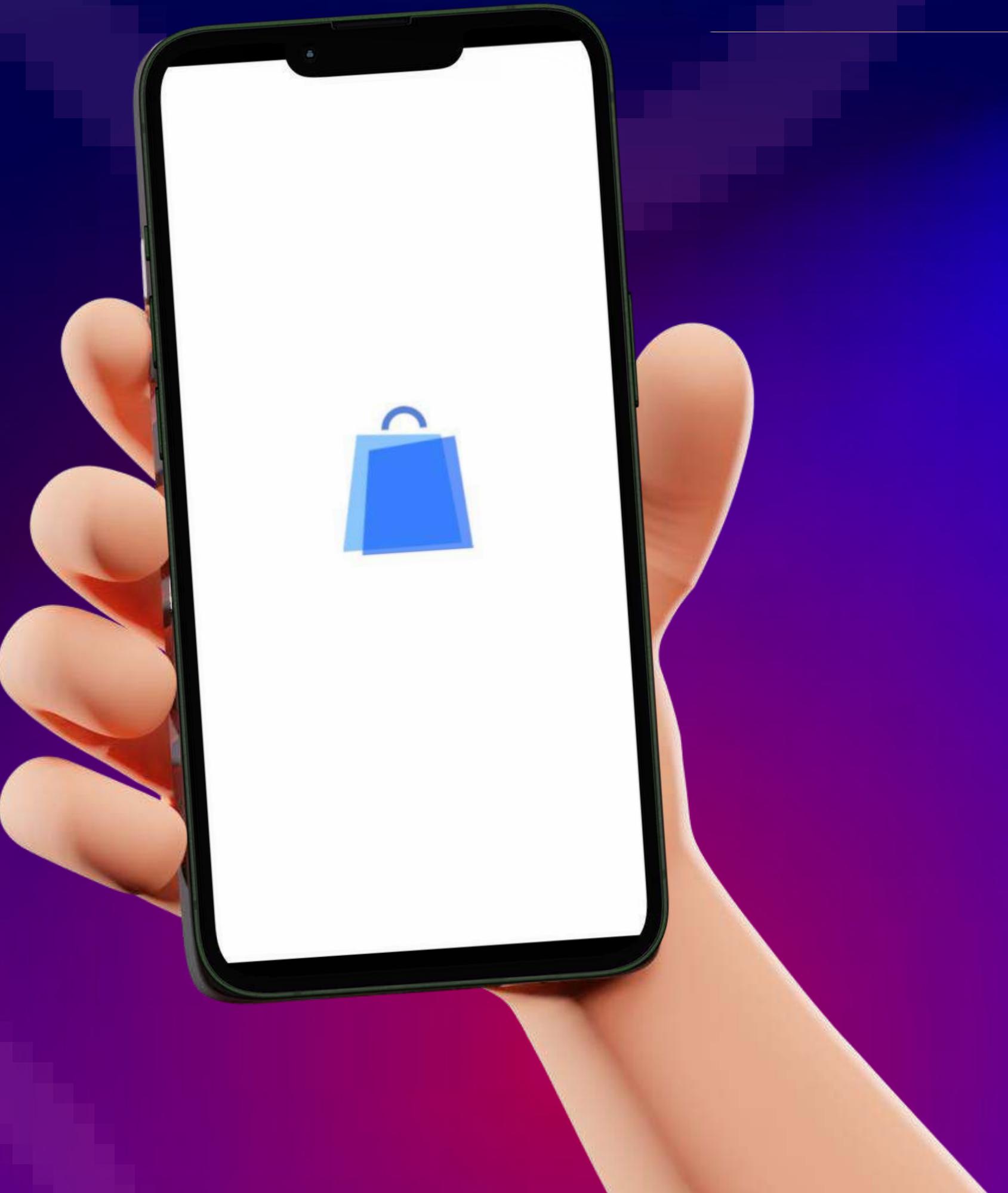




E-COMMERCE APP

Presented by SW3E

DEPI Project presentation



PROFESSIONAL TEAM

Team Members



Gaser Youssef
Software Engineer



Arsany Morcos
Mobile App developer



Mohamed
Flutter Developer



Mariem Elsaid
Flutter Developer



Malak Amgad
AI Engineer

AGANDA

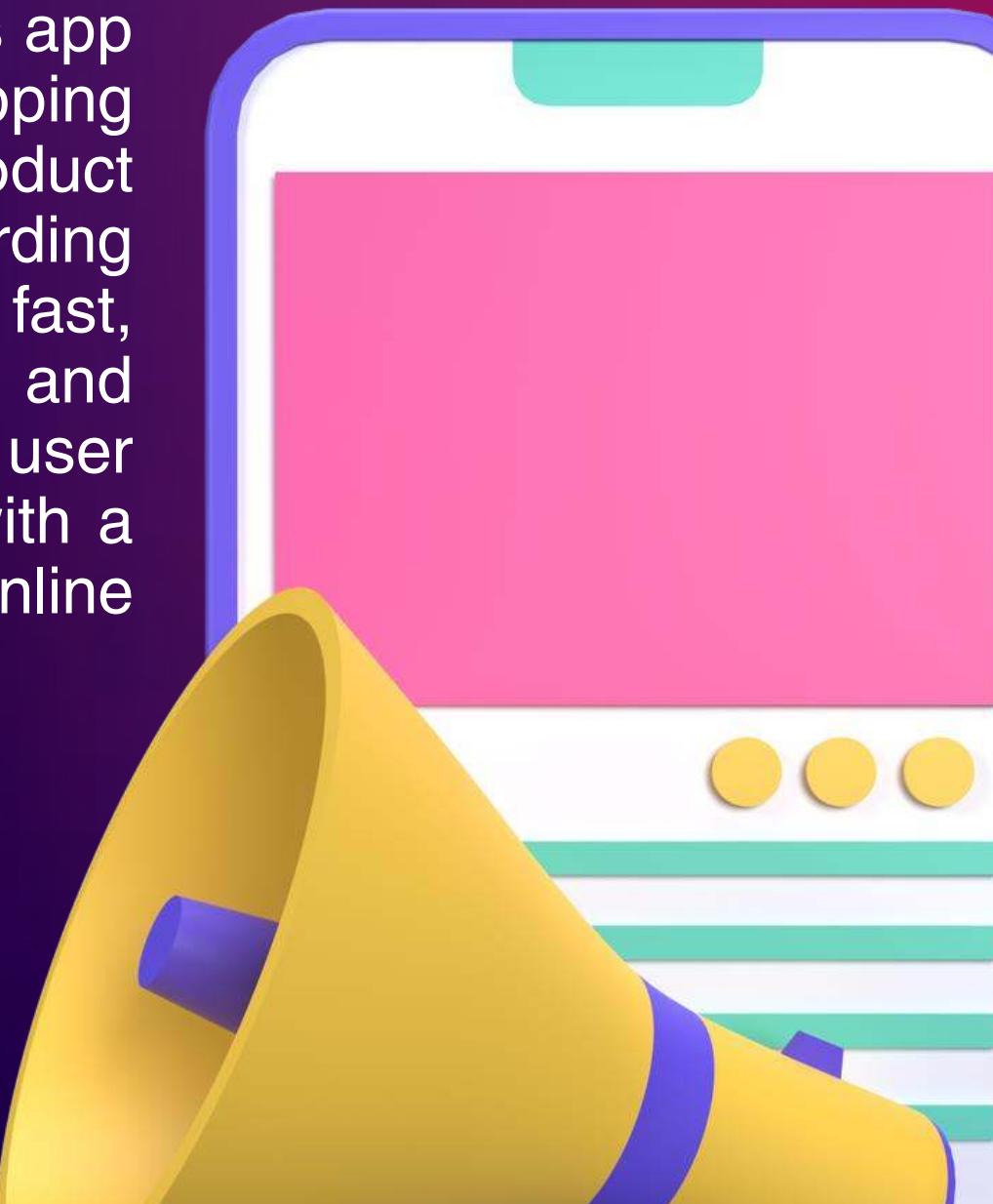
- Introduction
- Analysis
- Design
- Implementation
- Conclusion

/user



INTRODUCTION

Welcome to our presentation on the E-commerce App project. This app is designed to deliver a seamless and enjoyable online shopping experience. With features like personalized home pages, product search, shopping cart, user profiles, and a guided onboarding experience, our e-commerce platform aims to make shopping easy, fast, and secure. Built using Flutter for cross-platform compatibility and integrated with Firebase Authentication and Firestore for user management and data storage, this app combines functionality with a clean, modern design that caters to the needs of today's online shoppers.





ANALYSIS

/user



PROBLEM STATEMENT

- Users require a platform to browse, search, and purchase products seamlessly.
- E-commerce apps must ensure efficient navigation, personalization, and Validation.



OBJECTIVES

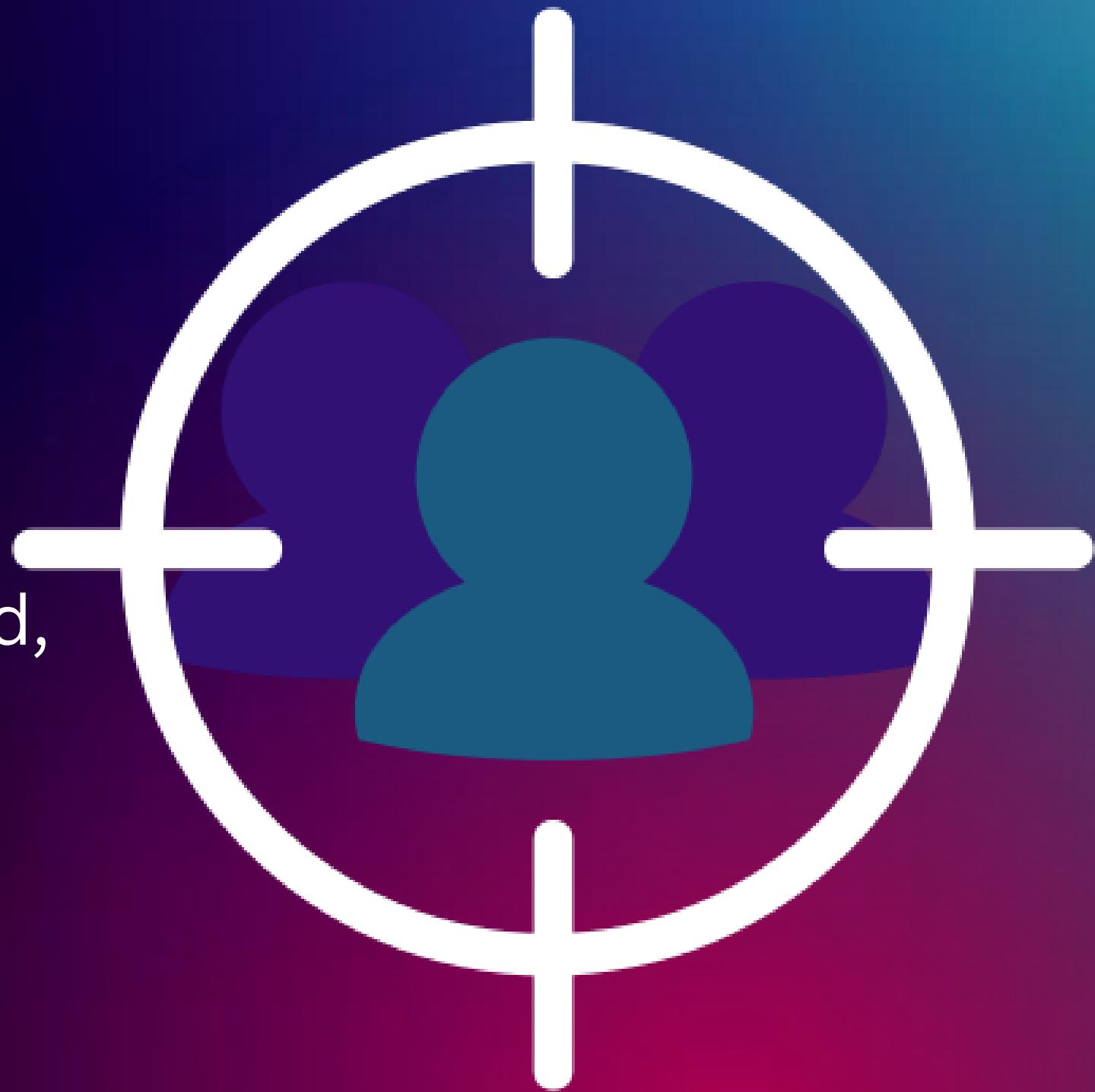
- Provide a user-friendly app with essential e-commerce functionalities.
- Ensure secure user authentication and seamless data interaction through APIs.

/user

TRAGET AUDIENCE

Online shoppers who value convenience, speed, and security in mobile commerce.

/user





FUNCTIONAL REQUIREMENTS

User Management
Core Features

USER MANAGEMENT CORE FEATURES

-
- HOME PAGE
 - SEARCH
 - CART
 - PROFILE PAGE

ONBOARDING EXPERIENCE VALIDATION

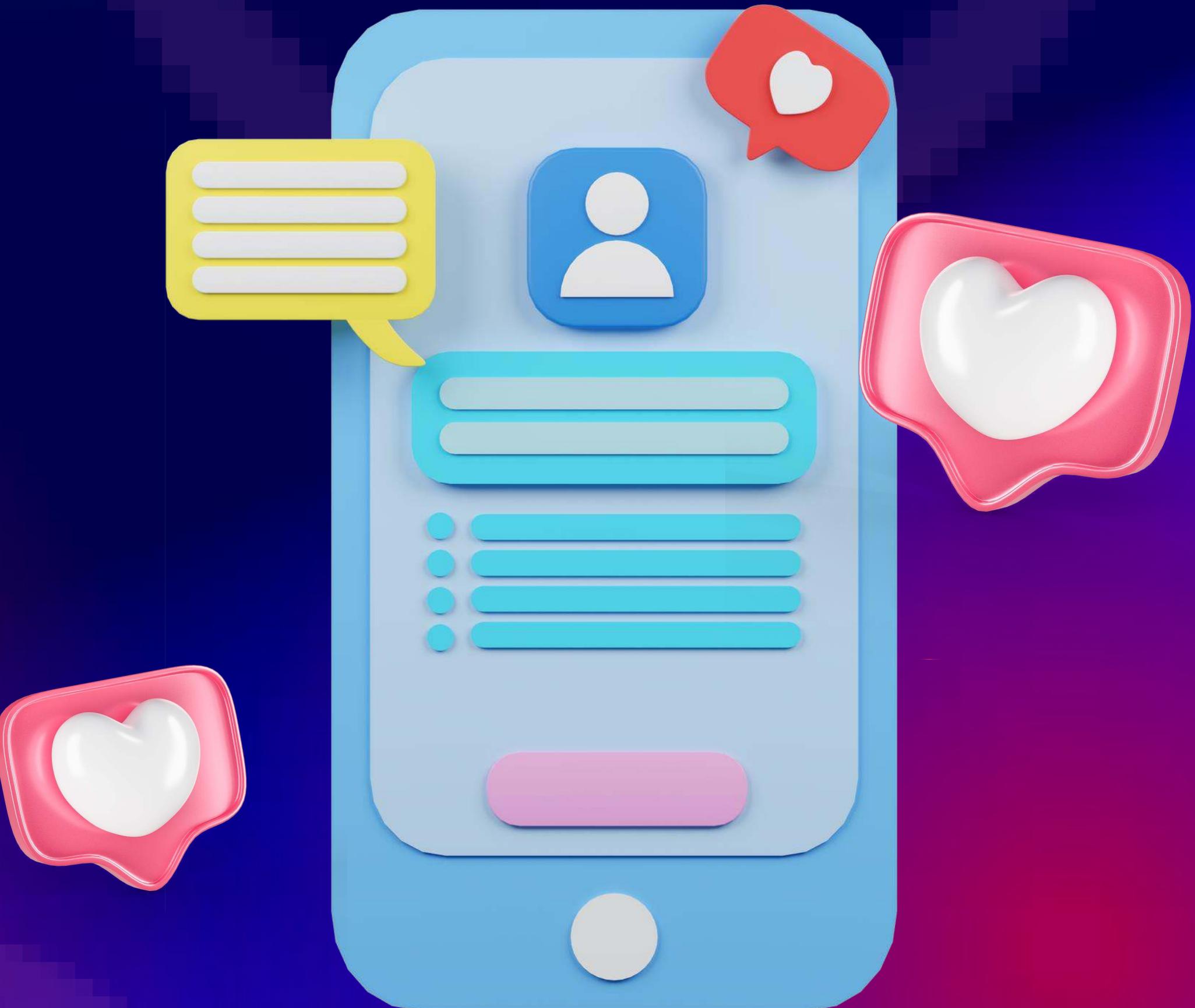


TECHNICAL REQUIREMENTS

- Frontend
Flutter framework for cross-platform compatibility.
- Backend:
Firebase Authentication for user management.
Cloud Firestore for real-time database management.
- APIs:
Fetching products and categories from the provided REST API .
/user

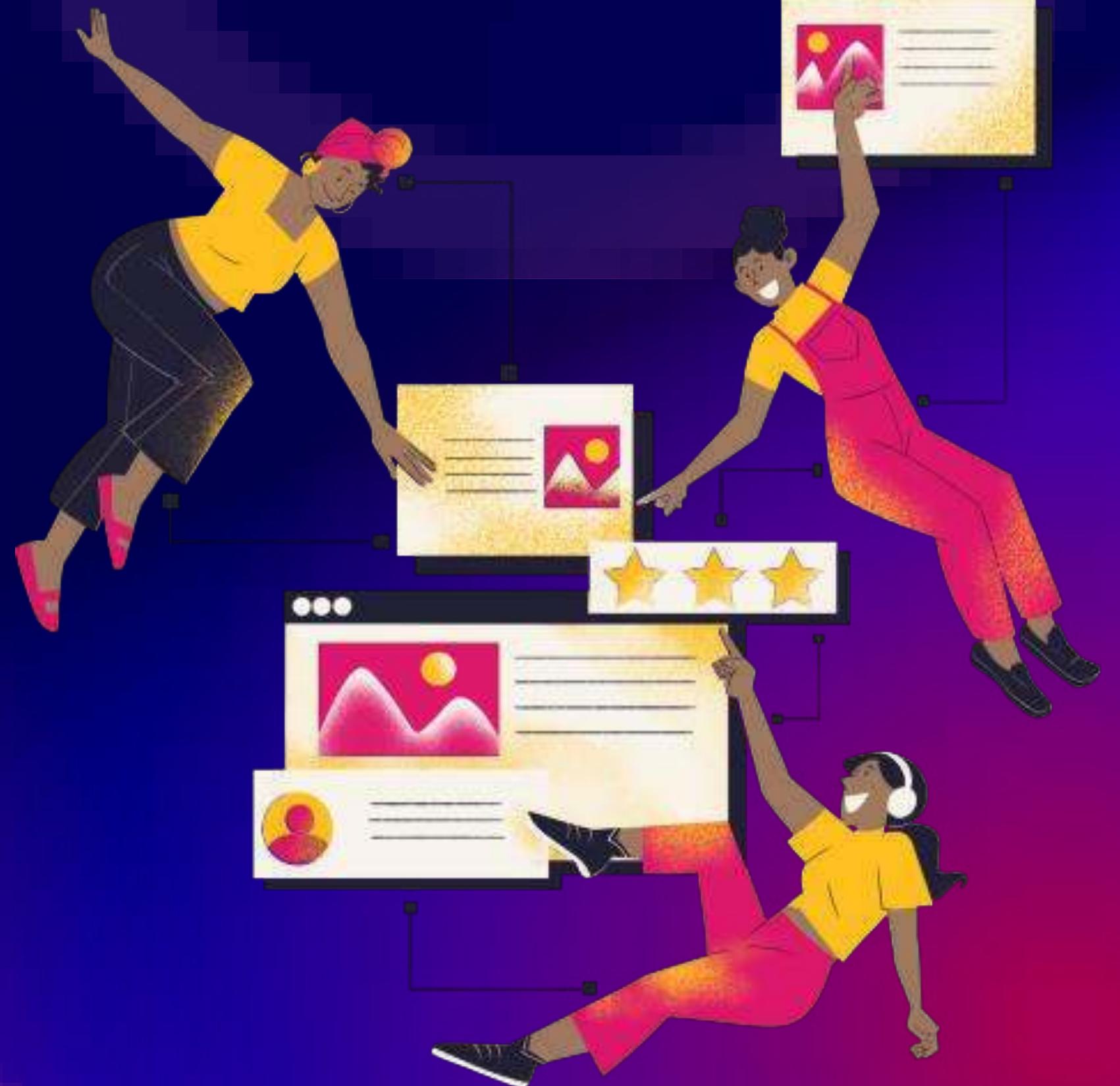


DESIGN



UI/UX GOALS

- EASE OF USE
- AESTHETIC APPEAL
- CONSISTENCY

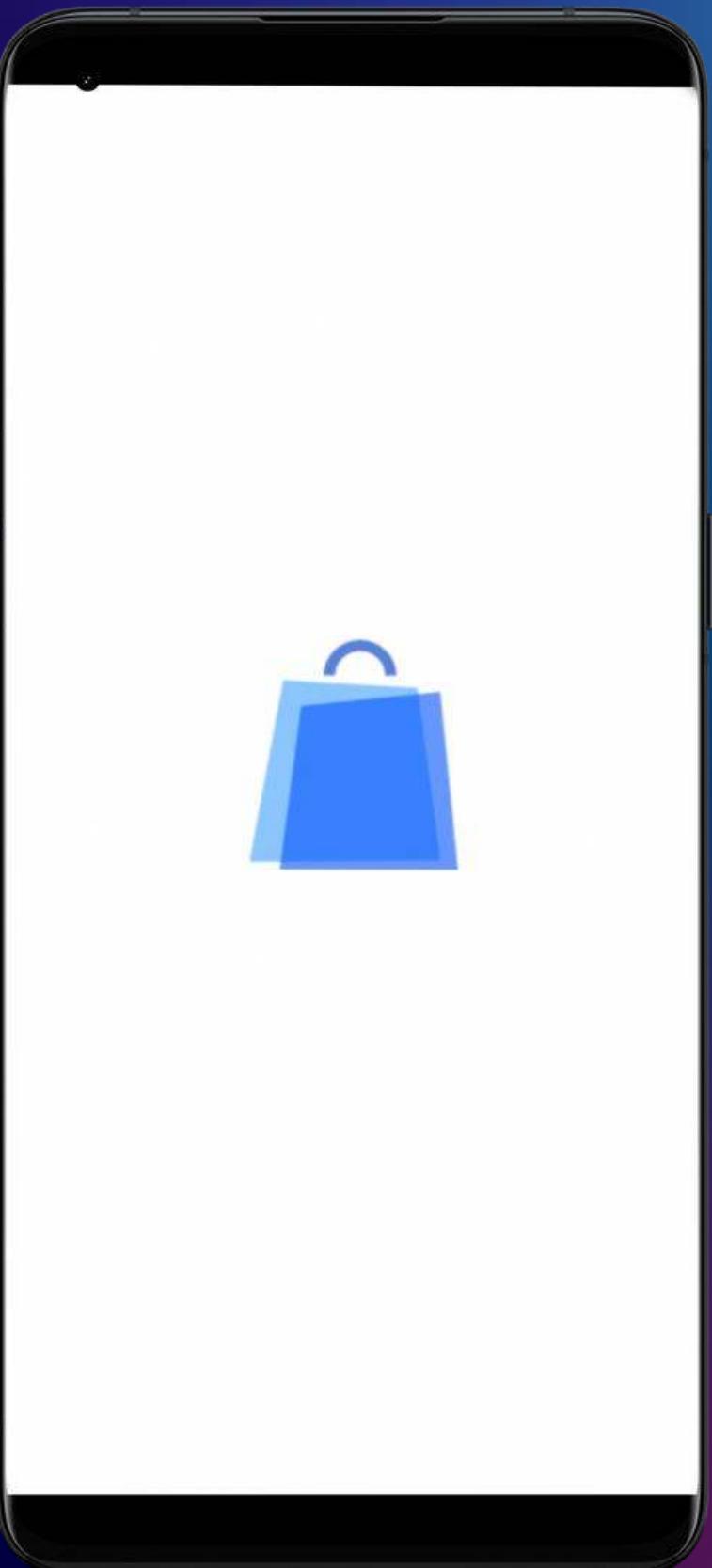




SCREENS

- Splash Screen:
App logo with a short fade-in animation.

/user

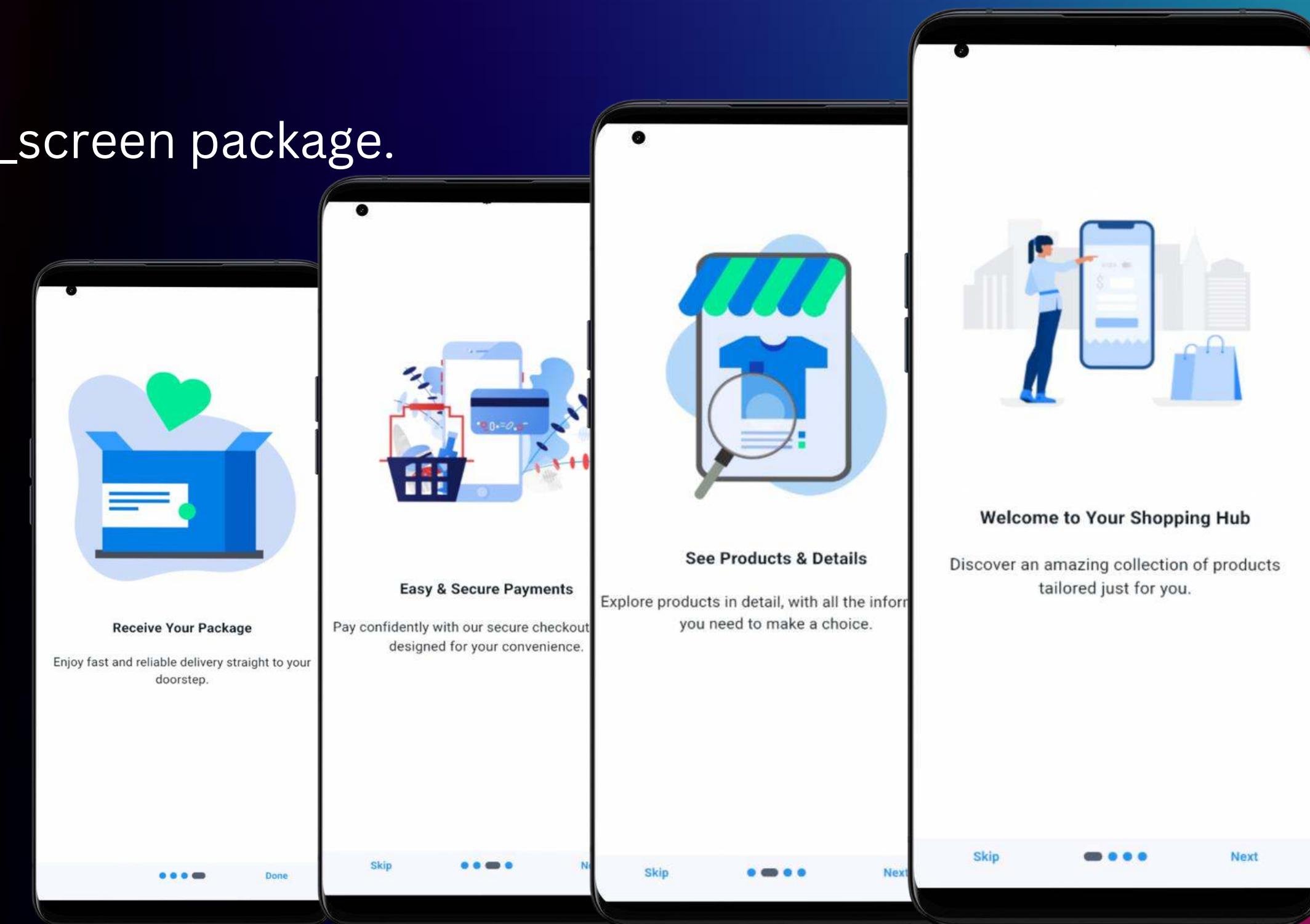




ONBOARDING SCREENS

- Carousel views using the introduction_screen package.
- Feature highlights

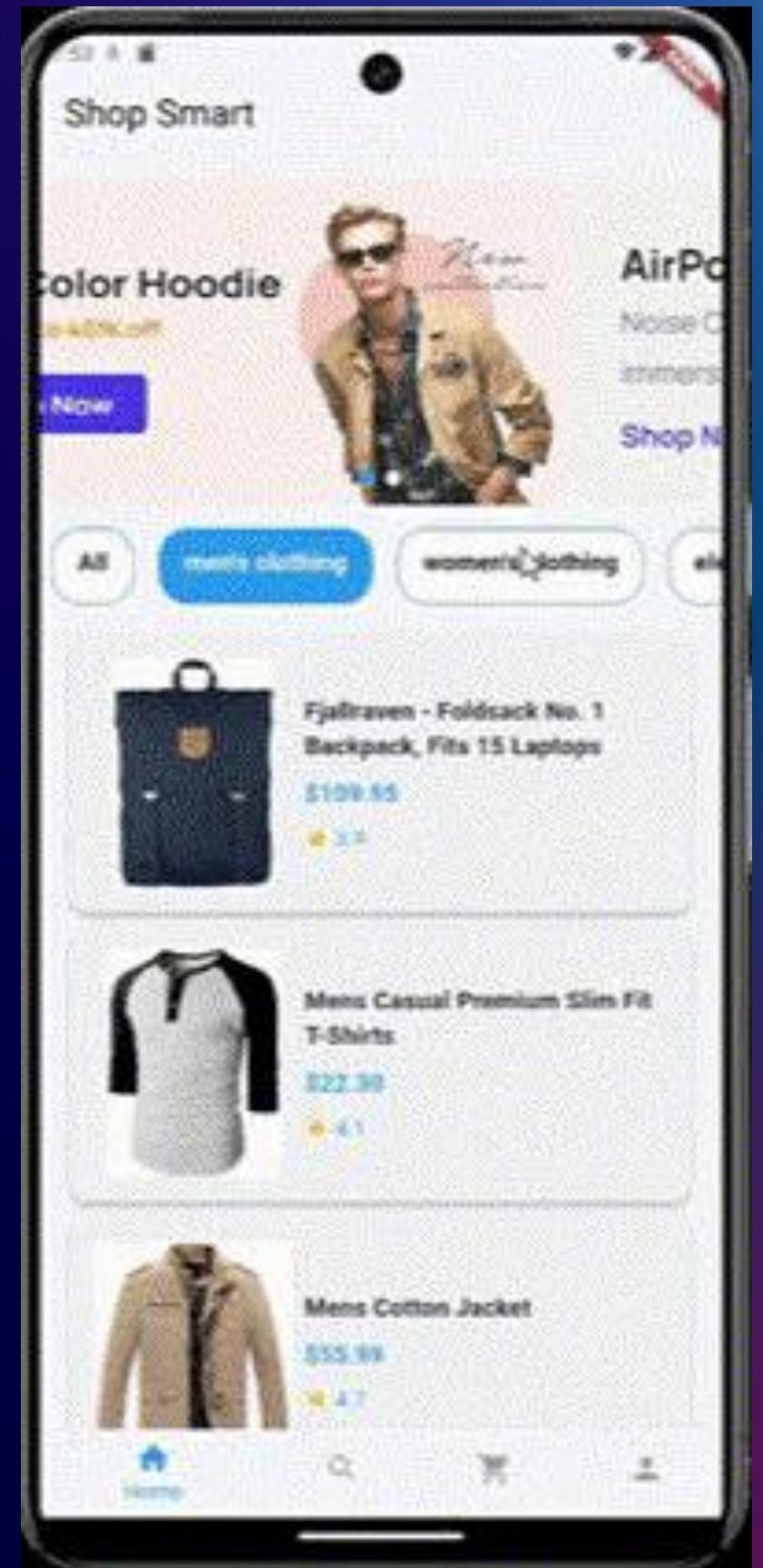
Easy shopping
Product details
secure Payment
receive Package



HOME PAGE

- Featured products section with scrollable categories.
- categories at the top for quick access.
- shimmer and card swiper packages

/user

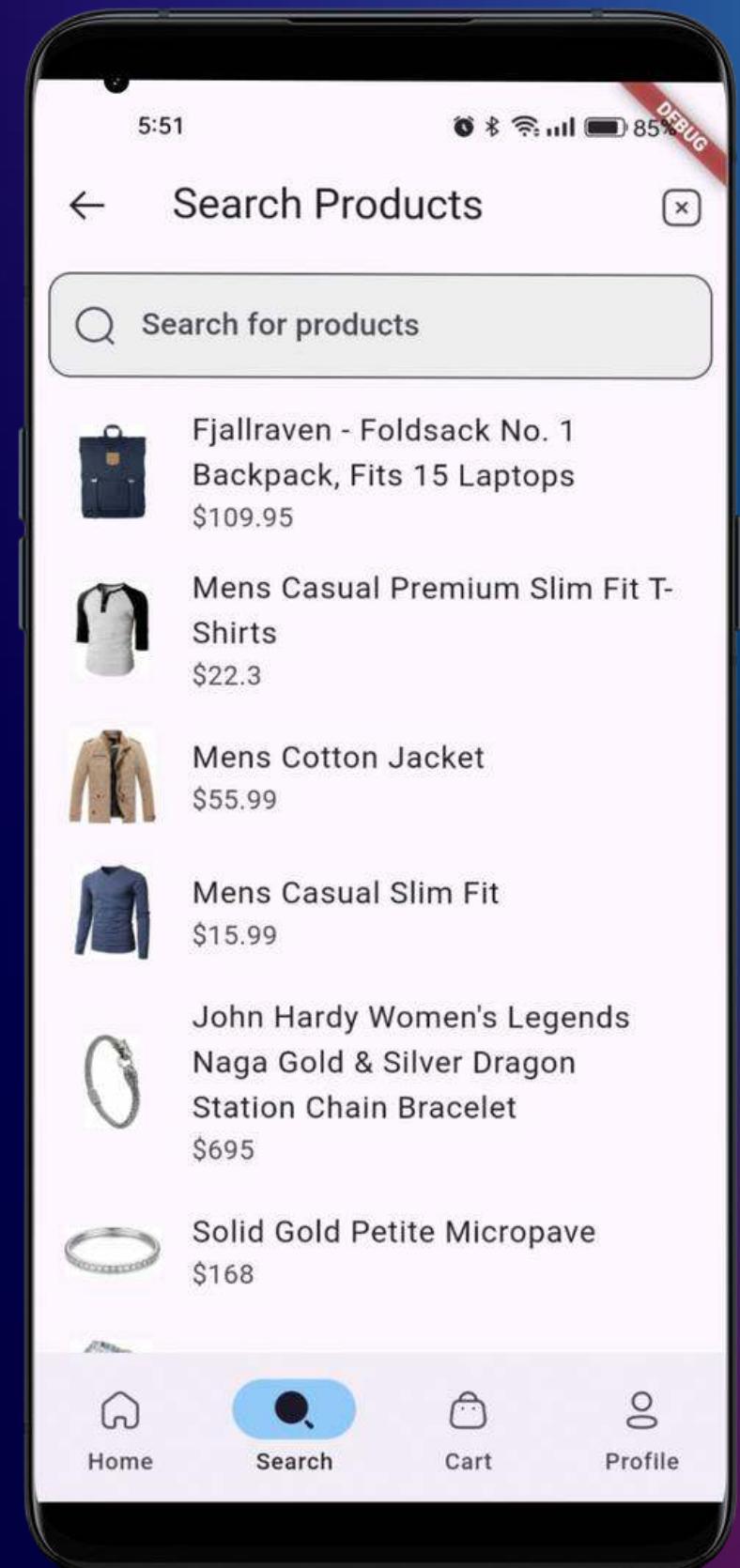




SEARCH SCREEN

- Input field for keyword-based search.
- List of matching products dynamically populated.

/user

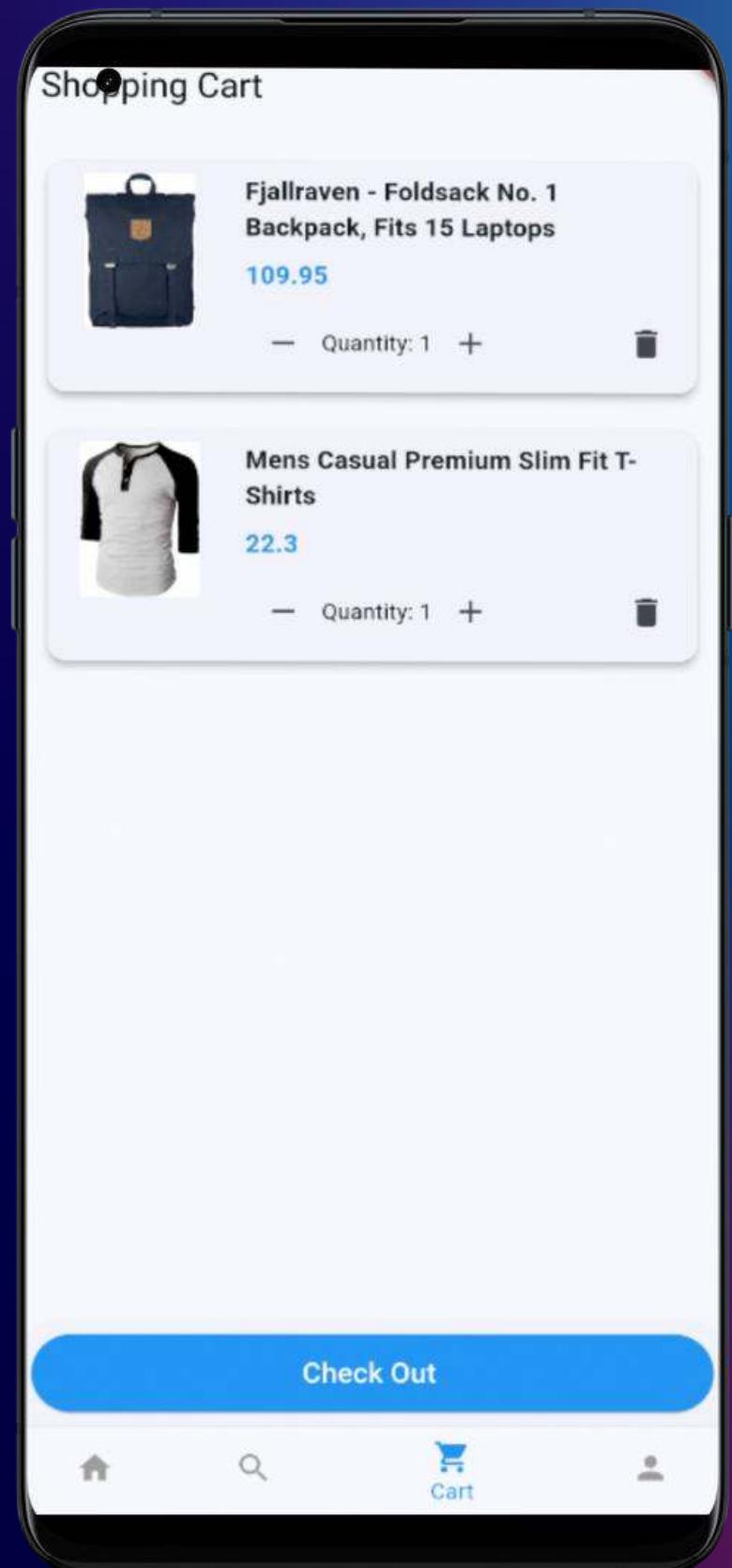




CART PAGE

- List view of selected items with total cost displayed.
- Option to update quantity or remove items.

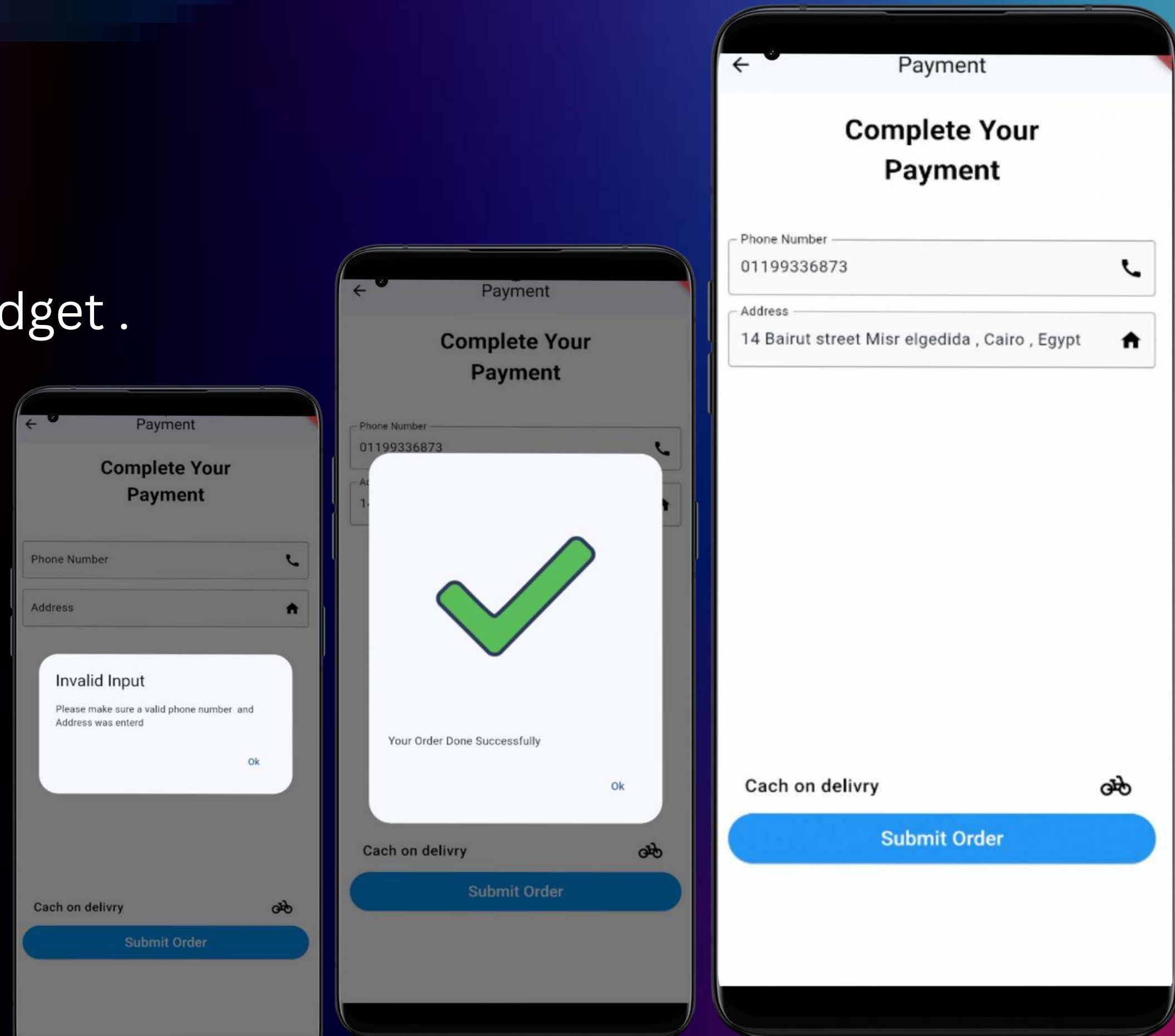
/user





CHECKOUT

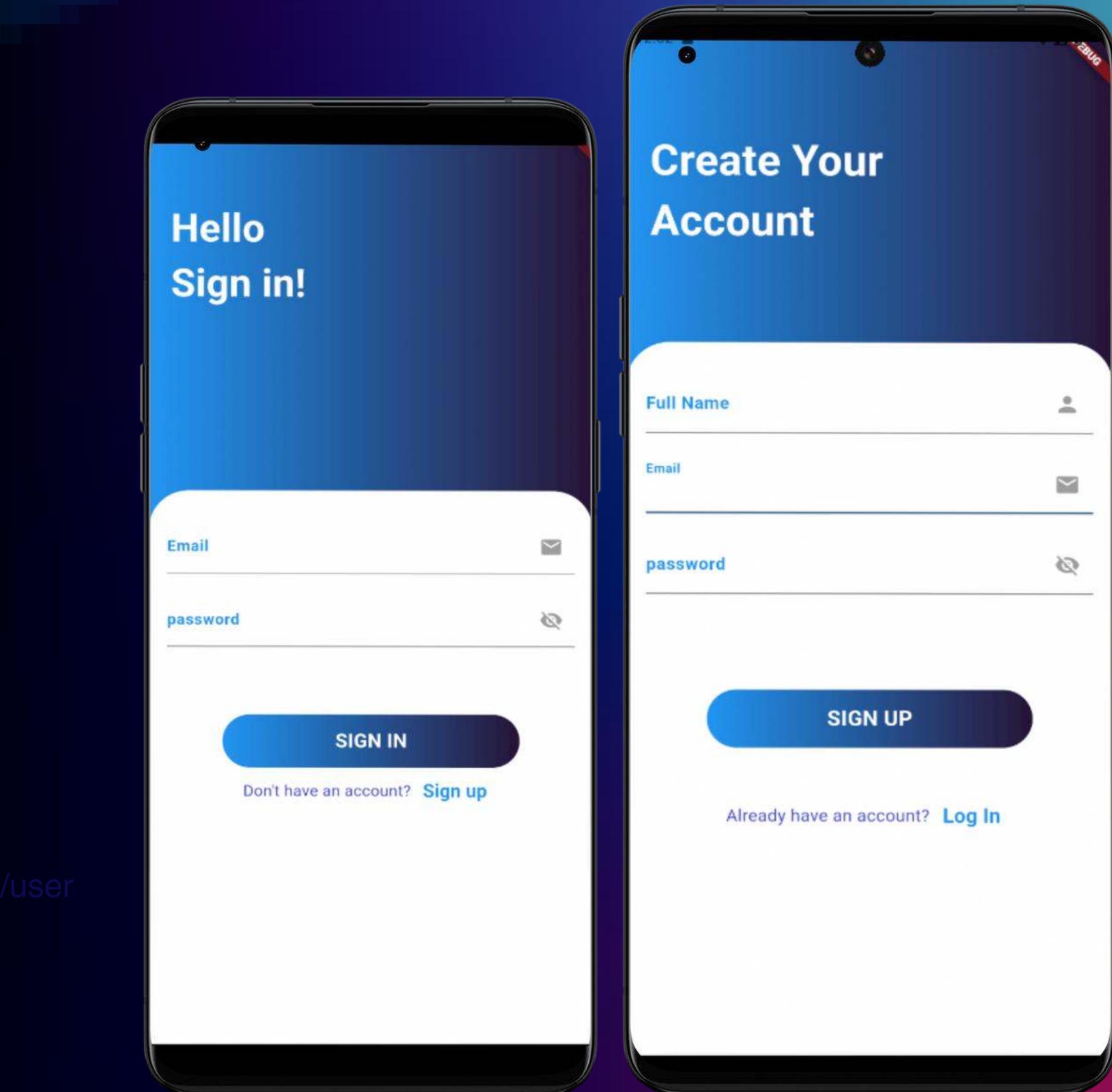
- Fast Checkout with successful widget .





PROFILE PAGE:

- User details (name, email).
- Navigation for settings and logout.



/user



STYLE GUIDE

- COLOR PALETTE



- TYPOGRAPHY

ROBOTO REGULAR WEIGHT MEDIUM SIZE

- ICONS

Underlined
text

- LAYOUTS

Grid Structure
space around
card:floating appearance





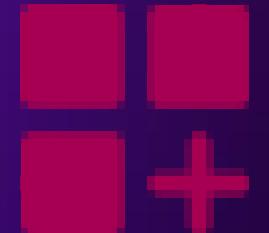
IMPLEMENTATION



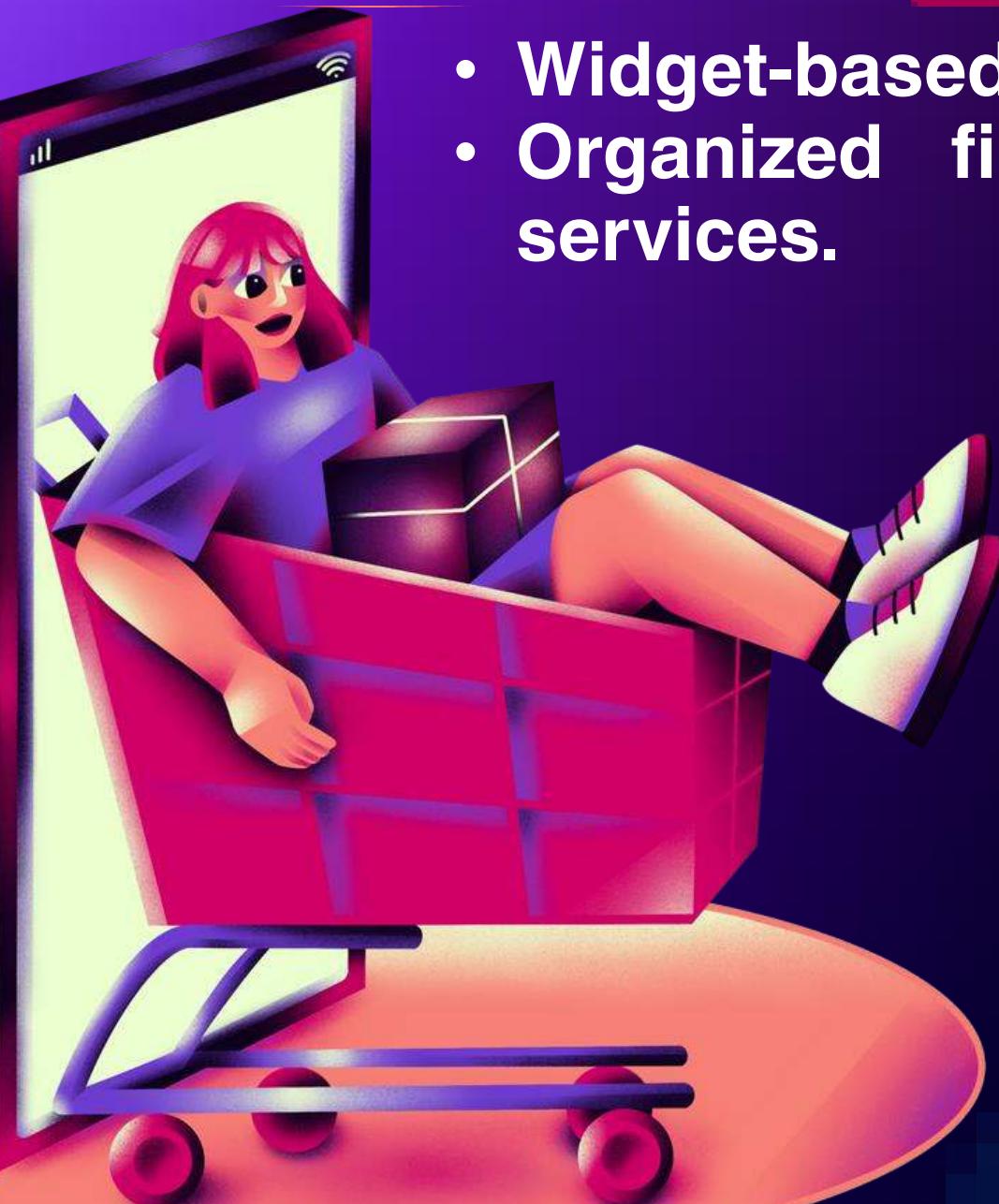


ARCHITECTURE

FRONTEND



- Widget-based structure for reusability.
- Organized files: screens, widgets, models, services.



BACKEND



- Firebase for authentication and data storage.
- REST API integration for product management.



KEY FEATURES

USER AUTHENTICATION

- user authentication
- validation

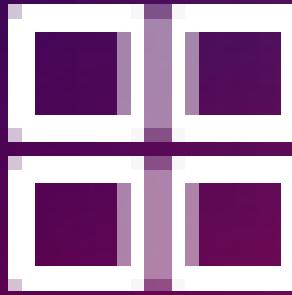
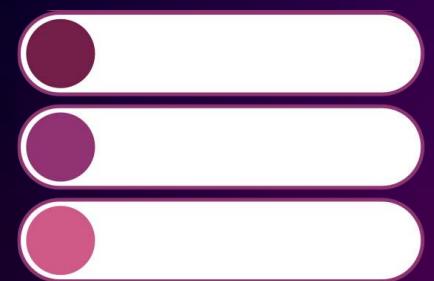


CART MANAGEMENT

- Stateful widget to manage cart items.
- Real-time updates based on changes.

PRODUCT DISPLAY

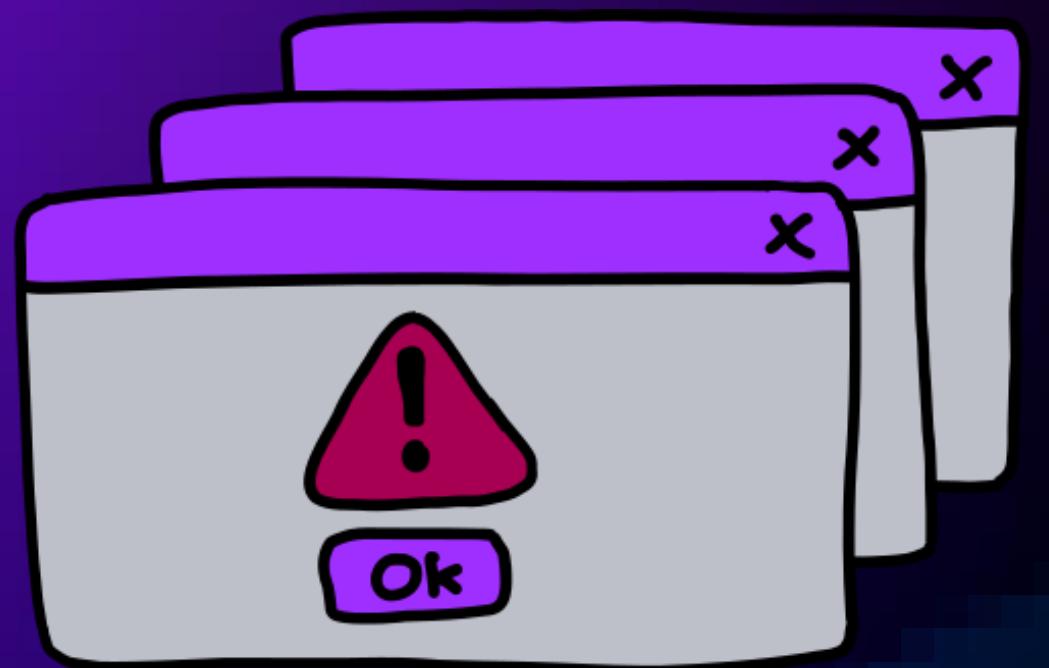
- dynamic loading via API
- ListView and GridView for layouts.



KEY FEATURES

ERROR HANDLING

- Graceful error messages for API failures.
- Retry mechanisms for failed API requests.



PACKAGES

- different flutter package where used





TECHNOLOGIES & DEPENDENCIES

FLUTTER

- Framework for cross-platform development.

FIREBASE

- `firebase_auth`
- `cloud_firestore`

PACKAGES

- `introduction_screen`
- `flutter_launcher_icons`
- `firebase_auth`
- `cloud_firestore`
- `card_swiper`
- `cupertino_icons`
- `iconly`
- `card_swiper`
- `cupertino_icons`
- `fancy_shimmer_image`





CODE SNIPPETS

A screenshot of a Microsoft Visual Studio Code (VS Code) interface displaying a Flutter application project. The project structure is visible in the Explorer sidebar, showing files like `banner1.png`, `banner2.png`, `pubspec.yaml`, `main_screen.dart`, `services.dart`, and `splash.dart`. The main editor tab shows Dart code for a service class named `EcommerceServices`. The code uses `http` to fetch products and users from a fake store API. The code is as follows:

```
import 'dart:convert';
import 'package:http/http.dart' as http;

class EcommerceServices {
  final String BASE_URL = "https://fakestoreapi.com";

  Future<List<dynamic>> fetchProducts() async {
    try {
      final response = await http.get(Uri.parse('${BASE_URL}/products'));
      if (response.statusCode == 200) {
        return jsonDecode(response.body) as List<dynamic>;
      } else {
        throw Exception('Failed to load');
      }
    } catch (e) {
      throw Exception(e.toString());
    }
  }

  Future<List<dynamic>> fetchUsers() async {
    try {
      final response = await http.get(Uri.parse('${BASE_URL}/users'));
      if (response.statusCode == 200) {
        return jsonDecode(response.body) as List<dynamic>;
      } else {
        throw Exception('Failed to load');
      }
    } catch (e) {
      throw Exception(e.toString());
    }
  }
}
```

The status bar at the bottom indicates the code is in Dart mode for a Pixel 8 Pro API 33 emulator, with the file being `services.dart`.



API INTEGRATION

```
Terminal Help ← → depi_project_2
[er2.png banner1.png pubspec.yaml 1, M main_screen.dart services.dart X onboard.dart splash.dart 2, M
services > services.dart > EcommerceServices > fetchUsers
import 'dart:convert';

import 'package:http/http.dart' as http;

class EcommerceServices {
  final String BASE_URL = "https://fakestoreapi.com";

  Future<List<dynamic>> fetchProducts() async {
    try {
      final response = await http.get(Uri.parse('${BASE_URL}/products'));
      if (response.statusCode == 200) {
        return jsonDecode(response.body) as List<dynamic>;
      } else {
        throw Exception('Failed to load');
      }
    } catch (e) {
      throw Exception(e.toString());
    }
  }

  Future<List<dynamic>> fetchUsers() async {
    try {
      final response = await http.get(Uri.parse('${BASE_URL}/users'));
      if (response.statusCode == 200) {
        return jsonDecode(response.body) as List<dynamic>;
      } else {
        throw Exception('Failed to load');
      }
    } catch (e) {
      throw Exception(e.toString());
    }
  }
}
```



HOME PAGE

The screenshot shows a Flutter project named "DEPI_PROJECT_2" open in a code editor. The main file displayed is `main_screen.dart`. The code implements a `StatefulWidget` for the home screen, featuring a `PageView` with a `NeverScrollableScrollPhysics` and a `PageController` initialized at index 0. It also includes methods for fetching products and building the UI with a `Scaffold` and a `BottomNavigationBar`.

```
import 'package:ecommerce_app/screens/pages/home_page.dart';
import 'package:ecommerce_app/screens/pages/search_page.dart';
import 'package:ecommerce_app/screens/pages/shopping_cart.dart';
import 'package:ecommerce_app/screens/pages/profile_page.dart';
import 'package:ecommerce_app/screens/widgets/app_constants.dart';
import 'package:ecommerce_app/services/services.dart';
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';

class MainScreen extends StatefulWidget {
  class MainScreenState extends State<MainScreen> {
    final PageController _pageController = PageController(initialPage: 0);
    int _selectedIndex = 0;
    List<dynamic> _products = [];

    @override
    void initState() {
      fetchProducts();
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        bottomNavigationBar: BottomNavigationBar(),
        body: PageView(
          physics: const NeverScrollableScrollPhysics(),
          controller: _pageController,
          children: [
            // PageView
            // Scaffold
          ],
        );
      }
    }
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    fetchProducts();
  }

  void fetchProducts() async {
    // Fetch products logic
  }
}
```

CHALLENGES

MANAGING API ERRORS.

- Solution: Implemented error handling and retry mechanisms.

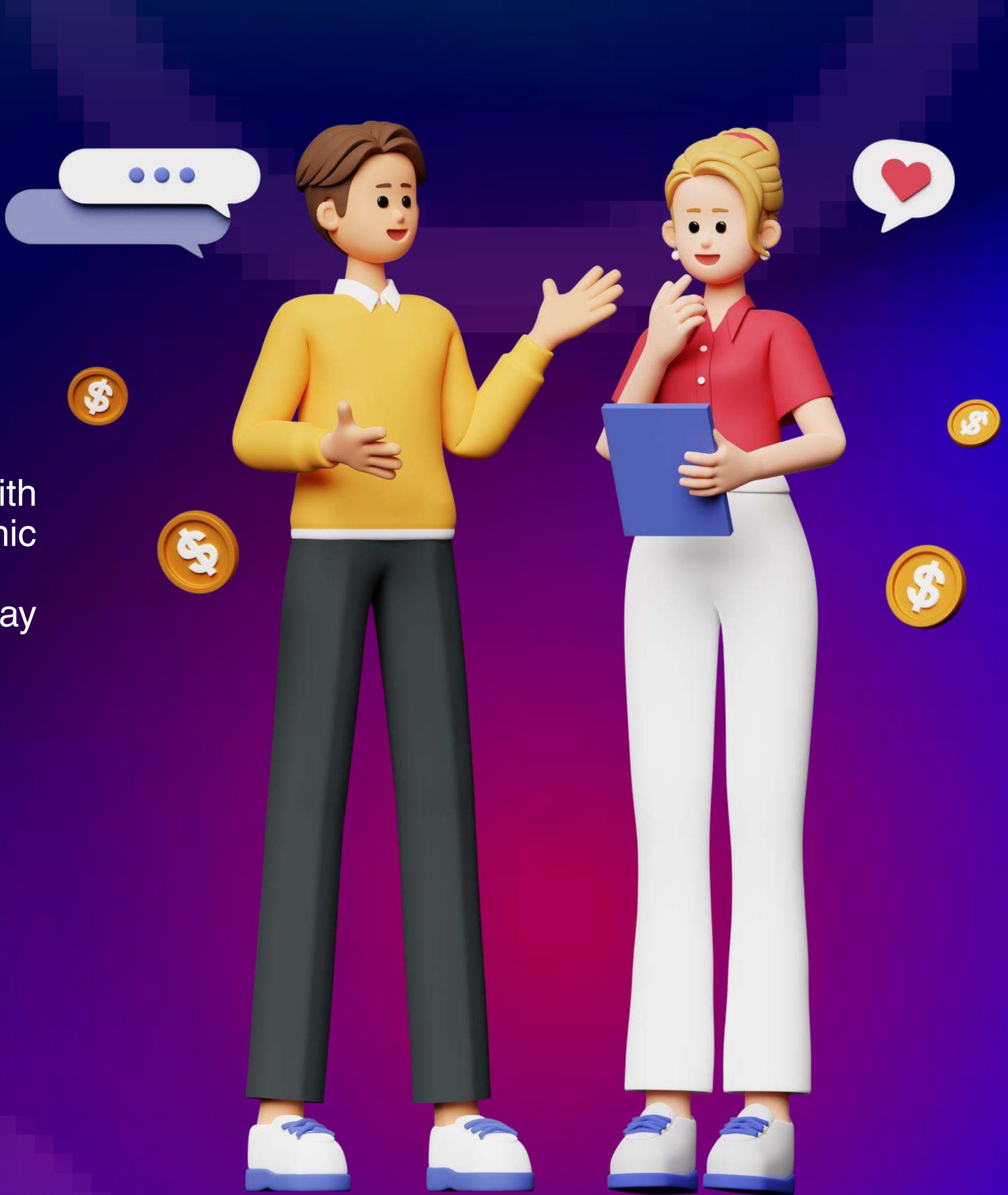
MANAGING API ERRORS.

- Solution: Used Flutter's MediaQuery for adaptive layouts.



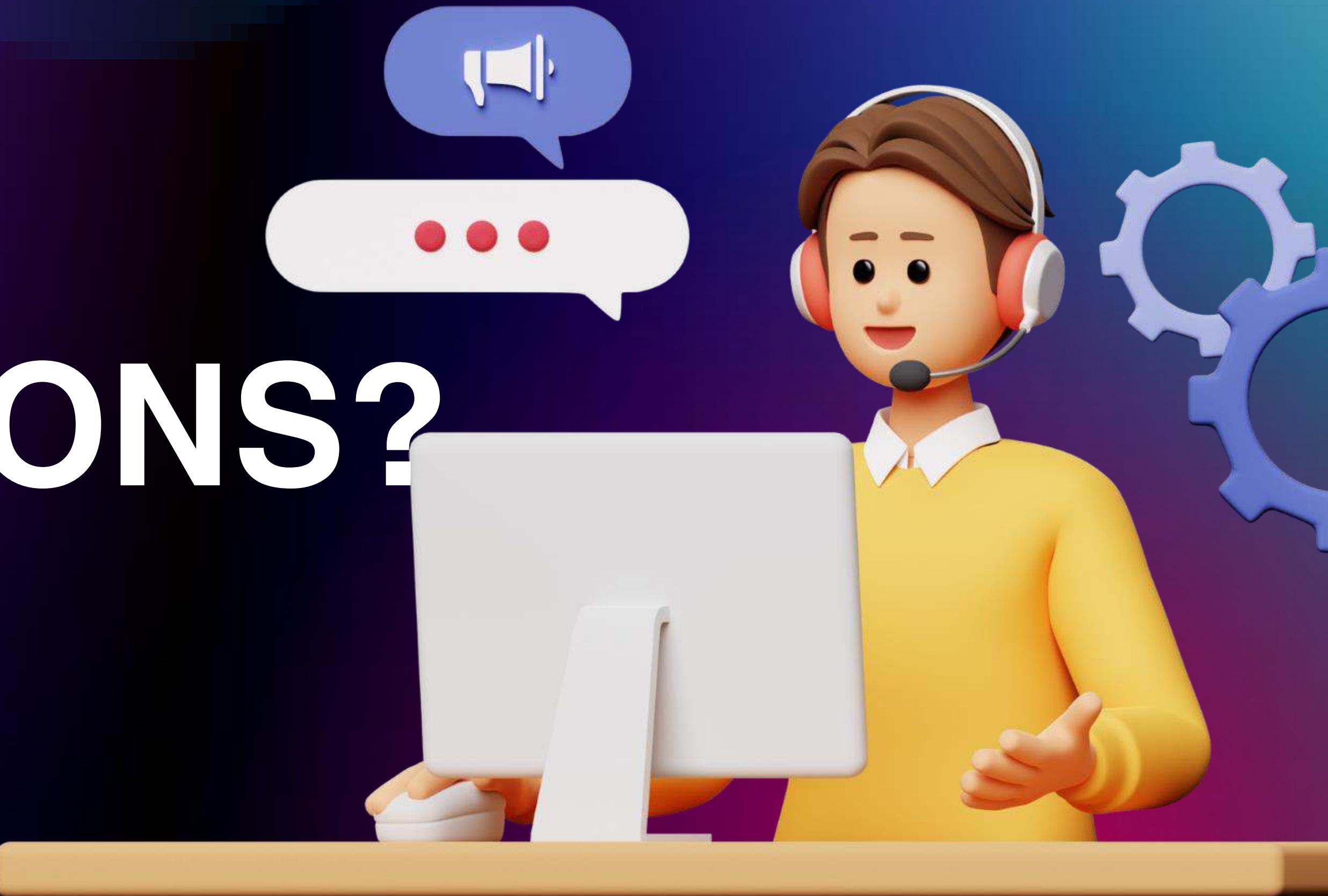
CONCLUSION

- The app provides a robust e-commerce solution with essential features like user authentication, dynamic product listing, and cart management.
- Future enhancements can include payment gateway integration, order tracking, and notifications.





QUESTIONS?





Thank you

