Destian Ardan Alfatanu

H1D022045

Responsi Mobile 2

Shift E (Baru)

Shift A (KRS)

**Jawaban Soal**

1. - Ionic menggunakan teknologi berbasis web (HTML, CSS, dan JavaScript) dan berjalan di WebView. Ini membuat performa aplikasi tidak sebaik aplikasi native yang dibuat dengan React Native atau Flutter.

   - Aplikasi Ionic cenderung terlihat seperti aplikasi web dibandingkan aplikasi native, yang dapat mengurangi pengalaman pengguna, terutama pada animasi dan interaksi yang kompleks.

   - Framework seperti React Native dan Flutter lebih menarik bagi banyak pengembang karena mereka menawarkan performa yang lebih dekat dengan aplikasi native.

2. - Menyediakan pengalaman pengguna untuk memperbarui data dengan menarik layar.

   - Menjalankan fungsi tertentu saat pengguna melepaskan refresh.

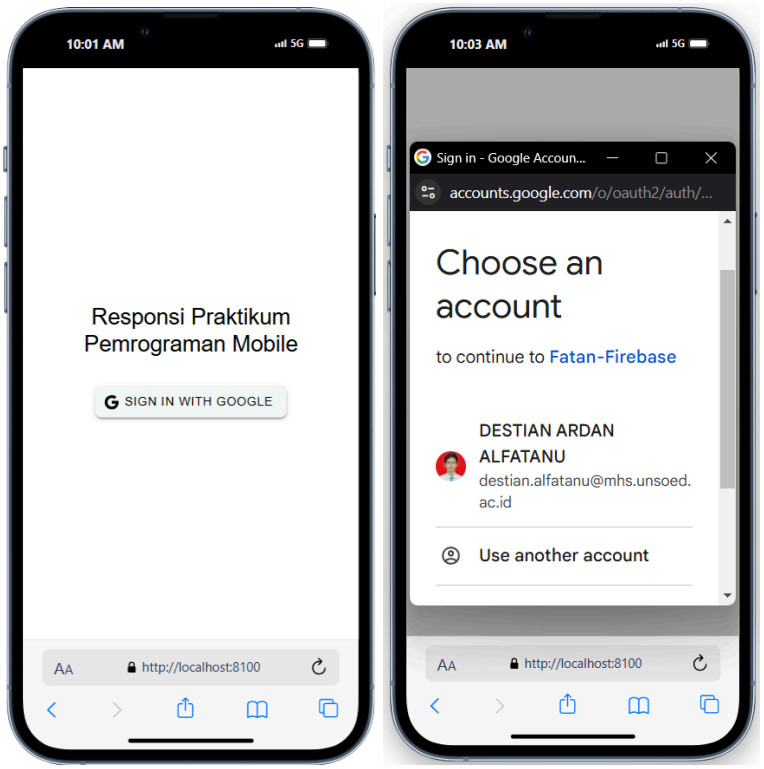
3. - Membuat Auth Guard : Memastikan hanya pengguna yang sudah login bisa mengakses halaman Home

   - Membuat NoAuthGuard : Mencegah Akses ke Halaman Login jika Sudah Login

   - Menggunakan Guard di app-routing.module.ts

# Penjelasan CRUD dan Login

## Login



Saat kita mengakses aplikasi, halaman pertama yang muncul adalah LoginPage. Kita bisa login menggunakan akun google yang kita punya. Langsung saja klik tombol SIGN IN WITH GOOGLE selanjutnya ada diarahkan untuk memilih akan menggunakan akun yang mana. Ini adalah bagaimana sistem login diterapkan

```
// Sign In with Google
    const loginWithGoogle = async () => {
        try {
            await GoogleAuth.initialize({
                clientId: '466997294826-n4bh0dfjconjalhei7mn0ldgglnmphbq.apps.googleusercontent.com',
                scopes: ['profile', 'email'],
                grantOfflineAccess: true,
            });

            const googleUser = await GoogleAuth.signIn();

            const idToken = googleUser.authentication.idToken;

            const credential = GoogleAuthProvider.credential(idToken);

            const result = await signInWithCredential(auth, credential);

            user.value = result.user;

            router.push("/home");
        } catch (error) {
            console.error("Google sign-in error:", error);

            const alert = await alertController.create({
                header: 'Login Gagal!',
                message: 'Terjadi kesalahan saat login dengan Google. Coba lagi.',
                buttons: ['OK'],
            });

            await alert.present();

            throw error;
        }
    };
```

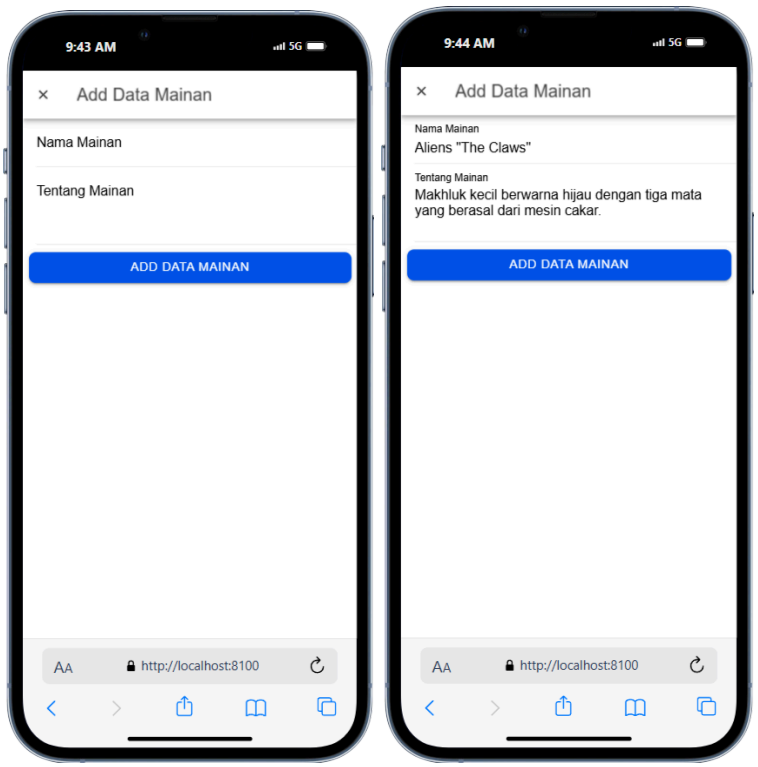
Tampilan LoginPage didapatkan dari kode di bawah ini

```vue
<template>
    <ion-page>
        <ion-content :fullscreen="true">
            <div id="container">
                <!-- Title -->
                <ion-text style="margin-bottom: 20px; text-align: center;">
                    <h1>Responsi Praktikum Pemrograman Mobile</h1>
                </ion-text>

                <!-- Button Sign In -->
                <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>

<style>
#container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100%;
}

ion-button {
    --border-radius: 8px;
}
</style>
```

## Home

Setelah Login kita akan masuk ke HomePage. Di HomePage ini kita dapat menambahkan data mainan yang akan kita masukkan ke dalam database. Tampilan dari HomePage berasal dari kode di bawah ini

```vue
<template>
  <ion-page>
    <ion-header :translucent="true">
      <ion-toolbar>
        <ion-title>HOME</ion-title>
      <!-- Logout Button -->
        <ion-button slot="end" fill="clear" @click="logout" style="--color: gray;">
            <ion-icon slot="end" :icon="exit"></ion-icon>
            <ion-label>Logout</ion-label>
         </ion-button>
      </ion-toolbar>
    </ion-header>

    <ion-content :fullscreen="true">
      <!-- Komponen Refresher -->
      <ion-refresher
        slot="fixed"
        :pull-factor="0.5"
        :pull-min="100"
        :pull-max="200"
        @ionRefresh="handleRefresh($event)"
      >
        <ion-refresher-content></ion-refresher-content>
      </ion-refresher>

      <!-- Kata Sambutan -->
      <div class="welcome-message">
        <h2>Selamat Datang Di Aplikasi Toys Story</h2>
      </div>

      <!-- Tombol Tambahkan Data di Tengah -->
      <div class="button-container">
        <ion-button expand="block" @click="isOpen = true">
          Tambahkan Data Mainan
        </ion-button>
      </div>

      <!-- Modal untuk Input -->
      <InputModal
        v-model:isOpen="isOpen"
        v-model:editingId="editingId"
        :mainan="mainan"
        @submit="handleSubmit"
      />
    </ion-content>
  </ion-page>
</template>
```



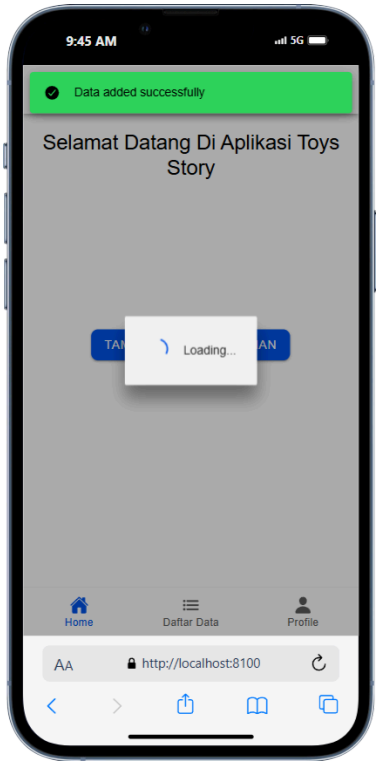

Jika kita ingin menambahkan data mainan, kita bisa menekan tombol “Tambahkan Data Mainan”. Kemudian halaman yang ditampilkan adalah AddPage. Kita harus mengisi semua kolom yang ada di formulir. Jika dirasa sudah benar langsung saja klik “ADD DATA MAINAN”. Tampilan dari AddPage didapatkan dari kode

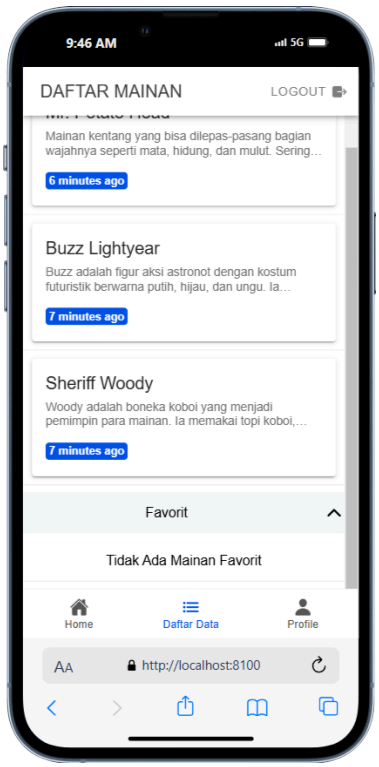
```
<template>
    <ion-modal :is-open="isOpen" @did-dismiss="cancel">
        <ion-header>
            <ion-toolbar>
                <ion-title>{{ editingId ? '' : 'Add' }} Data Mainan</ion-title>
                <ion-buttons slot="start">
                    <ion-button @click="cancel"><ion-icon :icon="close"></ion-icon></ion-button>
                </ion-buttons>
            </ion-toolbar>
        </ion-header>
        <ion-content>
            <ion-item>
                <ion-input v-model="localMainan.title" label="Nama Mainan" label-placement="floating"
                    placeholder="Masukkan Nama Mainan"></ion-input>
            </ion-item>
            <ion-item>
                <ion-textarea v-model="localMainan.description" label="Tentang Mainan" label-placement="floating"
                    placeholder="Masukkan Deskripsi Mainan" :autogrow="true" :rows="3"></ion-textarea>
            </ion-item>
            <ion-row>
                <ion-col>
                    <ion-button type="button" @click="input" shape="round" color="primary" expand="block">
                        {{ editingId ? '' : 'Add' }} Data Mainan
                    </ion-button>
                </ion-col>
            </ion-row>
        </ion-content>
    </ion-modal>
</template>
```

Logika yang digunakan dalam AddPage ini berasal dari kode

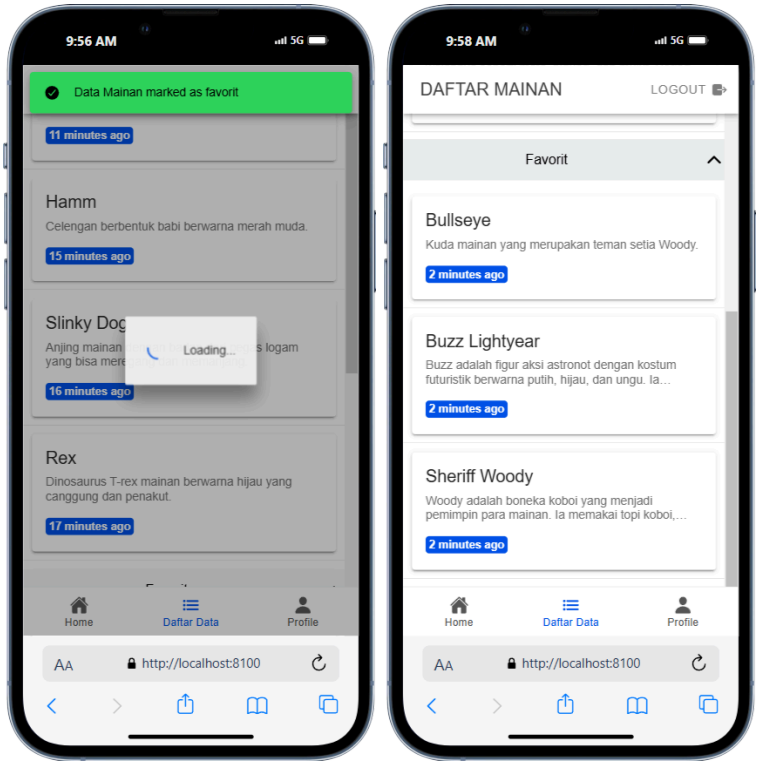
```ts
// handle submit add/edit pada modal
const handleSubmit = async (mainan: Omit<Mainan, 'id' | 'createdAt' | 'updatedAt' | 'status'>) => {
  if (!mainan.title) {
    await showToast('Title is required', 'warning', warningOutline);
    return;
  }
  try {
      await firestoreService.addMainan(mainan as Mainan);
      await showToast('Data added successfully', 'success', checkmarkCircle);
  } catch (error) {
    await showToast('An error occurred', 'danger', closeCircle);
    console.error(error);
  } finally {
    editingId.value = null;
  }
};
```

Kemudian ketika sudah berhasil menambahkan data maka akan muncul notifikasi succes seperti gambar di bawah ini



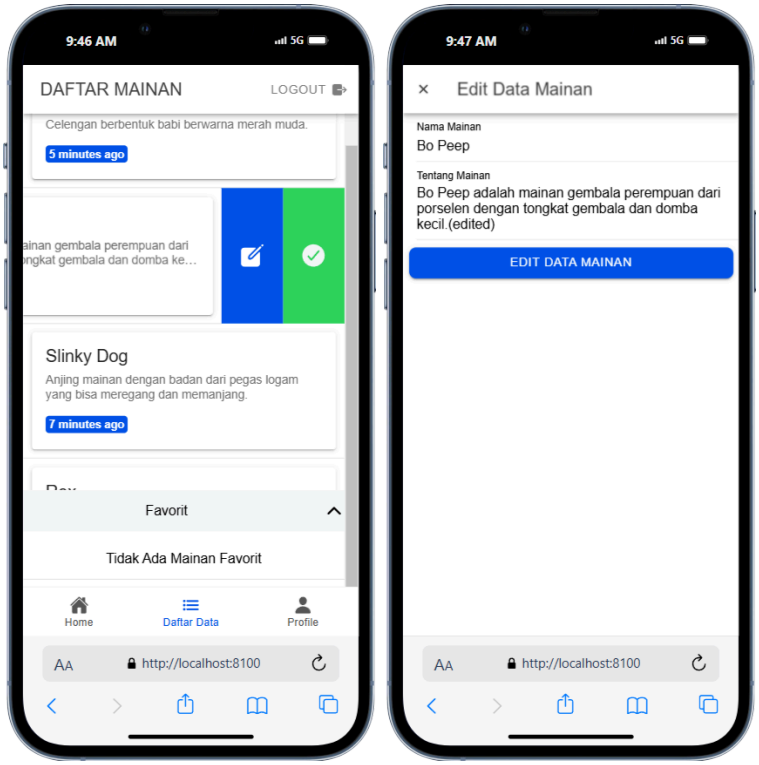



Data yang sudah ditambahkan bisa dilihat di Halaman DaftarPage. Tampilan DaftarPage bisa dilihat di atas. Dan terdapat mainan yang bisa ditambahkan ke Daftar Favorit seperti gambar di bawah ini



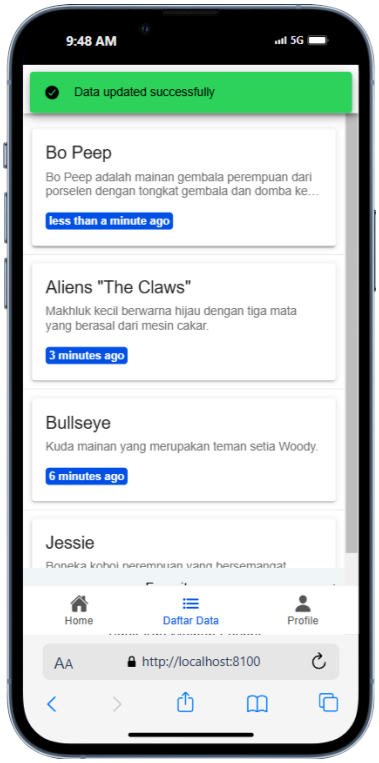
Tampilan DaftarPage berasal dari kode di bawah ini

```vue
<template>
    <ion-page>
      <ion-header :translucent="true">
        <ion-toolbar>
          <ion-title>DAFTAR MAINAN</ion-title>
          <!-- Logout Button -->
        <ion-button slot="end" fill="clear" @click="logout" style="--color: gray;">
            <ion-icon slot="end" :icon="exit"></ion-icon>
            <ion-label>Logout</ion-label>
        </ion-button>
        </ion-toolbar>
      </ion-header>

      <!-- modifikasi src/views/HomePage.vue pada bagian ion-content dalam template -->
  <ion-content :fullscreen="true">
      <!-- komponen paling atas dari ion-content -->
      <ion-refresher slot="fixed" :pull-factor="0.5" :pull-min="100" :pull-max="200"
      @ionRefresh="handleRefresh($event)">
      <ion-refresher-content></ion-refresher-content>
    </ion-refresher>

    <!-- bagian refresher -->
  <!-- normal mainans -->
  <div class="scrollable-container">
    <ion-list>
      <ion-item-sliding v-for="mainan in activeMainans" :key="mainan.id" :ref="(el) => setItemRef(el, mainan.id!)">
        <ion-item-options side="start" @ionSwipe="handleDelete(mainan)">
          <ion-item-option color="danger" expandable @click="handleDelete(mainan)">
            <ion-icon slot="icon-only" :icon="trash" size="large"></ion-icon>
          </ion-item-option>
        </ion-item-options>

        <ion-item>
          <ion-card>
            <ion-card-header>
              <ion-card-title class="ion-text-wrap limited-text">{{ mainan.title }}</ion-card-title>
              <ion-card-subtitle class="limited-text">{{ mainan.description }}</ion-card-subtitle>
            </ion-card-header>

            <ion-card-content>
              <ion-badge>{{ getRelativeTime(mainan.updatedAt) }}</ion-badge>
            </ion-card-content>
          </ion-card>
        </ion-item>

        <ion-item-options side="end" @ionSwipe="handleStatus(mainan)">
          <ion-item-option @click="handleEdit(mainan)">
            <ion-icon slot="icon-only" :icon="create" size="large"></ion-icon>
          </ion-item-option>
          <ion-item-option color="success" expandable @click="handleStatus(mainan)">
            <ion-icon slot="icon-only" :icon="checkmarkCircle" color="light" size="large"></ion-icon>
          </ion-item-option>
        </ion-item-options>
      </ion-item-sliding>
      <ion-item v-if="activeMainans.length === 0" class="ion-text-center">
        <ion-label>Tidak Ada Data Mainan</ion-label>
      </ion-item>
    </ion-list>
  </div>

  <!-- completed mainans -->
  <ion-item class="accordion-container">
    <ion-accordion-group>
      <ion-accordion value="first">
        <ion-item slot="header" color="light">
          <ion-label class="ion-text-center">Favorit</ion-label>
        </ion-item>
        <div slot="content" class="scrollable-container">
          <ion-list>
            <ion-item-sliding v-for="mainan in completedMainans" :key="mainan.id" :ref="(el) => setItemRef(el, mainan.id!)">
              <ion-item-options side="start" @ionSwipe="handleDelete(mainan)">
                <ion-item-option color="danger" expandable @click="handleDelete(mainan)">
                  <ion-icon slot="icon-only" :icon="trash" size="large"></ion-icon>
                </ion-item-option>
              </ion-item-options>

              <ion-item>
                <ion-card>
                  <ion-card-header>
                    <ion-card-title class="ion-text-wrap limited-text">{{ mainan.title }}</ion-card-title>
                    <ion-card-subtitle class="limited-text">{{ mainan.description }}</ion-card-subtitle>
                  </ion-card-header>

                  <ion-card-content>
                    <ion-badge>{{ getRelativeTime(mainan.updatedAt) }}</ion-badge>
                  </ion-card-content>
                </ion-card>
              </ion-item>

              <ion-item-options side="end" @ionSwipe="handleStatus(mainan)">
                <ion-item-option @click="handleEdit(mainan)">
                  <ion-icon slot="icon-only" :icon="create" size="large"></ion-icon>
                </ion-item-option>
                <ion-item-option color="warning" expandable @click="handleStatus(mainan)">
                  <ion-icon slot="icon-only" :icon="close" color="light" size="large"></ion-icon>
                </ion-item-option>
              </ion-item-options>
            </ion-item-sliding>
            <ion-item v-if="completedMainans.length === 0" class="ion-text-center">
              <ion-label>Tidak Ada Mainan Favorit</ion-label>
            </ion-item>
          </ion-list>
        </div>
      </ion-accordion>
    </ion-accordion-group>
  </ion-item>

  <!-- bagian tombol dan modal -->
    <ion-fab vertical="bottom" horizontal="end" slot="fixed">
    </ion-fab>
    <inputModal v-model:isOpen="isOpen" v-model:editingId="editingId" :mainan="mainan" @submit="handleSubmit" />
  </ion-content>
    </ion-page>
  </template>
```

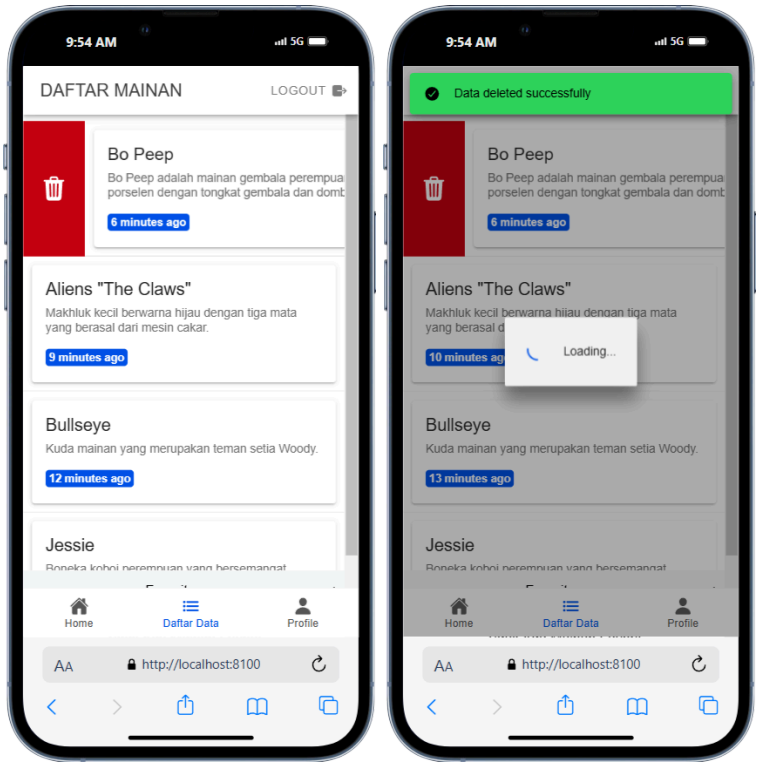



Di DaftarPage juga terdapat fitur Edit yang bisa kita pakai ketika ada kesalahan dalam penginputan data. Cara mengeditnya tinggal kalian geser ke kiri lalu pilih yang tombolnya warna biru. Setelah itu kita bisa mengisi kolom yang ingin kita ubah. Untuk kode logika sama seperti di bagian AddPage.

Setelah klik Edit data akan tersimpan dan notifikasi akan muncul seperti yang ada pada gambar di bawah ini



Jika kita ingin menghapus data yang ada, kita bisa menggeser ke kanan dan klik tombol Delete. Setelah delete berhasil dilakukan akan muncul notifikasi.




Logika penghapusan ini didapat dari kode

```
// handle delete click/swipe
  const handleDelete = async (deleteMainan: Mainan) => {
    try {
      await firestoreService.deleteMainan(deleteMainan.id!);
      await showToast('Data deleted successfully', 'success', checkmarkCircle);
      loadMainans();
    } catch (error) {
      await showToast('Failed to delete Data', 'danger', closeCircle);
      console.error(error);
    }
  };
```

Semua logika CRUD didapatkan dari kode

```
// operasi CRUD
export const firestoreService = {
    // get collection ref
    getMainanRef() {
        const uid = auth.currentUser?.uid;
        if (!uid) throw new Error('User not authenticated');
        return collection(db, 'users', uid, 'mainans');
    },

        // create
    async addMainan(mainan: Omit<Mainan, 'id'>) {
        try {
            const mainanRef = this.getMainanRef();
            const docRef = await addDoc(mainanRef, {
                ...mainan,
                status: false,
                createdAt: Timestamp.now(),
                updatedAt: Timestamp.now()
            });
            return docRef.id;
        } catch (error) {
            console.error('Error Tambah Data:', error);
            throw error;
        }
    },

        // read
    async getMainans(): Promise<Mainan[]> {
        try {
            const mainanRef = this.getMainanRef();
            const q = query(mainanRef, orderBy('updatedAt', 'desc'));
            const snapshot = await getDocs(q);
            return snapshot.docs.map((doc) => ({
                id: doc.id,
                ...doc.data()
            } as Mainan));
        } catch (error) {
            console.error('Error Get Data:', error);
            throw error;
        }
    },

        // update
    async updateMainan(id: string, mainan: Partial<Mainan>) {
        try {
            const mainanRef = this.getMainanRef();
            const docRef = doc(mainanRef, id);
            await updateDoc(docRef, {
                ...mainan,
                updatedAt: Timestamp.now()
            });
        } catch (error) {
            console.error('Error Update Data:', error);
            throw error;
        }
    },

        // delete
    async deleteMainan(id: string) {
        try {
            const mainanRef = this.getMainanRef();
            const docRef = doc(mainanRef, id);
            await deleteDoc(docRef);
        } catch (error) {
            console.error('Error Delete Data:', error);
            throw error;
        }
    },

        // update status
    async updateStatus(id: string, status: boolean) {
        try {
            const mainanRef = this.getMainanRef();
            const docRef = doc(mainanRef, id);
            await updateDoc(docRef, { status: status, updatedAt: Timestamp.now() });
        } catch (error) {
            console.error('Error Update Status:', error);
            throw error;
        }
    }

}
```