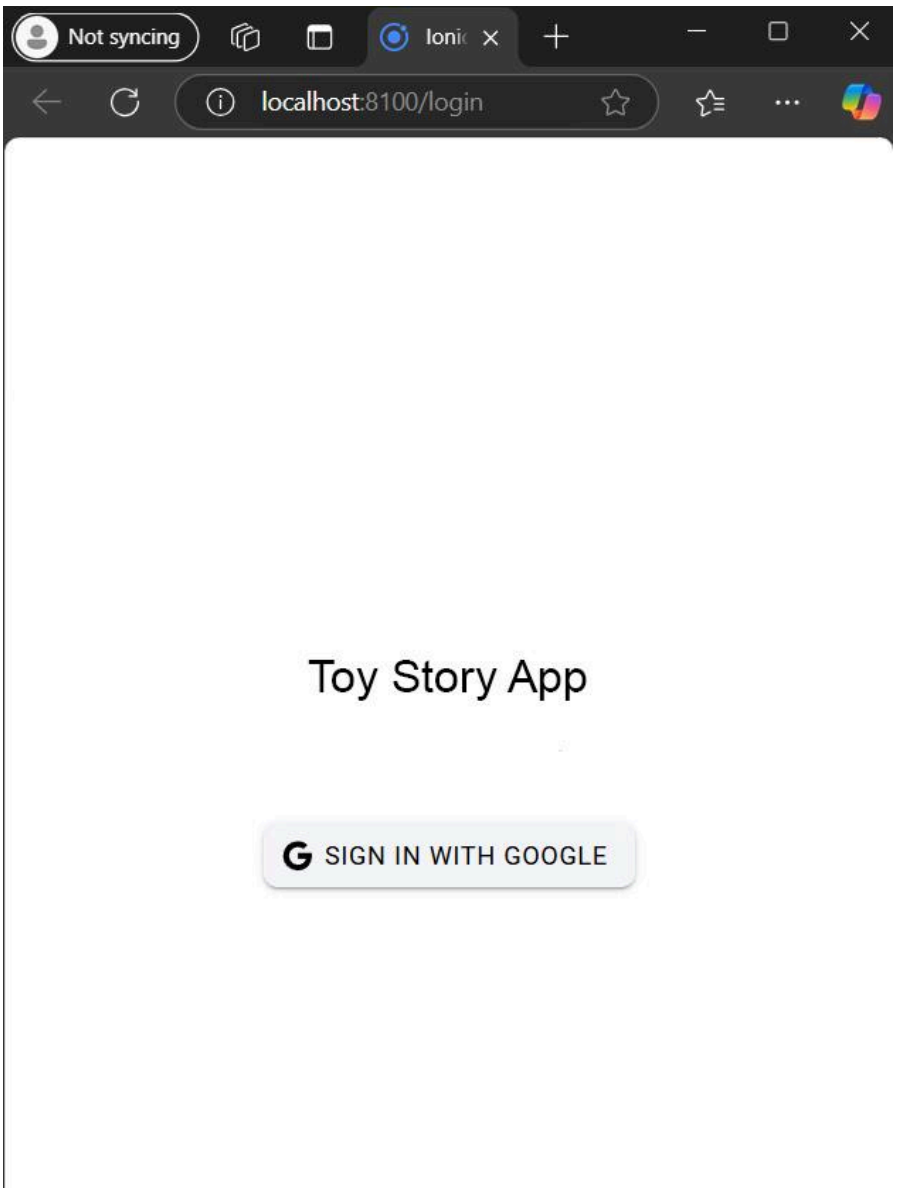
Nama : Fatur Sakti Arrafi
NIM : H1D022041
Shift KRS : A
Shift Baru : E

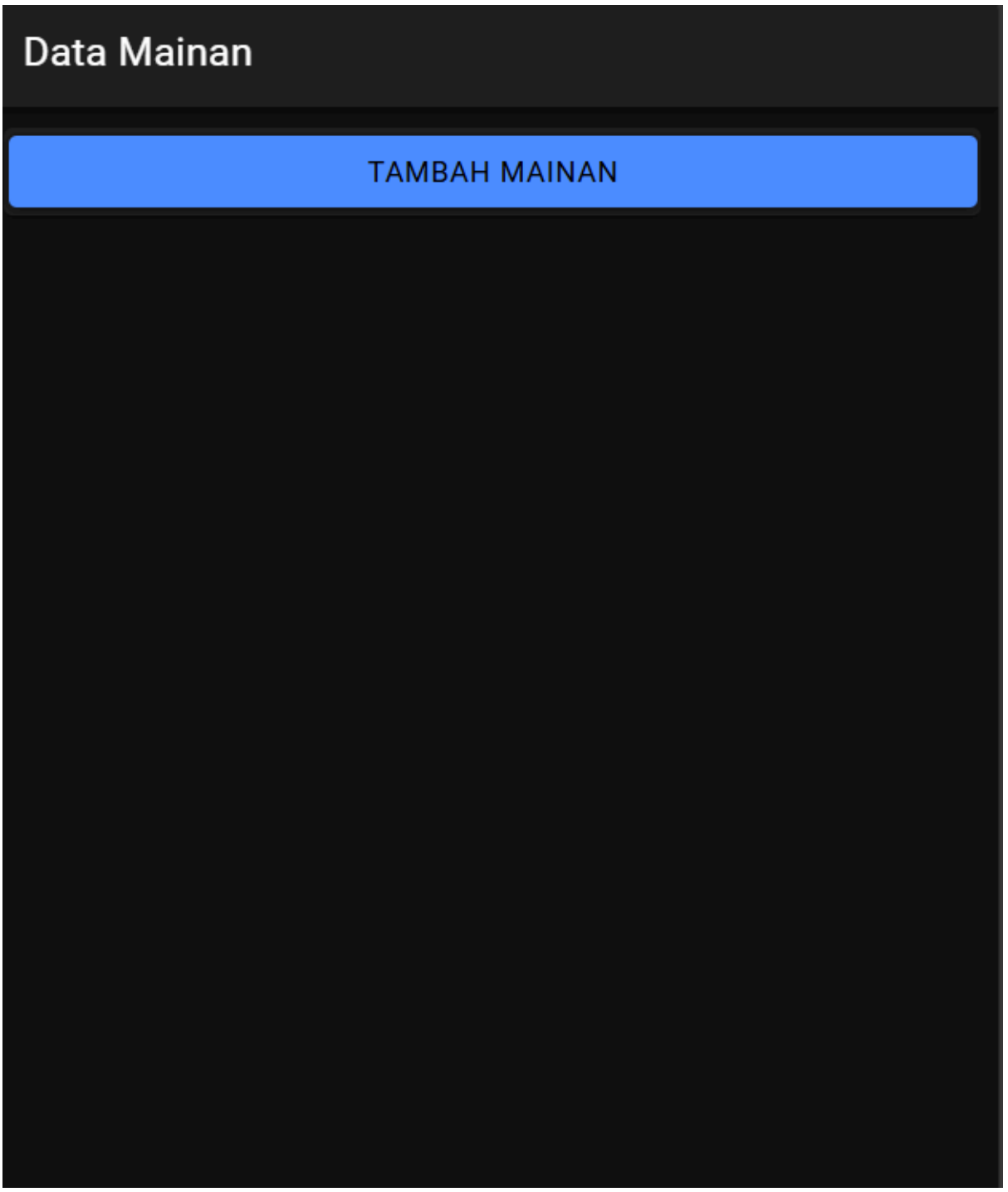
RESPONSI 2 - PEMROGRAMAN MOBILE

1. **Tampilan Login**



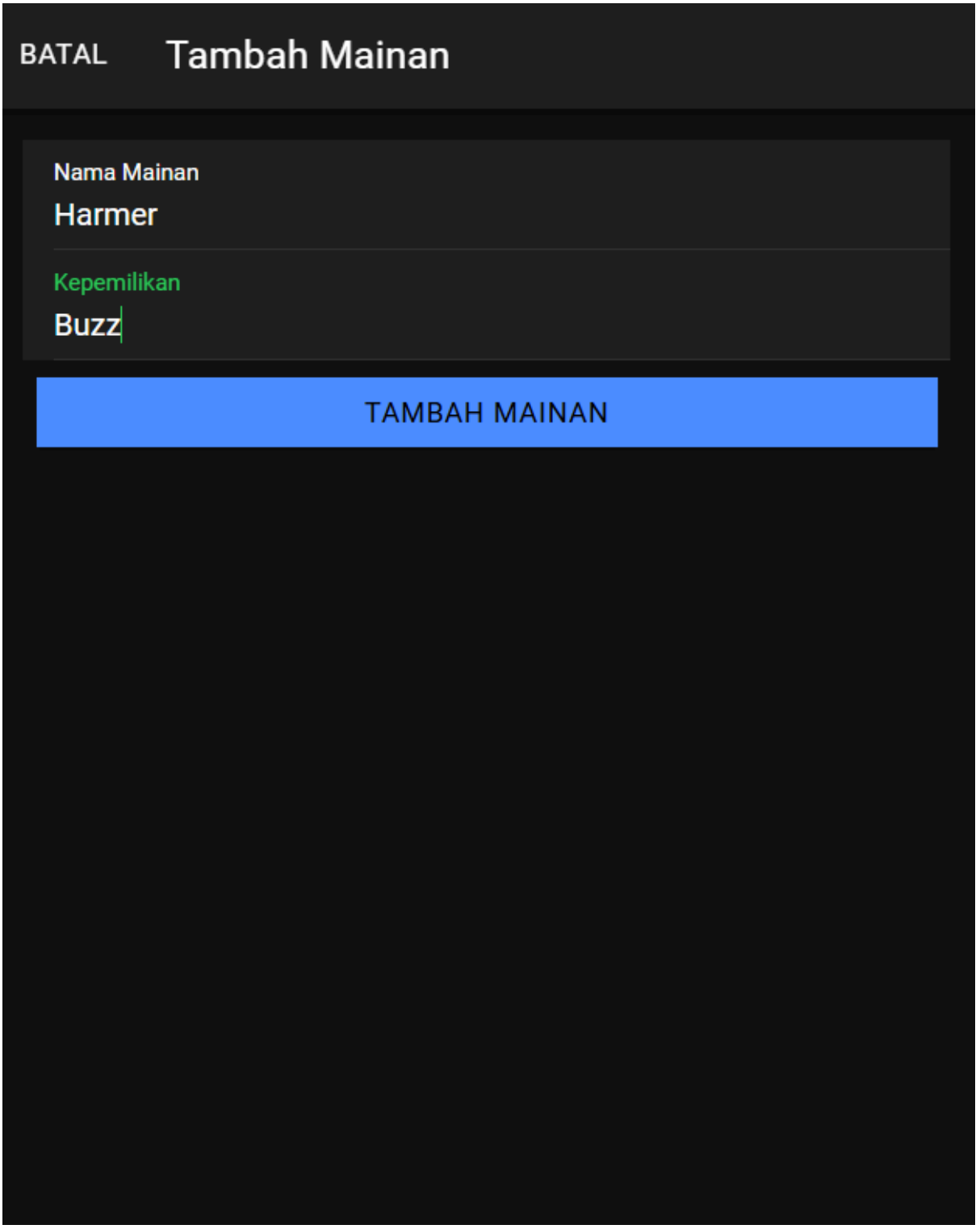
Keterangan : Ini merupakan tampilan paling awal pada ketika menjalankan Ionic Serve, terdapat button untuk melakukan login menggunakan akun Google

2. **Tampilan Data Mainan**



Keterangan: Pada halaman ini, terdapat Button “Tambah Mainan” untuk menambahkan item/nama mainan yang ingin ditambahkan.

3. **Tampilan Tambah Mainan**



Keterangan : Pada Halaman ini, kita dapat menambahkan data mainan dengan memasukkan **“Nama Mainan”** dan **“Kepemilikan”**. Kepemilikan disini maksudnya adalah nama pemilik mainan tersebut. setelah selesai mengisi, klik **“Tambah Mainan”**, maka data akan ditambahkan secara otomatis.

SOAL PADA FILE “RESPONSI 2 PAKET C.PDF”

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. a. Performa Kurang pada Aplikasi Berat

   Untuk aplikasi yang memerlukan animasi kompleks atau rendering grafis yang intensif, Ionic bisa terasa lebih lambat dibanding framework lain karena berjalan di atas teknologi web.

   b. Pengalaman Pengguna Kurang Konsisten

   Karena tampilan Ionic sangat bergantung pada rendering CSS dan HTML, aplikasi terkadang terlihat seperti aplikasi web biasa (bukan aplikasi native), yang dapat mempengaruhi pengalaman pengguna.

   c. Kompleksitas Pengaturan untuk Platform yang Berbeda

   Meskipun mendukung **Android**, **iOS**, dan **Web**, pengaturan untuk setiap platform dapat menjadi lebih rumit, terutama jika aplikasi menggunakan plugin native yang tidak sepenuhnya kompatibel.

2. Komponen **ion-refresher** digunakan untuk membuat fitur *pull-to-refresh* di aplikasi Ionic. Fitur ini memungkinkan pengguna untuk menarik layar ke bawah untuk memuat ulang atau merefresh data pada halaman.

   Fungsi Utama:

   a. Meningkatkan pengalaman pengguna
   b. Memuat/Merefresh Ulang Data

3. a. Buat Auth Guard

   ng generate guard auth

   b. Edit file **auth.guard.ts**

   import { Injectable } from '@angular/core';

   import { CanActivate, Router } from '@angular/router';

```
@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate {
  constructor(private router: Router) {}

  canActivate(): boolean {
    const isLoggedIn = !!localStorage.getItem('user'); // Periksa jika pengguna sudah login
    if (!isLoggedIn) {
      this.router.navigate(['/login']); // Redirect ke halaman login jika belum login
      return false;
    }
    return true; // Izinkan akses jika sudah login
  }
}
```

c. Terapkan Guard pada Routing

Edit **app-routing.module.ts**

```
const routes: Routes = [
  { path: 'login', loadChildren: () => import('./login/login.module').then(m => m.LoginPageModule) },
  { path: 'home', loadChildren: () => import('./home/home.module').then(m => m.HomePageModule), canActivate: [AuthGuard] },
];
```

d. Redirect Pengguna yang Sudah Login dari Halaman Login

Di halaman login (**login.page.ts**), periksa apakah pengguna sudah login dan alihkan mereka ke halaman home

```
import { Router } from '@angular/router';


constructor(private router: Router) {}


ngOnInit() {
  const isLoggedIn = !!localStorage.getItem('user'); // Periksa status login
  if (isLoggedIn) {
    this.router.navigate(['/home']); // Redirect ke home jika sudah login
  }
}


login() {
  // Simulasi login
  localStorage.setItem('user', JSON.stringify({ email: 'user@example.com' }));
  this.router.navigate(['/home']);
}
```

e. Log Out

Tambahkan fungsi logout untuk membersihkan status login

```
logout() {
  localStorage.removeItem('user'); // Hapus data login
  this.router.navigate(['/login']); // Redirect ke halaman login
}
```