objek `TextEditingController` digunakan untuk mengelola masukan pengguna: `_namaTextboxController` untuk nama, `_emailTextboxController` untuk email, dan `_passwordTextboxController` untuk kata sandi. Setiap masukan dilengkapi dengan validator untuk memastikan bahwa data yang dimasukkan memenuhi kriteria: Nama harus minimal 3 karakter, email harus diisi dan valid, kata sandi minimal 6 karakter, dan konfirmasi kata sandi harus sama dengan kata sandi yang telah dimasukkan. Ketika tombol "Registrasi" ditekan, fungsi `_submit()` dipanggil setelah validasi formulir berhasil. Jika validasi berhasil dan tombol tidak dalam keadaan loading, proses registrasi akan dimulai.

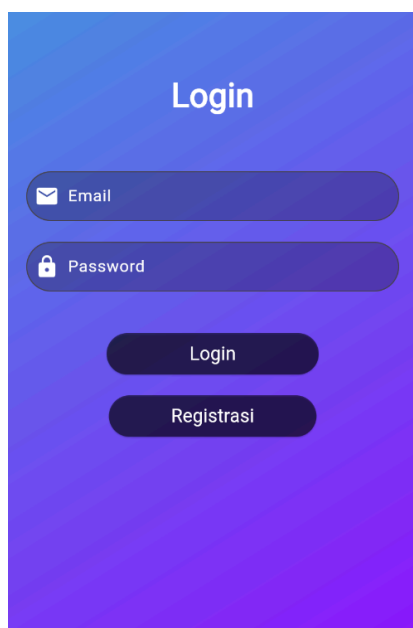
Dalam proses validasi dan manajemen status, langkah pertama adalah memanggil `save()` pada `_formKey.currentState` untuk menyimpan data input. Selanjutnya, status `_isLoading` diubah menjadi `true` untuk menampilkan indikator loading. Kemudian, fungsi `RegistrasiBloc.registrasi` dipanggil untuk mengirimkan permintaan pendaftaran ke API dengan data yang telah dikumpulkan dari controller, yaitu nama, email, dan kata sandi.

Setelah menerima respons dari API, jika pendaftaran berhasil, kita akan menampilkan dialog sukses menggunakan `SuccessDialog` yang menginformasikan pengguna bahwa pendaftaran telah berhasil dan mereka dapat login. Sebaliknya, jika pendaftaran gagal, kita akan menampilkan dialog peringatan menggunakan `WarningDialog` yang memberi tahu pengguna tentang kegagalan pendaftaran. Terakhir, status diatur ulang dengan mengubah `_isLoading` kembali menjadi `false` setelah proses selesai, baik berhasil maupun gagal.

`RegistrasiBloc` mengelola logika bisnis untuk pendaftaran dengan fungsi registrasi yang menerima parameter nama, email, dan kata sandi. Dalam proses ini, `RegistrasiBloc` membuat `apiUrl` dari `ApiUrl.registrasi` untuk menentukan endpoint API yang sesuai. Kemudian, ia menyusun body permintaan yang berisi data pengguna. Permintaan POST selanjutnya dikirimkan ke API menggunakan metode `Api().post()`. Setelah menerima respons, `RegistrasiBloc` mengonversi body respons dalam format JSON menjadi objek `Registrasi` dengan menggunakan metode `fromJson`.

Model `Registrasi` mendefinisikan struktur data yang akan diterima setelah proses pendaftaran. Model ini memiliki beberapa atribut, yaitu `code` yang menyimpan kode respons dari server, `status` yang mencatat status proses pendaftaran (apakah berhasil atau gagal), dan data yang menyimpan informasi tambahan yang mungkin diberikan oleh server. Selain itu, model ini juga dilengkapi dengan metode `fromJson` yang berfungsi untuk mengonversi data JSON menjadi objek `Registrasi`.

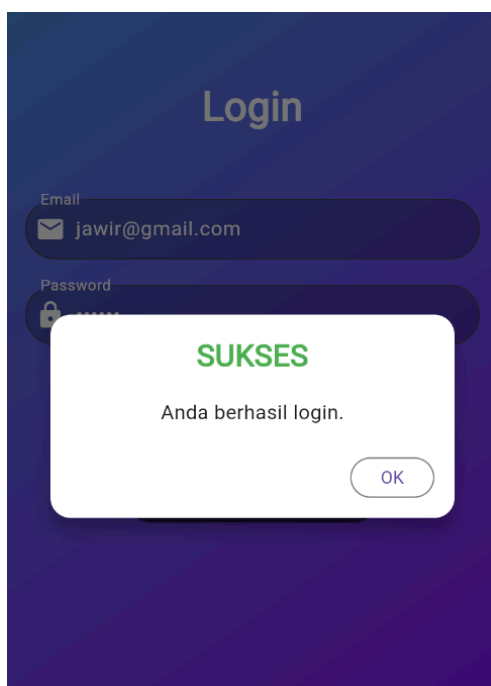
2. Proses Login



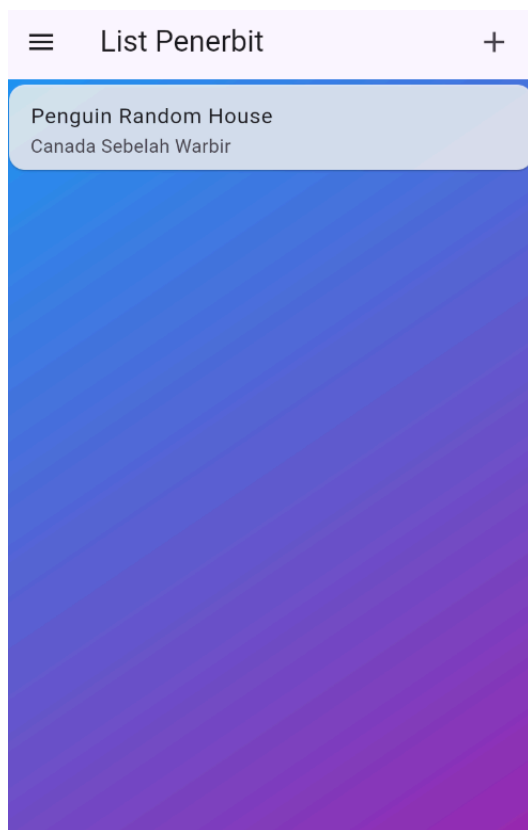
Pengguna menginput email dan kata sandi pada formulir yang tersedia. Input ini dikelola dengan `TextEditingController`, di mana `_emailTextboxController` digunakan untuk email dan `_passwordTextboxController` untuk kata sandi. Validasi dilakukan untuk memastikan bahwa kedua field tidak kosong.

Saat tombol "Login" diklik, aplikasi akan melakukan validasi pada formulir. Jika formulir valid, fungsi `_submit()` akan dipanggil. Dalam proses pengiriman, status `_isLoading` diatur menjadi `true` untuk menampilkan indikator loading, kemudian memanggil `LoginBloc.login` dengan email dan kata sandi yang dimasukkan oleh pengguna. Jika respons `code` dari API adalah 200, berarti login berhasil; dalam hal ini, token dan ID pengguna akan disimpan menggunakan kelas `UserInfo`, dan dialog sukses akan ditampilkan. Pengguna kemudian akan diarahkan ke halaman `PenerbitPage`. Namun, jika login gagal (respons tidak 200 atau terjadi kesalahan), dialog peringatan akan muncul dengan pesan bahwa login tidak berhasil.

Jika login berhasil (dalam hal ini, jika `value.code` sama dengan 200), kita menyimpan token dan ID pengguna menggunakan kelas `UserInfo`. Setelah menyimpan informasi tersebut, dialog sukses akan ditampilkan dengan menggunakan `SuccessDialog`, yang menginformasikan pengguna bahwa login telah berhasil. Jika pengguna menekan tombol "OK" pada dialog sukses, mereka akan diarahkan ke halaman `PenerbitPage`.



3. Create Data



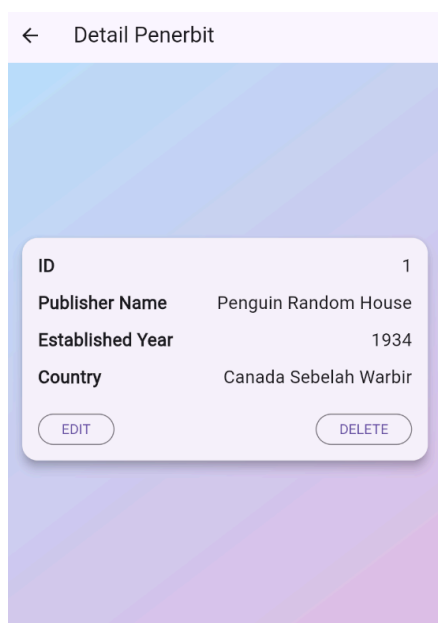
Di halaman **Penerbit_Page**, pengguna dapat melihat daftar penerbit. Ketika pengguna mengklik ikon tambah di sudut kanan atas halaman, mereka akan diarahkan ke halaman untuk menambahkan data.

A mobile application screenshot of the 'Tambah Penerbit' form. The header bar is light purple with a back arrow icon on the left and the text 'Tambah Penerbit' in the center. The main content area has a blue-to-purple gradient background. In the center, there is a white rounded rectangle containing the title 'Tambah Penerbit' in bold. Below the title are four text input fields stacked vertically: 'ID (Otomatis)', 'Publisher Name', 'Established Year', and 'Country'. At the bottom of this white box is a blue rounded button with the text 'Tambah' in white.

Pengguna mengisi formulir di `PenerbitForm`. Dalam `PenerbitForm`, terdapat tiga input teks yang digunakan untuk mengambil data dari pengguna, yaitu `originalLanguage` untuk memasukkan nama penerbit asli, `translatedLanguage` untuk memasukkan nama penerbit terjemahan, dan `translatorName` untuk memasukkan nama penerjemah. Setiap field menggunakan `TextEditingController` untuk menangani masukan dari pengguna.

Sebelum menyimpan data, formulir melakukan validasi untuk memastikan bahwa semua field terisi dengan benar. Jika ada field yang kosong, pengguna akan menerima pemberitahuan dengan pesan yang sesuai.

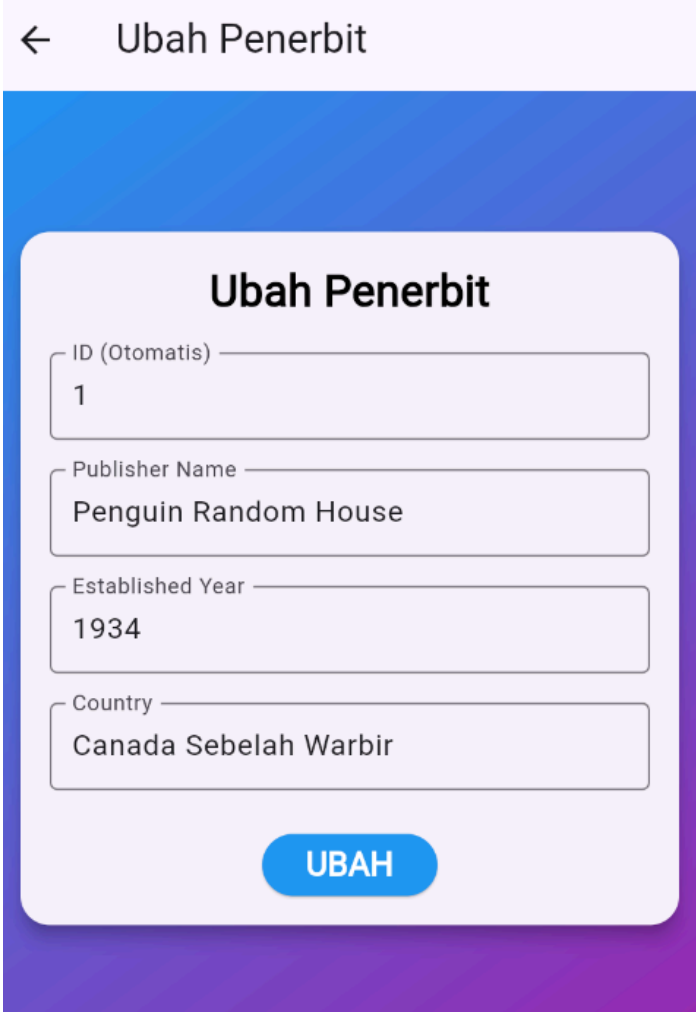
Ketika pengguna menekan tombol "Simpan", fungsi `simpan()` akan dipanggil. Di dalam fungsi ini, langkah pertama adalah melakukan validasi formulir. Jika validasi berhasil, data diambil dari `TextEditingController` dan dimasukkan ke dalam model `Penerbit`. Selanjutnya, fungsi `addPenerbit()` dari `PenerbitBloc` dipanggil dengan objek `Penerbit` yang baru dibuat sebagai parameter. Dalam `addPenerbit()` pada `PenerbitBloc`, data kemudian dikirim ke server melalui API menggunakan permintaan HTTP POST. Jika permintaan berhasil, API akan mengembalikan respons yang menunjukkan bahwa data telah berhasil ditambahkan.



Setelah data berhasil disimpan, pengguna dapat dinavigasikan kembali ke halaman **PenerbitPage**, yang akan memperbarui daftar penerbit dengan data terbaru.

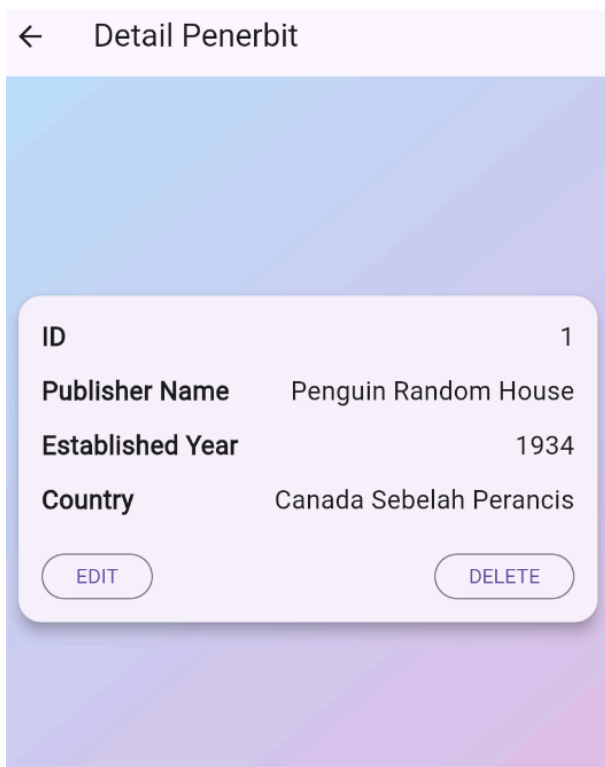
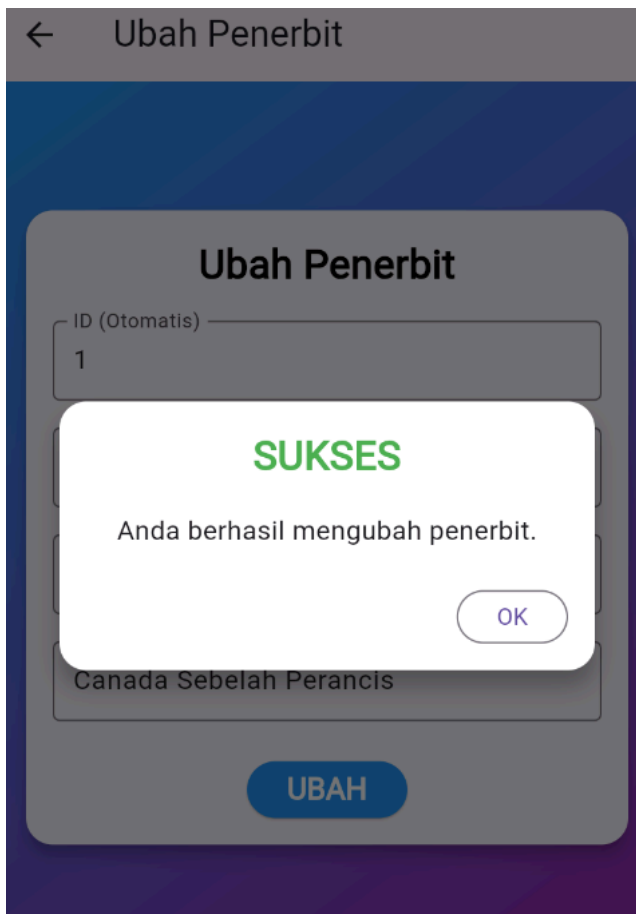
4. Edit Data

Saat pengguna memilih penerbit yang ditampilkan di halaman **PenerbitPage**, aplikasi akan mengarahkan mereka ke halaman `penerbit_detail`.



The screenshot shows a mobile application interface for editing a publisher. At the top, there is a header bar with a back arrow and the title "Ubah Penerbit". Below this is a form with a light purple background and rounded corners. The form has a title "Ubah Penerbit" and four input fields, each with a label and a value: "ID (Otomatis)" with "1", "Publisher Name" with "Penguin Random House", "Established Year" with "1934", and "Country" with "Canada Sebelah Warbir". At the bottom of the form is a blue button with the text "UBAH".

Ketika pengguna mengklik tombol "edit," data yang relevan akan diambil dan dimuat ke dalam field input di **PenerbitForm** menggunakan **TextEditingController**. Field yang telah diisi sebelumnya adalah **originalLanguage**, **translatedLanguage**, dan **translatorName**. Di halaman edit data, pengguna dapat mengubah informasi seperti bahasa asli, bahasa terjemahan, serta nama penerjemah.



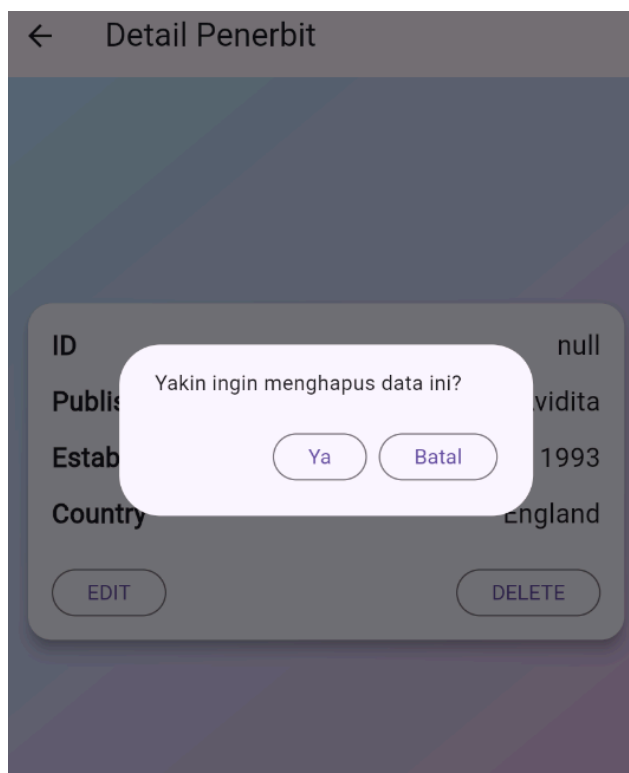
Diatas adalah tampilan setelah user edit data. Ketika pengguna melakukan perubahan dan menekan tombol "Ubah," fungsi `simpan()` akan dipanggil. Dalam fungsi ini, validasi

formulir dilakukan untuk memastikan bahwa semua input valid. Jika validasi berhasil, data terbaru diambil dari `TextEditingController` dan dimasukkan ke dalam objek `Penerbit` yang ada.

Fungsi `updatePenerbit()` dari `PenerbitBloc` kemudian dipanggil dengan objek `Penerbit` yang telah diperbarui sebagai parameter. Proses pengiriman data ke API dilakukan di dalam fungsi `updatePenerbit()` pada `PenerbitBloc`, di mana data dikirim menggunakan permintaan HTTP PUT. Jika permintaan berhasil, API akan mengembalikan respons yang menunjukkan bahwa data telah berhasil diperbarui.

Setelah data berhasil diperbarui, pengguna akan diarahkan kembali ke halaman `PenerbitPage`, di mana daftar penerbit akan diperbarui untuk mencerminkan perubahan yang baru saja dilakukan.

5. Delete Data

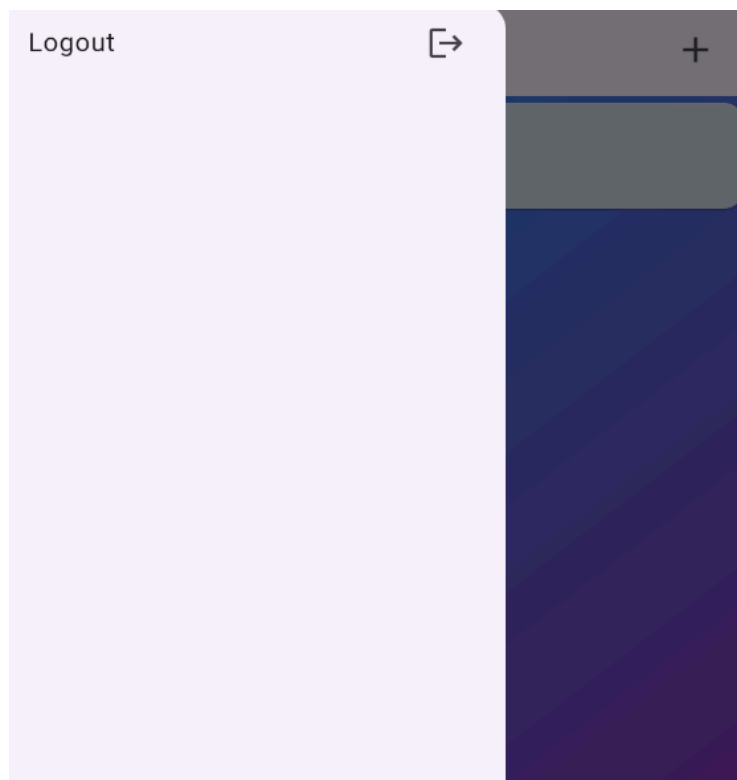


Setelah pengguna mengklik tombol "hapus," aplikasi akan meminta konfirmasi untuk memastikan bahwa pengguna benar-benar ingin menghapus data tersebut. Konfirmasi ini bertujuan untuk mencegah penghapusan data secara tidak sengaja.

Setelah pengguna memberikan konfirmasi, fungsi `deletePenerbit()` pada `PenerbitBloc` akan dipanggil. Fungsi ini bertanggung jawab untuk menghapus data penerbit yang telah dipilih oleh pengguna. Di dalam fungsi `deletePenerbit()`, permintaan HTTP DELETE akan dikirim ke server menggunakan API, yang mencakup ID penerbit yang ingin dihapus. Server kemudian memproses permintaan dan menghapus data penerbit dari basis data. Jika permintaan penghapusan berhasil, API akan mengembalikan respons yang menyatakan bahwa data penerbit telah berhasil dihapus.

Setelah data penerbit berhasil dihapus, pengguna akan diarahkan kembali ke halaman `PenerbitPage`, di mana daftar penerbit akan diperbarui secara otomatis untuk menghapus penerbit yang baru saja dihapus. Tidak ada notifikasi yang ditampilkan setelah proses penghapusan berhasil.

6). Logout



Ketika pengguna mengklik tombol "Logout," aplikasi akan memulai proses pengeluaran pengguna dari akun mereka.

Di dalam fungsi `logout()`, aplikasi akan menghapus token otentikasi dan informasi pengguna yang tersimpan di dalam penyimpanan lokal, seperti menggunakan

`shared_preferences`. Dengan cara ini, semua data yang memungkinkan pengguna untuk tetap terhubung akan dihapus.

Setelah proses penghapusan data selesai, aplikasi akan mengarahkan pengguna kembali ke halaman login. Hal ini memastikan bahwa pengguna harus memasukkan kredensial mereka lagi untuk mengakses akun. Di halaman login, aplikasi dapat memberikan pesan yang mengonfirmasi bahwa pengguna telah berhasil keluar dari akun mereka.

Dengan cara ini, fitur logout tidak hanya mengakhiri sesi pengguna, tetapi juga melindungi data pengguna dengan memastikan bahwa informasi sensitif tidak tetap tersedia setelah pengguna keluar.