

Nama : Yacobus Daeli
NIM : H1D022024
Shift Baru : E
Shift KRS : A

SOAL

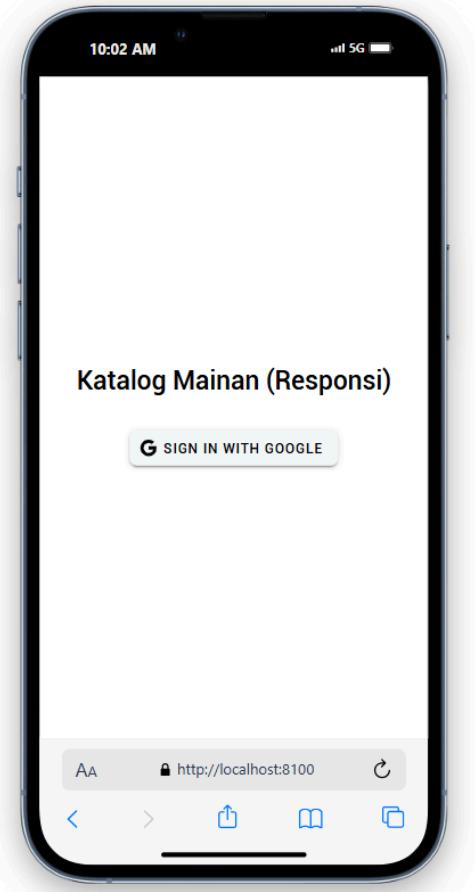
1. Jelaskan alasan mengapa Ionic framework tidak sepopuler framework mobile lainnya!
2. Sebutkan fungsi dari komponen ion refresher dalam Ionic ?
3. Bagaimana cara mengatur agar pengguna yang sudah login tidak bisa kembali ke halaman login dan pengguna yang belum login tidak bisa masuk ke dalam halaman home atau dashboard?

JAWABAN

1. Menurut saya ada beberapa hal yang memang membuat Ionic Framework tidak sepopuler framework lainnya, diantaranya :
 - Kinerja yang tidak sesuai dengan native, Ionic memakai metode hybrid yang memanfaatkan WebView, sehingga kinerjanya terkadang dianggap kurang optimal dibandingkan dengan framework native seperti Flutter atau React Native yang menghasilkan kode asli secara langsung.
 - Keterbatasan akses ke fitur native, Walaupun Ionic menyediakan plugin seperti Cordova dan Capacitor untuk mengakses fitur perangkat, tidak semua fitur native didukung secara langsung atau seoptimal dengan framework lain.
 - Tampilan UI / UX, Karena menggunakan teknologi web, tampilan dan responsivitas aplikasi Ionic terkadang kurang terasa "native." Framework seperti Flutter memiliki widget yang dirancang khusus untuk memberikan pengalaman yang sangat mirip dengan aplikasi native.
 - Komunitas dan Ekosistem, Ionic memiliki komunitas yang cukup aktif, tetapi tidak sebesar Flutter atau React Native.
 - Popularitas Teknologi Pendukung, Ionic sangat mengandalkan teknologi web (HTML, CSS, JavaScript), yang kadang dianggap kurang sesuai untuk pengembangan mobile modern dibandingkan framework seperti Flutter (berbasis Dart) atau React Native (JavaScript dengan fokus pada mobile).

2. Ion Refresher pada Ionic Framework adalah komponen yang menyediakan fitur pull-to-refresh, ini berfungsi agar pengguna menarik layar ke bawah untuk memperbarui konten. Fungsinya termasuk memperbarui data dari server atau sumber lain, memberikan interaksi yang lebih intuitif, serta meningkatkan pengalaman pengguna dengan animasi modern. Selain itu, komponen ini mendukung event khusus seperti onIonRefresh, yang memungkinkan pengembang menulis logika kustom untuk pembaruan data atau reset status aplikasi. Dengan implementasi sederhana menggunakan elemen seperti ion-refresher-content dan event handler seperti doRefresh(), Ion Refresher menjadi fitur yang sangat berguna untuk aplikasi dinamis berbasis data.
3. Untuk mencegah pengguna yang sudah login kembali ke halaman login dan menghalangi pengguna yang belum login mengakses halaman home atau dashboard dalam aplikasi Ionic, Anda dapat menggunakan mekanisme route guard atau memanfaatkan local storage/session storage untuk memeriksa status login pengguna. Langkah langkahnya kurang lebih sebagai berikut :
 - **Menyimpan Status login**, Simpan status login pengguna (seperti token atau flag boolean) di dalam storage menggunakan localStorage atau package seperti @ionic/storage.
 - **Route Guard untuk Melindungi Halaman**, Gunakan route guard (misalnya, AuthGuard) untuk memeriksa status login pengguna sebelum masuk ke halaman tertentu.
 - **Menentukan Guard pada Routing**, Tambahkan guard pada routing module di file app-routing.module.ts untuk mengamankan halaman tertentu.
 - **Cegah Kembali ke Halaman Login Setelah Login**, Saat pengguna sudah login dan mencoba mengakses halaman login, Anda bisa memeriksa status login dan redirect mereka ke halaman home/dashboard.
 - **Logout**, Pastikan saat pengguna logout, status login dihapus dari storage, dan pengguna diarahkan kembali ke halaman login.

Penjelasan Login



Di atas adalah tampilan pertama kalinya ketika kita membuka programn, untuk bisa akses ke dalam aplikasi kita akan melakukan login dengan cara memasukkan akun google dengan cara memencet button 'SIGN IN WITH GOOGLE'.

```
<template>
  <ion-page>
    <ion-content fullscreen="true">
      <div id="container">
        <!-- Title -->
        <ion-text style="margin-bottom: 20px; text-align: center;">
          <h1>Katalog Mainan (Responsi)</h1>
        </ion-text>
        <!-- Button Sign In -->
    </ion-content>
  </ion-page>
</template>
```

```

<ion-button @click="login" color="light">

  <ion-icon slot="start" :icon="logoGoogle"></ion-icon>

  <ion-label>Sign In with Google</ion-label>

</ion-button>

</div>

</ion-content>

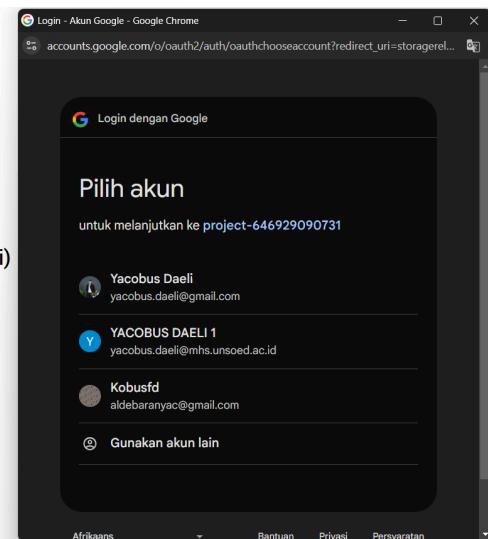
</ion-page>

</template>

```

Diatas adalah Kode diatas adalah kode untuk menunjukkan teks dan button yang ada digambar tampilan login.

Autentikasi Login Page



```
1 <script setup lang="ts">
2 import { IonContent, IonPage, IonButton, IonIcon, IonText, IonLabel } from
'@ionic/vue';
3 import { logoGoogle } from 'ionicons/icons';
4 import { useAuthStore } from '@/stores/auth';
5
6 const authStore = useAuthStore();
7
8 const login = async () => {
9     await authStore.loginWithGoogle();
10 };
11 </script>
```

Untuk imports, kode diatas adalah untuk mengimpor beberapa komponen dari Ionic seperti

IonContent, IonPage, IonButton, IonIcon, IonText, dan IonLabel yang digunakan untuk membangun tampilan halaman.

logoGoogle } from 'ionicons/icons'; , digunakan untuk Mengimpor logoGoogle dari ionicons/icons untuk menampilkan ikon Google dalam UI.

{ useAuthStore } from '@/stores/auth';, digunakan untuk Mengimpor useAuthStore dari file store autentikasi (auth) yang telah didefinisikan sebelumnya. Store ini berisi fungsi-fungsi yang menangani autentikasi pengguna.

const authStore = useAuthStore(); -> untuk kode ini menginisialisasi authStore dengan memanggil useAuthStore(). Ini memungkinkan penggunaan fungsi dan data yang ada dalam store auth di komponen ini.

```
const login = async () => { await authStore.loginWithGoogle();}
```

Untuk kode ini digunakan untuk memanggil authStore.loginWithGoogle() untuk menjalankan proses login dengan Google. Ini menginisialisasi proses autentikasi yang telah didefinisikan di auth.ts.

Sebelum ini adanya pemanggilan instance diatas kita harus membuat file store autentikasi (auth) terlebih dahulu pada file auth.ts. sebelum itu kita Buat sebuah folder stores di dalam folder src . Kemudian buat sebuah file auth.ts di dalam folder stores.

Berikutnya jalankan perintah npm i --save @codetrix-studio/capacitor-google-auth

```
1 // file untuk autentikasi
2 // src/stores/auth.ts
3 import { defineStore } from 'pinia';
4 import { ref, computed } from 'vue';
5 import router from '@/router';
6 import { auth } from '@/utils/firebase';
7 import { GoogleAuthProvider, onAuthStateChanged, signInWithCredential, signOut,
User } from 'firebase/auth';
8 import { GoogleAuth } from '@codetrix-studio/capacitor-google-auth';
9 import { alertController } from '@ionic/vue';
10
11 export const useAuthStore = defineStore('auth', () => {
12   // Variabel User
13   const user = ref<User | null>(null);
14
15   // Variabel isAuthenticated mengembalikan true or false
16   // Cek jika user sudah login atau belum
17   const isAuthenticated = computed(() => user.value !== null);
18
19   // Sign In with Google
20   const loginWithGoogle = async () => {
21     try {
22       await GoogleAuth.initialize({
23         clientId: '646929090731-kj8dalq5npha4hkvvum8bkjg67ro7c7.apps.googleusercontent.com',
24         scopes: ['profile', 'email'],
25         grantOfflineAccess: true,
26       });
27
28       const googleUser = await GoogleAuth.signIn();
29
30       const idToken = googleUser.authentication.idToken;
31
32       const credential = GoogleAuthProvider.credential(idToken);
33
34       const result = await signInWithCredential(auth, credential);
35
36       user.value = result.user;
37
38       router.push("/home");
39     } catch (error) {
40       console.error("Google sign-in error:", error);
41
42       const alert = await alertController.create({
43         header: 'Login Gagal!',
44         message: 'Terjadi kesalahan saat login dengan Google. Coba
lagi.',
45         buttons: ['OK'],
46       });
47
48       await alert.present();
49
50       throw error;
51     }
52   };
53
54   // Logout
55   const logout = async () => {
56     try {
57       await signOut(auth);
58       await GoogleAuth.signOut();
59       user.value = null;
60       router.replace("/login");
61     } catch (error) {
62       console.error("Sign-out error:", error);
63       throw error;
64     }
65   };
66
67   // Fungsi bawaan firebase/auth untuk menyimpan informasi autentikasi
68   // pengguna
69   onAuthStateChanged(auth, (currentUser) => {
70     user.value = currentUser;
71   });
72
73   return { user, isAuthenticated, loginWithGoogle, logout };
74 });


```

Penjelasan kode auth.ts :

Imports

- **defineStore** dari Pinia

Digunakan untuk mendefinisikan store, yaitu objek yang menyimpan state atau data yang bisa diakses di seluruh aplikasi.

- **ref** dan **computed** dari Vue

Untuk mendefinisikan variabel reaktif dan nilai terkomputasi.

- **router** dari `@/router`

Untuk navigasi antar halaman di aplikasi.

- **auth** dari `@/utils/firebase`

Menggunakan beberapa metode dari Firebase Authentication seperti:

- `GoogleAuthProvider`
- `onAuthStateChanged`
- `signInWithCredential`
- `signOut`
- `User`

Metode ini digunakan untuk autentikasi pengguna.

- **GoogleAuth** dari `@codetrix-studio/capacitor-google-auth`

Digunakan untuk autentikasi Google pada platform mobile/hybrid.

- **alertController** dari `@ionic/vue`

Untuk menampilkan alert dalam aplikasi berbasis Ionic.

Store `useAuthStore`

- **const user = ref<User | null>(null);**

Variabel `user` menyimpan data pengguna yang sedang login atau `null` jika belum ada pengguna yang login.

- **const isAuthenticated = computed(() => user.value !== null);**

Properti terkomputasi `isAuthenticated` mengembalikan:

- `true` jika pengguna sudah login (`user` tidak `null`).
- `false` jika pengguna belum login.

Fungsi `loginWithGoogle`

Fungsi ini menangani login menggunakan akun Google dengan langkah berikut:

1. **GoogleAuth.initialize**

Mengatur konfigurasi autentikasi Google, seperti:

- `clientId`
- `scopes`
- `grantOfflineAccess`

2. **GoogleAuth.signIn()**

Membuka layar login G`ouser.value = result.user;`

Menyimpan data pengguna yang berhasil login.

3. **router.push("/home");**

Mengarahkan pengguna ke halaman beranda (`/home`) setelah login.

Fungsi logout

Fungsi ini menangani proses logout dengan langkah berikut:

1. **signOut(auth);**

Logout dari Firebase.

2. **GoogleAuth.signOut();**

Logout dari akun Google.

3. **user.value = null;**

Mengosongkan data pengguna.

4. **router.replace("/login");**

Mengarahkan pengguna ke halaman login setelah logout.

Listener **onAuthStateChanged**

- Fungsi ini otomatis memperbarui data `user` setiap kali ada perubahan status
- **Persiapan Firebase**

Sebelum mengimplementasikan, lakukan langkah berikut:

1. **Integrasi Firebase**

Firebase adalah platform pengembangan aplikasi yang dibuat oleh Google. Platform ini menyediakan layanan dan alat untuk membantu pengembang membangun, meningkatkan, dan mengelola aplikasi secara efisien.

Dalam kasus ini, yang digunakan adalah Firebase Authentication.

2. Membuat folder utils

Buat folder bernama `utils` di dalam folder `src`. Lalu, buat file `firebase.ts` di dalam folder tersebut.

3. Konfigurasi Firebase

Sesuaikan `firebaseConfig` dengan konfigurasi yang telah dibuat sebelumnya di Firebase Console.

- autentikasi, seperti login atau logout.

`firebase.ts`

```
1 // src/utils/firebase.ts
2 import { initializeApp } from "firebase/app";
3 import { getAuth, GoogleAuthProvider } from 'firebase/auth';
4
5 const firebaseConfig = {
6   apiKey: "",
7   authDomain: "",
8   projectId: "",
9   storageBucket: "",
10  messagingSenderId: "",
11  appId: "",
12 };
13
14 const firebase = initializeApp(firebaseConfig);
15 const auth = getAuth(firebase);
16 const googleProvider = new GoogleAuthProvider();
17
18 export { auth, googleProvider };
```

Penjelasan kode :

```
1 import { initializeApp } from "firebase/app";
2 import { getAuth, GoogleAuthProvider } from "firebase/auth";
```

`initializeApp`

Fungsi untuk menginisialisasi aplikasi Firebase dengan konfigurasi yang diberikan.

getAuth

Fungsi untuk mendapatkan instance dari layanan autentikasi Firebase.

GoogleAuthProvider

Objek yang digunakan untuk mengatur login dengan Google.

Konfigurasi Firebase

```
const firebaseConfig = {  
  apiKey: "",  
  authDomain: "",  
  projectId: "",  
  storageBucket: "",  
  messagingSenderId: "",  
  appId: "",  
  measurementId: ""  
};
```

Objek `firebaseConfig` berisi konfigurasi untuk menghubungkan aplikasi ke proyek Firebase.

Beberapa komponennya:

- **apiKey**: Kunci API untuk aplikasi.
- **authDomain**: Domain autentikasi aplikasi Firebase.
- **projectId**: ID proyek Firebase.
- **storageBucket**: Alamat penyimpanan file Firebase.
- **messagingSenderId**: ID pengirim notifikasi Firebase.
- **appId**: ID aplikasi Firebase.
- **measurementId**: ID untuk analitik Firebase.

Inisialisasi Firebase

```
1 const firebase = initializeApp(firebaseConfig);
```

initializeApp(firebaseConfig)

Fungsi ini digunakan untuk menginisialisasi aplikasi Firebase dengan konfigurasi yang diberikan, menghubungkan aplikasi ke proyek Firebase.

Inisialisasi Autentikasi dan Provider Google

```
1 const auth = getAuth(firebase);
2 const googleProvider = new GoogleAuthProvider();
```

- **getAuth(firebase)**

Mengambil instance dari layanan autentikasi Firebase.

- **googleProvider = new GoogleAuthProvider()**

Membuat instance dari **GoogleAuthProvider** untuk autentikasi menggunakan akun Google.

Ekspor **auth** dan **googleProvider**

- **export { auth, googleProvider }**

Menyediakan **auth** dan **googleProvider** untuk digunakan di bagian lain aplikasi, misalnya untuk mengelola autentikasi pengguna dengan Google.

Penjelasan CRUD

Tampilan untuk homepage



Penjelasan code :

- Header

```
1 <ion-header :translucent="true">
2   <ion-toolbar>
3     <ion-title>Home</ion-title>
4   </ion-toolbar>
5 </ion-header>
```

ion-header: Membuat header halaman.

ion-toolbar: Toolbar di dalam header.

ion-title: Menampilkan teks judul "Home".

- Refresher

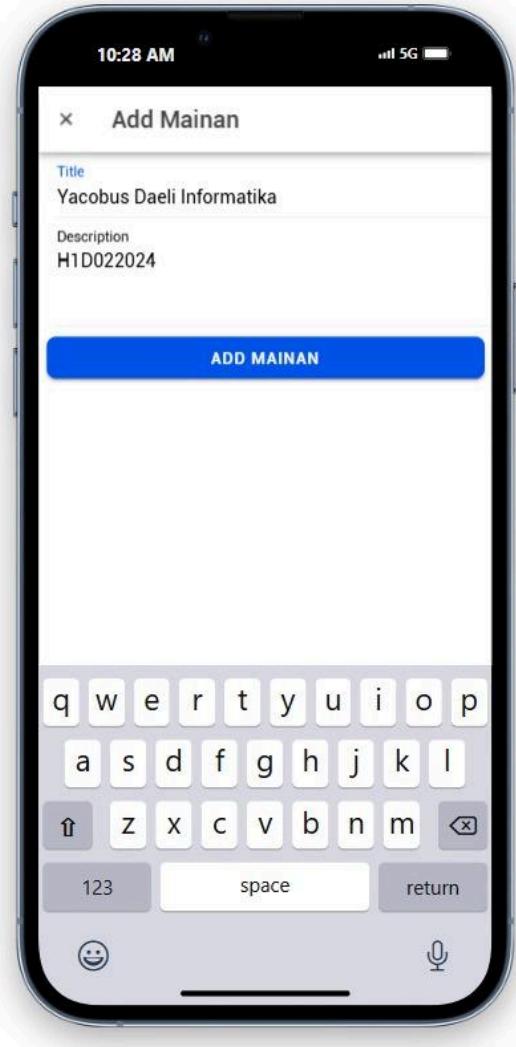
```
1 <ion-refresher slot="fixed" :pull-factor="0.5" :pull-min="100" :pull-max="200"
2   @ionRefresh="handleRefresh($event)">
3   <ion-refresher-content></ion-refresher-content>
4 </ion-refresher>
```

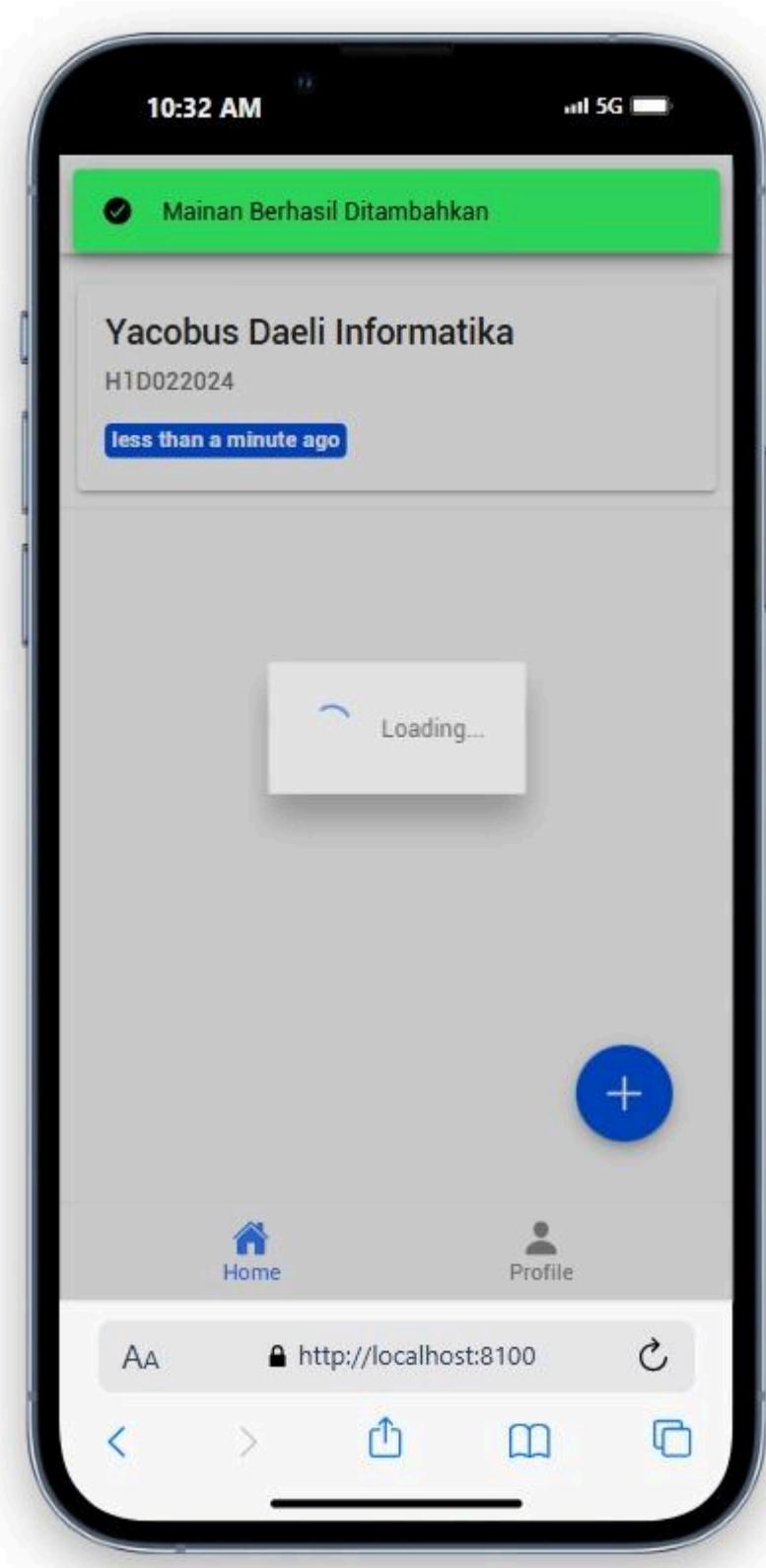
Ion refresher digunakan untuk menyegarkan data dengan gesture menarik layar ke bawah.

pull-factor: Mengontrol sensitivitas tarik. pull-min & pull-max: Menentukan jarak minimum/maksimum untuk memicu refresh.

@ionRefresh : Memanggil fungsi handleRefresh saat gesture selesai.

Menambahkan data mainan

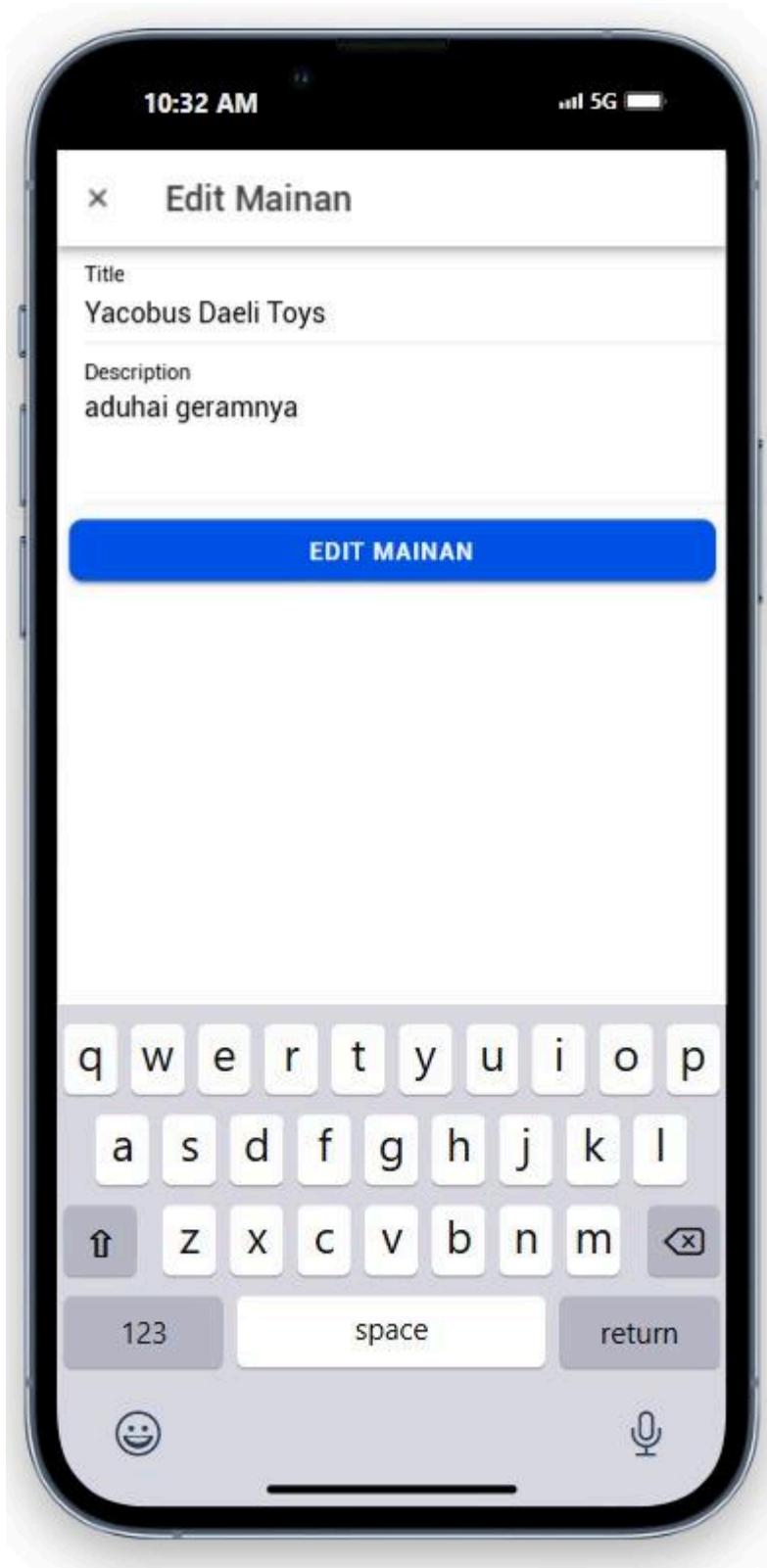


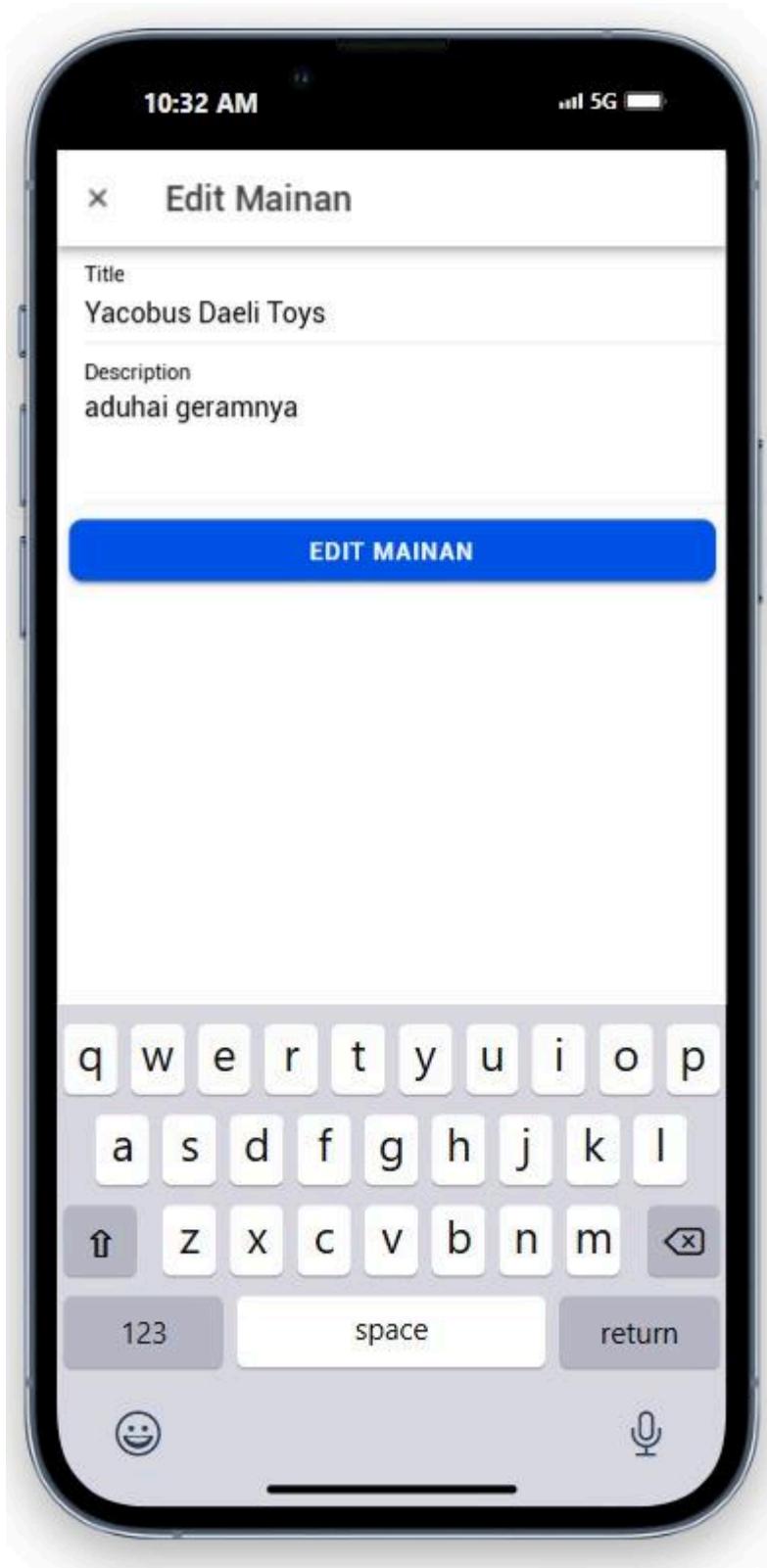


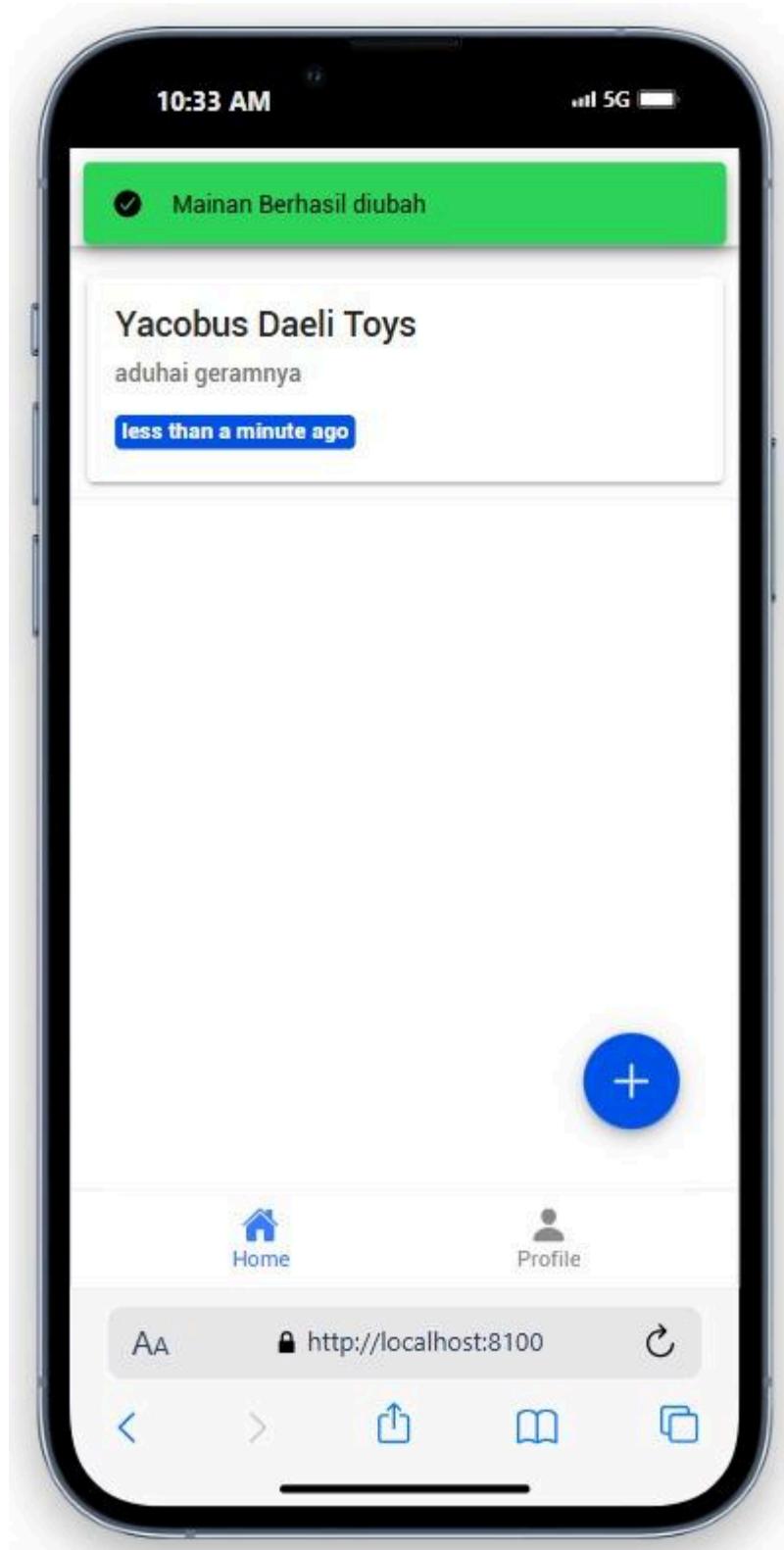
```
1  async addTodo(todo: Omit<Todo, 'id'>) {
2      const todoRef = this.getTodoRef();
3      const docRef = await addDoc(todoRef, {
4          ...todo,
5          status: false,
6          createdAt: Timestamp.now(),
7          updatedAt: Timestamp.now()
8      });
9      return docRef.id;
10 }
```

Menambahkan data ke daftar lengkap mainan.

Mengedit Data Mainan







```
1  async updateTodo(id: string, todo: Partial<Todo>) {
2      const todoRef = this.getTodoRef();
3      const docRef = doc(todoRef, id);
4      await updateDoc(docRef, {
5          ...todo,
6          updatedAt: Timestamp.now()
7      });
8  }
```

Kode diatas adalah untuk memperbarui data dengan ID tertentu.

Menghapus Data Mainan



```
1  async deleteTodo(id: string) {  
2      const todoRef = this.getTodoRef();  
3      const docRef = doc(todoRef, id);  
4      await delete
```

Kode diatas adalah untuk menghapus data berdasarkan ID.