Tutorial: Building a Flutter E-Commerce App (Part 1: Setup, Landing Page, and Home Page)

Overview

In this tutorial series, you'll build a complete Flutter-based E-commerce app. We'll approach this step-by-step in waves, starting today with the basic project setup, landing page, and home page. I'll provide code snippets of each page, but the code itself won't be the complete version of the pages, which, if you are looking for that, can be found on my GitHub.

GitHub Link: https://github.com/ChristianSaenz/mini-project-comp375.git

---

Step 1: Setting Up Flutter in Android Studio

Before we dive into coding, let's ensure your environment is ready.

Prerequisites

- **Android Studio**: Download from Android Studio Official Site
- **Flutter SDK**: Get the latest SDK from the Flutter official website

Installing Flutter and Dart Plugins in Android Studio

1. Open **Android Studio**.
2. Navigate to `File > Settings` on Windows/Linux or `Android Studio > Preferences` on Mac.
3. Click on **Plugins**.
4. Search for and install:
    - **Flutter**
    - **Dart**

Restart Android Studio to apply changes.

Verify Flutter Installation

Open a terminal or command prompt and type:

flutter doctor

Resolve any issues indicated by `flutter doctor` output before proceeding. If the command `flutter doctor` doesn't work, it's likely because the Flutter SDK path is missing from your environmental variables. For troubleshooting and additional instructions on setting environmental variables, check [Flutter's official installation guide](#).

Step 2: Create a New Flutter Project

1. In Android Studio, select `File > New > New Flutter Project`.
2. Choose `Flutter` as the project type.
3. Enter your project's name (e.g., `ecommerce_app`) and location.
4. Click **Finish**.

Step 3: Building the Landing Page

This app opens with a simple landing page featuring a logo, background, and navigation button to the home page.

Project Structure Setup

Inside `lib`, create:

- `pages` folder (landing and home pages will go here)
- `assets/img` folder within the `lib` folder for images
  - You can also host this outside the lib folder if you would like

Update `pubspec.yaml` to include:

- Any Images that you want to provide must be added to the assets like so

```
flutter:
  assets:
    - lib/assets/img/logo.png
    - lib/assets/img/background.png
```

Landing Page Code (`landing_page.dart`)

```
import 'package:flutter/material.dart';
import 'package:timeless_toys/widgets/main_screen.dart';
import 'home_page.dart';

class LandingPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
// Overlay of images on top of each other
      body: Stack(
        children: [
          // Background image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/background.webp"),
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Column layout for logo and navigation button
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Image.asset('assets/logo.webp', height: 300), // App logo
              const SizedBox(height: 30),
              ElevatedButton(
                onPressed: () {
                  Navigator.pushReplacement(context,
                    MaterialPageRoute(builder: (context) => const MainScreen()));
                },
                child: const Text("Get Started"),
                style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold, color:
Colors.white),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

Step 3: Understanding Key Widgets

Here's a breakdown of the primary widgets used in the landing page:

- `Scaffold`: Provides a basic layout structure for your page (app bars, floating action buttons, etc.).
- **Stack**: Allows you to overlay widgets. Useful for backgrounds or layered UI elements.
- **Container**: A versatile widget for styling (e.g., backgrounds, padding).
  - **DecorationImage**: For placing and fitting images as backgrounds.
- **Column**: Organizes widgets vertically.
- **Image.asset**: Loads images from your project's assets.
- **ElevatedButton**: A styled button for user interactions.
- **Navigator.pushReplacement**: This is used to navigate to a new page and remove the current page from the navigation stack.

Step 4: Installing Required Packages

Our home page uses the **Carousel Slider** and **Smooth Page Indicator**. Add the following dependencies to your `pubspec.yaml` file:

```
dependencies:

  carousel_slider: ^4.2.1

  smooth_page_indicator: ^1.1.0
```

Run the command below in your terminal to fetch the dependencies:

- flutter pub get

Why We Use These Packages:

- **carousel_slider**: This package provides a simple way to create image sliders with auto-play, transitions, and customization.
- **smooth_page_indicator**: Helps add elegant page indicators to the carousel to improve the user experience.

Where to Find Flutter Packages:

If you're looking for additional Flutter packages, browse the official Dart Pub Repository. This website hosts thousands of open-source Flutter packages for UI components, networking, database management, animations, and more.

Step 4: Home Page Code (`home_page.dart`)

Below is the full implementation of the home page:

```dart
import 'package:flutter/material.dart';
import 'package:carousel_slider/carousel_slider.dart';
import 'package:smooth_page_indicator/smooth_page_indicator.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int activeIndex = 0;
  final List<String> carouselImages = [
    "assets/banner1.webp",
    "assets/banner2.jpg",
    "assets/banner3.jpg",
  ];

  final List<Map<String, String>> toys = [
    {"name": "Teddy Bear", "image": "assets/teddy.webp"},
    {"name": "Race Car", "image": "assets/toy_car.jpg"},
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Timeless Toys"),
        backgroundColor: Colors.blueAccent,
      ),
      body: Column(
        children: [
          // Image carousel
          CarouselSlider.builder(
            itemCount: carouselImages.length,
            options: CarouselOptions(autoPlay: true),
            itemBuilder: (context, index, realIndex) => Image.asset(carouselImages[index]),
          ),
          // Page indicator
          AnimatedSmoothIndicator(
            activeIndex: activeIndex,
            count: carouselImages.length,
            effect: ExpandingDotsEffect(dotWidth: 10, dotHeight: 10),
          ),
          // Product grid
          Expanded(
            child: GridView.builder(
              gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: 2,
              ),
              itemCount: toys.length,
              itemBuilder: (context, index) {
                return Card(
                  child: Column(
                    children: [
                      Image.asset(toys[index]["image"]!),
                      Text(toys[index]["name"]!),
                    ],
                  ),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Step 4: Key Concepts Used in the Home Page

- **Stateful Widget (`HomePage`)**: Allows dynamic changes, such as updating the carousel indicator.
- **`super.key` in Constructors**: Ensures widget uniqueness and proper state management when Flutter rebuilds widgets.
- **AppBar**: Provides a header with a title for the page.
- **CarouselSlider**: Enables a horizontally scrolling image carousel.
- **Stack**: Overlays elements, used here for placing the carousel and its indicator.
- **AnimatedSmoothIndicator**: Displays page indicators for the carousel.
- **GridView.builder**: Dynamically displays items in a structured grid layout.
- **GestureDetector**: Detects taps on the product cards.
- **SnackBar**: Provides instant feedback when an item is clicked.

Step 5 (MainScreen): The Main Screen (`main_screen.dart`) serves as the navigation hub for our Flutter E-Commerce app. Unlike the Landing Page and Home Page, which display content directly, this screen manages navigation between multiple pages using a Bottom Navigation Bar.

Step 5: Main Screen Code Overview

```
import 'package:flutter/material.dart';

import 'package:timeless_toys/pages/account_page.dart';

import 'package:timeless_toys/pages/home_page.dart';

import 'package:timeless_toys/pages/shop_page.dart';


class MainScreen extends StatefulWidget {

  const MainScreen({super.key});


  @override
  _MainScreenState createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  int _selectedIndex = 0; // Tracks the selected tab

  // List of pages managed by the navigation bar
  final List<Widget> _screens = [
    HomePage(),  // Home Page
    ShopPage(),  // Shop Page
    AccountPage() // Account Page
  ];

  // Function that updates the selected index when a tab is tapped
  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: _screens[_selectedIndex], // Displays the selected page
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
        items: const [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.shopping_cart), label: 'Shop'),
          BottomNavigationBarItem(icon: Icon(Icons.person), label: 'Account'),
        ],
      ),
    );
  }
}
```

Step 5: How the Main Screen Works Compared to Other Screens?

- The Main Screen is a StatefulWidget because it needs to track which tab is selected and update the UI accordingly.
- Unlike the Landing Page, which only serves as an introduction, or the Home Page, which displays content, the Main Screen provides persistent navigation, allowing users to switch between different sections of the app seamlessly.

Logic Behind the Code

1. **Tracking the Active Tab:**
   - The `_selectedIndex` variable stores the currently selected tab.
   - It starts at `0`, meaning the Home Page is displayed initially.
2. **Handling Navigation:**
   - `_screens` is a list storing the three main pages (Home, Shop, Account).
   - The `Scaffold` widget sets `body` to `_screens[_selectedIndex]`, meaning it displays whichever page is currently selected.
3. **Bottom Navigation Bar Logic:**

    - The `BottomNavigationBar` widget provides a menu with three tabs.
    - `currentIndex: _selectedIndex` ensures that the correct tab is highlighted.
    - When a tab is tapped, `_onItemTapped` updates `_selectedIndex` using `setState()`, causing the UI to rebuild with the new page.
4. **Stateful Behavior:**
    - Since `_selectedIndex` is stored in the widget's state, switching between tabs doesn't rebuild the entire app.
    - The `setState()` function ensures that only the relevant parts of the UI update.

Key Concepts Used in the Main Screen

- **Stateful Widget (`MainScreen`):** Allows dynamic tab selection.
- BottomNavigationBar: Provides persistent navigation at the bottom of the screen.
- **List of Screens (`_screens`):** Stores references to different app sections.
- **setState():** Updates the selected index when a tab is tapped, triggering a UI update.
- **Scaffold:** Wraps the page and includes a navigation bar.

Step 6 (Home Page): In this section, we'll focus on the **main entry point** of our Flutter E-Commerce app. The `main.dart` file is responsible for initializing the app and defining the app-wide theme and navigation structure.

Step 6: Main Page Code (`main.dart`)

```
import 'package:flutter/material.dart';
import 'package:timeless_toys/pages/landing_page.dart';
import 'package:timeless_toys/widgets/main_screen.dart';
```

```
void main() {
  runApp(const MyApp()); // Starts the Flutter application
}

class MyApp extends StatelessWidget {
  // The main application widget that acts as the root of the app.
  // 'super.key' ensures widget uniqueness and helps maintain state during hot reloads.
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo', // App title
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple), // Defines the
app-wide color scheme
        useMaterial3: true, // Enables Material 3 design system
      ),
      home: const LandingPage(), // Sets the initial screen when the app loads
    );
  }
}
```

Step 6: Key Concepts Used in the Main Page

- **`void main()`**: This is the entry point of the app. The `runApp()` function launches the `MyApp` widget.
- **Stateless Widget (`MyApp`)**: Represents the root widget of the application. It doesn't change state once built.
- **`super.key` in Constructors**: Helps Flutter track widget state properly, especially during hot reloads.
- **`MaterialApp`**: The core widget that wraps the entire application and provides app-wide configurations.
- **`title`**: The name of the application, used in debugging tools.
- **`ThemeData`**: Defines the overall theme and styling of the app.
- **`home: LandingPage()`**: Specifies the first screen to load when the app starts.