Giftcard project

This document will take us through the process of building and maintain the giftcard project

What we need for the application

* We need the administration page
* We need the client page

Some dependencies needed in the giftcard application

cloud\_firestore:  
firebase\_auth:  
flutter\_hooks:  
google\_fonts:  
google\_sign\_in:  
sign\_button:  
url\_strategy:  
url\_launcher:  
flutter\_svg:  
firebase\_core:  
flutter\_riverpod:

the theme details for the application

theme: ThemeData(  
 colorScheme: ColorScheme.fromSeed(seedColor: Colors.*red*),  
 textTheme: TextTheme(  
 displayLarge: const TextStyle(  
 fontSize: 20,  
 fontWeight: FontWeight.*bold*,  
 ),  
 titleLarge: GoogleFonts.*oswald*(  
 fontSize: 30,  
 fontStyle: FontStyle.italic,  
 ),  
 bodyMedium: GoogleFonts.*merriweather*(),  
 displaySmall: GoogleFonts.*pacifico*(),  
 ),  
 useMaterial3: true,  
),

Steps to create the Giftcard project

Declare the riverpod data for the application

how to register a user and saving it to database with stateNotifier riverpod flutter

ChatGPT

!

Using Riverpod's StateNotifier in Flutter to register a user and save it to a database involves several steps. Here's a basic outline of how you can achieve this:

Set up Riverpod: Ensure you have Riverpod set up in your Flutter project. You'll need to add the necessary dependencies to your pubspec.yaml file and import the required packages in your Dart files.

Create a User model: Define a class to represent a user. This class will contain properties such as username, email, password, etc.

Set up a Database Provider: Create a provider for your database instance. This provider will handle interactions with the database.

Create a StateNotifier: Build a StateNotifier class that will handle the state of user registration. This class will contain methods for registering a user and updating the state accordingly.

Register User: Implement the functionality to register a user. This typically involves validating user input, saving the user to the database, and updating the state based on the result.

Update UI: Update your UI to reflect the state of the user registration process. This might include showing loading indicators, error messages, or success notifications.

Here's a simplified example to illustrate these steps:

dart

Copy code

// 1. Set up Riverpod

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

// 2. Create a User model

class User {

final String username;

final String email;

final String password;

User({required this.username, required this.email, required this.password});

}

// 3. Set up a Database Provider (dummy implementation)

final databaseProvider = Provider<Database>((ref) => DummyDatabase());

abstract class Database {

Future<void> saveUser(User user);

}

class DummyDatabase implements Database {

@override

Future<void> saveUser(User user) {

// Dummy implementation to simulate saving user to a database

return Future.delayed(Duration(seconds: 2));

}

}

// 4. Create a StateNotifier

class UserRegistrationState extends StateNotifier<bool> {

final Database \_database;

UserRegistrationState(this.\_database) : super(false);

Future<void> registerUser(User user) async {

try {

// Start loading state

state = true;

// Save user to the database

await \_database.saveUser(user);

// Registration successful

state = false;

// Add further logic here like showing a success message or navigation

} catch (e) {

// Handle errors

state = false;

// Add error handling logic like displaying error messages

print("Error registering user: $e");

}

}

}

// 5. Register User

final userRegistrationProvider =

StateNotifierProvider<UserRegistrationState, bool>((ref) {