Gamewise

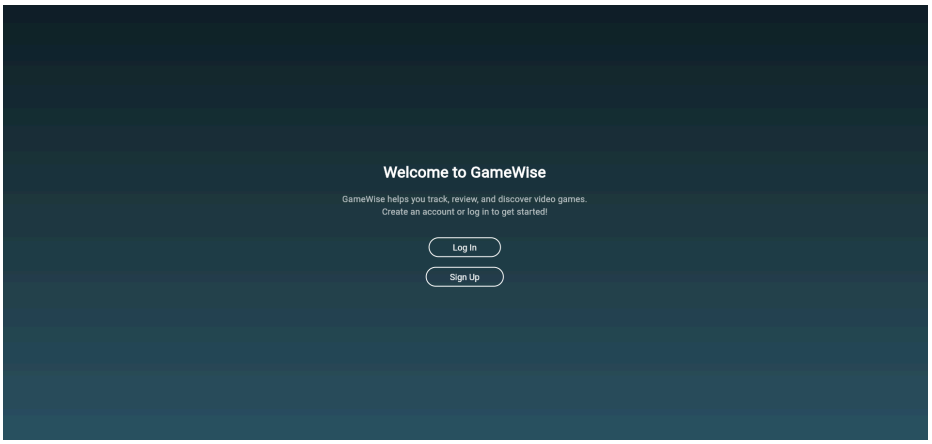
main.dart

```
/// Main entry point of the Gamewise Flutter app.
///
/// This file initializes Firebase and runs the app. Firebase is used for authentication
/// and possibly other database operations. We first ensure that Flutter bindings are
/// initialized before any asynchronous code executes.

import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'firebase_options.dart';

// Import navigation targets
import 'login_page.dart';
import 'signup_page.dart';
import 'dashboard_page.dart';

/// The `main` function is the starting point for any Flutter app.
/// Here, we make sure everything related to Flutter is initialized,
/// then initialize Firebase, and finally launch the app.
void main() async {
  WidgetsFlutterBinding.ensureInitialized(); // Required for async setup in main()
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform, // Platform-specific Firebase config
  );
  runApp(const GamewiseApp()); // Launch the root widget
}

/// Root widget of the Gamewise app.
/// This is a stateless widget because the root itself doesn't need to manage state.
class GamewiseApp extends StatelessWidget {
  const GamewiseApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Gamewise', // Title shown in app switcher
      debugShowCheckedModeBanner: false, // Hides the red debug banner
      home: const HomePage(), // Initial route of the app
    );
  }
}

/// Home screen shown on app launch.
/// Provides two main navigation options: Log In or Sign Up.
class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          // Background gradient styling
```

```
          gradient: LinearGradient(
            colors: [
              Color(0xFF0F2027),
              Color(0xFF203A43),
              Color(0xFF2C5364),
            ],
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
          ),
        ),
        child: Center(
          child: Padding(
            padding: const EdgeInsets.all(32.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                // Welcome title
                const Text(
                  'Welcome to GameWise',
                  style: TextStyle(
                    fontSize: 32,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ),
                  textAlign: TextAlign.center,
                ),
                const SizedBox(height: 20),

                // Introductory description
                const Text(
                  'GameWise helps you track, review, and discover video games.\n'
                  'Create an account or log in to get started!',
                  style: TextStyle(
                    fontSize: 18,
                    color: Colors.white70,
                  ),
                  textAlign: TextAlign.center,
                ),
                const SizedBox(height: 40),

                // Log In button
                OutlinedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => const LoginPage()),
                    );
                  },
                  style: OutlinedButton.styleFrom(
                    side: const BorderSide(color: Colors.white, width: 2),
                    foregroundColor: Colors.white,
                    padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 18),
                    textStyle: const TextStyle(fontSize: 18),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(30),
                    ),
                  ),
                  child: const Text('Log In'),
                ),
```

```
            const SizedBox(height: 20),

            // Sign Up button
            OutlinedButton(
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const SignUpPage()),
                );
              },
              style: OutlinedButton.styleFrom(
                side: const BorderSide(color: Colors.white, width: 2),
                foregroundColor: Colors.white,
                padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 18),
                textStyle: const TextStyle(fontSize: 18),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(30),
                ),
              ),
              child: const Text('Sign Up'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}

/*
Key Learning Points:
- Use WidgetsFlutterBinding.ensureInitialized() before async code in main().
- Use StatelessWidget for screens without internal state.
- Use Navigator.push() for navigation between pages.
- Prefer SizedBox for spacing instead of manual padding.
- Organize major pages in separate files for clarity and modularity.
- Apply early styling (e.g., background gradients) to improve UX.
*/
```

This is what we see when entering the website. A login button and a signup button.



login_page.dart

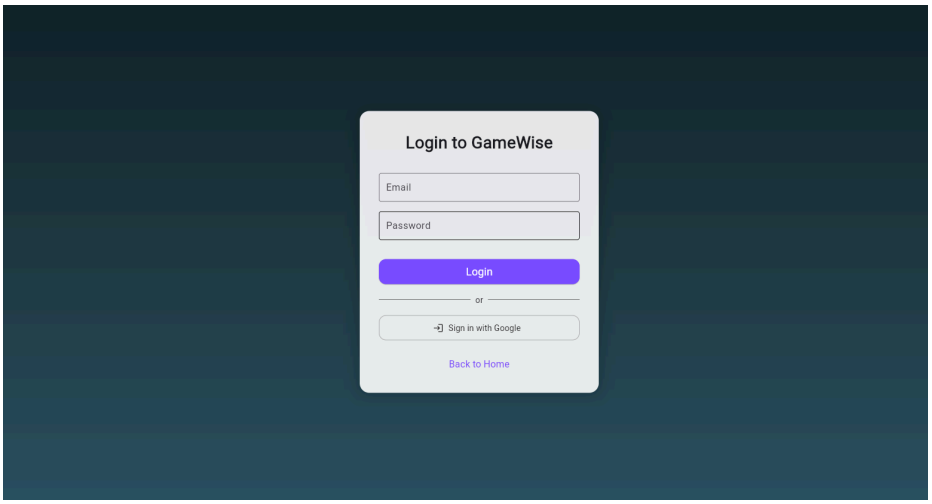
```
// Import core Flutter material UI library
import 'package:flutter/material.dart';
// Firebase Auth library for email/password and credential-based login
import 'package:firebase_auth/firebase_auth.dart';
// Google Sign-In library to allow Google-based authentication
import 'package:google_sign_in/google_sign_in.dart';
// Dashboard page shown after successful login
import 'dashboard_page.dart';
// HomePage allows users to return from login
import 'main.dart';

/// LoginPage is a stateful widget because it needs to track the values
/// of user inputs and handle login interactions.
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  // Controllers to capture user input for email and password
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  /// Handles login using email and password via Firebase Authentication.
  void _login() async {
    String email = _emailController.text.trim();
    String password = _passwordController.text.trim();

    // Simple input validation
    if (email.isEmpty || password.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Please enter email and password')),
      );
      return;
    }

    try {
      // Attempt to sign in with Firebase using email/password
      await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email,
        password: password,
      );

      // If successful, navigate to the dashboard and replace the current screen
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const DashboardPage()),
      );
    } on FirebaseAuthException catch (e) {
      // Handle Firebase-specific errors and display user-friendly messages
      String message = '';
      if (e.code == 'user-not-found') {
        message = 'No user found for that email.';
      } else if (e.code == 'wrong-password') {
        message = 'Wrong password provided.';
      } else {
        message = 'Login failed: ${e.message}';
```

```
    }
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(message)),
    );
  }
}

/// Handles Google Sign-In and Firebase Authentication with Google credentials.
Future<void> _signInWithGoogle() async {
  try {
    // Launch Google sign-in prompt
    final GoogleSignInAccount? googleUser = await GoogleSignIn(
      clientId: '602783046019-5333cekolocth3ntgnbcgdummml1kg4l.apps.googleusercontent.com',
    ).signIn();

    // If user cancels login
    if (googleUser == null) {
      return;
    }

    // Get auth credentials from Google account
    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    // Use credentials to sign in with Firebase
    await FirebaseAuth.instance.signInWithCredential(credential);

    // Navigate to the dashboard
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const DashboardPage()),
    );
  } catch (e) {
    // Catch any unexpected errors
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Google sign-in failed: ${e.toString()}')),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      // Gradient background
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          colors: [
            Color(0xFF0F2027),
            Color(0xFF203A43),
            Color(0xFF2C5364),
          ],
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
        ),
      ),
```

```
      child: Center(
        child: SingleChildScrollView(
          child: Container(
            constraints: const BoxConstraints(maxWidth: 400), // max width for better desktop/tablet layout
            padding: const EdgeInsets.all(32),
            decoration: BoxDecoration(
              color: Colors.white.withOpacity(0.9),
              borderRadius: BorderRadius.circular(16),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.2),
                  blurRadius: 20,
                  spreadRadius: 5,
                ),
              ],
            ),
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                const Text(
                  'Login to GameWise',
                  style: TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 32),

                // Email input field
                TextField(
                  controller: _emailController,
                  decoration: const InputDecoration(
                    labelText: 'Email',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 16),

                // Password input field
                TextField(
                  controller: _passwordController,
                  obscureText: true, // hides password input
                  decoration: const InputDecoration(
                    labelText: 'Password',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 32),

                // Login button (email + password)
                ElevatedButton(
                  onPressed: _login,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.deepPurpleAccent,
                    foregroundColor: Colors.white,
                    minimumSize: const Size.fromHeight(50),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(12),
                    ),
```

```
                ),
                child: const Text('Login', style: TextStyle(fontSize: 18)),
              ),
              const SizedBox(height: 16),

              // Divider with "or"
              Row(
                children: const [
                  Expanded(child: Divider(color: Colors.black54)),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: 8.0),
                    child: Text("or"),
                  ),
                  Expanded(child: Divider(color: Colors.black54)),
                ],
              ),
              const SizedBox(height: 16),

              // Google sign-in button
              OutlinedButton.icon(
                onPressed: _signInWithGoogle,
                icon: const Icon(Icons.login),
                label: const Text('Sign in with Google'),
                style: OutlinedButton.styleFrom(
                  foregroundColor: Colors.black87,
                  minimumSize: const Size.fromHeight(50),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                  side: const BorderSide(color: Colors.black26),
                ),
              ),
              const SizedBox(height: 24),

              // Navigation: Back to home
              TextButton(
                onPressed: () {
                  Navigator.pushAndRemoveUntil(
                    context,
                    MaterialPageRoute(builder: (context) => const HomePage()),
                    (route) => false, // Clear the entire navigation stack
                  );
                },
                child: const Text(
                  'Back to Home',
                  style: TextStyle(color: Colors.deepPurpleAccent, fontSize: 16),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

When we click on the log in button we get into here:



signup_page.dart

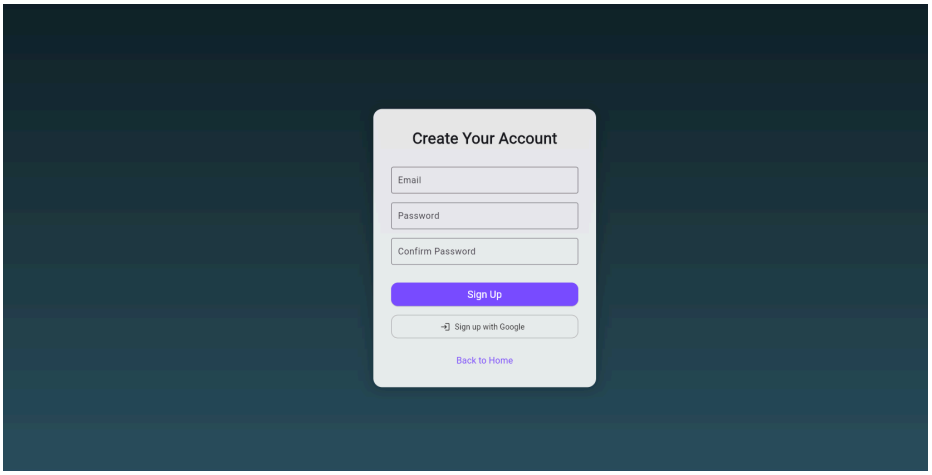
```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'dashboard_page.dart';
import 'main.dart';

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

  @override
  State<SignUpPage> createState() => _SignUpPageState();
}

class _SignUpPageState extends State<SignUpPage> {
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final _confirmPasswordController = TextEditingController();

  void _signUp() async {
    String email = _emailController.text.trim();
    String password = _passwordController.text.trim();
    String confirmPassword = _confirmPasswordController.text.trim();

    if (email.isEmpty || password.isEmpty || confirmPassword.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Please fill all fields')),
      );
      return;
    }

    if (password != confirmPassword) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Passwords do not match')),
      );
      return;
    }
```

```
  try {
    final userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Account created for ${userCredential.user?.email ?? 'unknown'}')),
    );

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const DashboardPage()),
    );
  } on FirebaseAuthException catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Error: ${e.message}')),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Unexpected error: ${e.toString()}')),
    );
  }
}

Future<void> _signInWithGoogle() async {
  try {
    final GoogleSignInAccount? googleUser = await GoogleSignIn(
      clientId: '602783046019-5333cekolocth3ntgnbcgdummml1kg4l.apps.googleusercontent.com',
    ).signIn();

    if (googleUser == null) return;

    final googleAuth = await googleUser.authentication;
    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    await FirebaseAuth.instance.signInWithCredential(credential);

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const DashboardPage()),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Google sign-up failed: ${e.toString()}')),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: const BoxDecoration(
        gradient: LinearGradient(
          colors: [
```

```
          Color(0xFF0F2027),
          Color(0xFF203A43),
          Color(0xFF2C5364),
        ],
        begin: Alignment.topCenter,
        end: Alignment.bottomCenter,
      ),
    ),
    child: Center(
      child: SingleChildScrollView(
        child: Container(
          constraints: const BoxConstraints(maxWidth: 400),
          padding: const EdgeInsets.all(32),
          decoration: BoxDecoration(
            color: Colors.white.withOpacity(0.9),
            borderRadius: BorderRadius.circular(16),
            boxShadow: [
              BoxShadow(
                color: Colors.black.withOpacity(0.2),
                blurRadius: 20,
                spreadRadius: 5,
              ),
            ],
          ),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              const Text(
                'Create Your Account',
                style: TextStyle(fontSize: 28, fontWeight: FontWeight.bold),
              ),
              const SizedBox(height: 32),
              TextField(
                controller: _emailController,
                decoration: const InputDecoration(
                  labelText: 'Email',
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 16),
              TextField(
                controller: _passwordController,
                obscureText: true,
                decoration: const InputDecoration(
                  labelText: 'Password',
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 16),
              TextField(
                controller: _confirmPasswordController,
                obscureText: true,
                decoration: const InputDecoration(
                  labelText: 'Confirm Password',
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 32),
              ElevatedButton(
```

```
                onPressed: _signUp,
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.deepPurpleAccent,
                  foregroundColor: Colors.white,
                  minimumSize: const Size.fromHeight(50),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                ),
                child: const Text('Sign Up', style: TextStyle(fontSize: 18)),
              ),
              const SizedBox(height: 16),
              OutlinedButton.icon(
                onPressed: _signInWithGoogle,
                icon: const Icon(Icons.login),
                label: const Text('Sign up with Google'),
                style: OutlinedButton.styleFrom(
                  foregroundColor: Colors.black87,
                  minimumSize: const Size.fromHeight(50),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12),
                  ),
                  side: const BorderSide(color: Colors.black26),
                ),
              ),
              const SizedBox(height: 24),
              TextButton(
                onPressed: () {
                  Navigator.pushAndRemoveUntil(
                    context,
                    MaterialPageRoute(builder: (context) => const HomePage()),
                    (route) => false,
                  );
                },
                child: const Text(
                  'Back to Home',
                  style: TextStyle(color: Colors.deepPurpleAccent, fontSize: 16),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

When click on the sign up button:



dashboard_page.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'login_page.dart';
import 'add_game_page.dart';
import 'create_list_page.dart';
import 'edit_profile_page.dart';
import 'edit_game_page.dart';
import 'view_list_page.dart';

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

  @override
  State<DashboardPage> createState() => _DashboardPageState();
}

class _DashboardPageState extends State<DashboardPage> {
  int _selectedIndex = 0;
  final _searchController = TextEditingController();
  String _searchQuery = '';

  void _logout() async {
    await FirebaseAuth.instance.signOut();
    Navigator.pushReplacement(context, MaterialPageRoute(builder: (_) => const LoginPage()));
  }

  void _navigateTo(int index) {
    setState(() {
      _selectedIndex = index;
    });
    Navigator.pop(context); // Close the drawer
  }

  Future<void> deleteGameEverywhere(String gameId) async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null) return;

  final userRef = FirebaseFirestore.instance.collection('users').doc(user.uid);
```

```
  // 1. Delete from "My Games"
  await userRef.collection('games').doc(gameId).delete();

  // 2. Fetch all user lists
  final listsSnapshot = await userRef.collection('lists').get();

  for (final listDoc in listsSnapshot.docs) {
    final listId = listDoc.id;
    final gameDocRef = userRef
        .collection('lists')
        .doc(listId)
        .collection('games')
        .doc(gameId);

    final gameDoc = await gameDocRef.get();
    if (gameDoc.exists) {
      await gameDocRef.delete();
    }
  }
}


  @override
  Widget build(BuildContext context) {
    final user = FirebaseAuth.instance.currentUser;

    return Scaffold(
      appBar: AppBar(
        title: const Text("GameWise Dashboard"),
      ),
      drawer: Drawer(
        child: ListView(
          padding: EdgeInsets.zero,
          children: [
            DrawerHeader(
              decoration: const BoxDecoration(color: Colors.deepPurpleAccent),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Icon(Icons.videogame_asset, size: 48, color: Colors.white),
                  const SizedBox(height: 8),
                  Text(user?.email ?? '', style: const TextStyle(color: Colors.white)),
                ],
              ),
            ),
            ListTile(
              leading: const Icon(Icons.gamepad),
              title: const Text("My Games"),
              selected: _selectedIndex == 0,
              onTap: () => _navigateTo(0),
            ),
            ListTile(
              leading: const Icon(Icons.list),
              title: const Text("My Lists"),
              selected: _selectedIndex == 1,
              onTap: () => _navigateTo(1),
            ),
            ListTile(
              leading: const Icon(Icons.person),
```

```
            title: const Text("My Profile"),
            selected: _selectedIndex == 2,
            onTap: () => _navigateTo(2),
          ),
          const Divider(),
          ListTile(
            leading: const Icon(Icons.logout),
            title: const Text("Logout"),
            onTap: _logout,
          ),
        ],
      ),
    ),
    body: IndexedStack(
      index: _selectedIndex,
      children: [
        _buildGamesView(context, user),
        _buildListsView(context, user),
        _buildProfileView(context, user),
      ],
    ),
    floatingActionButton: _selectedIndex == 0
        ? FloatingActionButton(
            onPressed: () {
              Navigator.push(context, MaterialPageRoute(builder: (_) => const AddGamePage()));
            },
            child: const Icon(Icons.add),
          )
        : _selectedIndex == 1
            ? FloatingActionButton(
                onPressed: () {
                  Navigator.push(context, MaterialPageRoute(builder: (_) => const CreateListPage()));
                },
                child: const Icon(Icons.add),
              )
            : null,
  );
}

Widget _buildGamesView(BuildContext context, User? user) {
final gamesRef = FirebaseFirestore.instance
    .collection('users')
    .doc(user?.uid)
    .collection('games')
    .orderBy('createdAt', descending: true);

return Container(
color: const Color(0xFF1E1E2C),
padding: const EdgeInsets.all(16),
child: Container(
  padding: const EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.white.withOpacity(0.08),
    borderRadius: BorderRadius.circular(12),
    boxShadow: [BoxShadow(color: Colors.black26, blurRadius: 6)],
  ),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
```

```
const Text(
  'My Games',
  style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, color: Colors.white),
),
const SizedBox(height: 12),
TextField(
  controller: _searchController,
  decoration: InputDecoration(
    hintText: 'Search games...',
    fillColor: Colors.white,
    filled: true,
    prefixIcon: const Icon(Icons.search),
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(12)),
  ),
  onChanged: (value) {
    setState(() => _searchQuery = value.toLowerCase());
  },
),
const SizedBox(height: 16),
Expanded(
  child: StreamBuilder<QuerySnapshot>(
    stream: gamesRef.snapshots(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Center(child: CircularProgressIndicator());
      }
      if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
        return const Center(
          child: Text('You haven\'t added any games yet.',
              style: TextStyle(color: Colors.white70)),
        );
      }

      final games = snapshot.data!.docs.where((doc) {
        final title = (doc['title'] ?? '').toString().toLowerCase();
        return title.contains(_searchQuery);
      }).toList();

      if (games.isEmpty) {
        return const Center(
          child: Text('No matching games found.',
              style: TextStyle(color: Colors.white70)),
        );
      }

      return ListView.separated(
        itemCount: games.length,
        separatorBuilder: (_, __) => const Divider(indent: 72, endIndent: 16, thickness: 0.4, color: Colors.white24),
        itemBuilder: (context, index) {
          final game = games[index];
          final data = game.data() as Map<String, dynamic>;

          return Dismissible(
            key: Key(game.id),
            direction: DismissDirection.endToStart,
            background: Container(
              color: Colors.red,
              alignment: Alignment.centerRight,
              padding: const EdgeInsets.only(right: 20),
```

```
                child: const Icon(Icons.delete_forever, color: Colors.white, size: 32),
              ),
              confirmDismiss: (_) async {
                return await showDialog(
                  context: context,
                  builder: (context) => AlertDialog(
                    title: const Text('Delete Game?'),
                    content: const Text('Are you sure you want to delete this game?'),
                    actions: [
                      TextButton(onPressed: () => Navigator.of(context).pop(false), child: const Text('Cancel')),
                      TextButton(onPressed: () => Navigator.of(context).pop(true), child: const Text('Delete')),
                    ],
                  ),
                );
              },
              onDismissed: (_) async {
                await deleteGameEverywhere(game.id);
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text('Game deleted from My Games and all lists')),
                );
              },
              child: ListTile(
                leading: data['coverUrl'] != null && data['coverUrl'] != ''
                    ? ClipRRect(
                        borderRadius: BorderRadius.circular(6),
                        child: Image.network(data['coverUrl'], width: 50, height: 50, fit: BoxFit.cover),
                      )
                    : const Icon(Icons.videogame_asset, color: Colors.deepPurpleAccent),
                title: Text(data['title'] ?? 'No Title',
                    style: const TextStyle(color: Colors.white, fontWeight: FontWeight.w600)),
                subtitle: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    if (data['platform'] != null) Text('Platform: ${data['platform']}', style: const TextStyle(color: Colors.white70
                    if (data['releaseDate'] != null) Text('Release: ${data['releaseDate']}', style: const TextStyle(color: Colors.w
                    if (data['rating'] != null) Text('Rating: ${data['rating']}/5', style: const TextStyle(color: Colors.amber)),
                    if (data['review'] != null && data['review'].toString().trim().isNotEmpty)
                      Padding(
                        padding: const EdgeInsets.only(top: 4),
                        child: Text('"${data['review']}"', style: const TextStyle(color: Colors.white60, fontStyle: FontStyle.italic)
                      ),
                  ],
                ),
                trailing: const Icon(Icons.chevron_right, color: Colors.white54),
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder: (_) => EditGamePage(gameId: game.id, gameData: data)),
                  );
                },
                contentPadding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
              ),
            );
          },
        );
      },
    ),
  ),
],
```

```
    ),
  ),
);
}
  Widget _buildListsView(BuildContext context, User? user) {
  final listsRef = FirebaseFirestore.instance
      .collection('users')
      .doc(user?.uid)
      .collection('lists')
      .orderBy('createdAt', descending: true);

  return Container(
    color: const Color(0xFF1E1E2C),
    padding: const EdgeInsets.all(16),
    child: Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: Colors.white.withOpacity(0.08),
        borderRadius: BorderRadius.circular(12),
        boxShadow: [BoxShadow(color: Colors.black26, blurRadius: 6)],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Text(
            'My Lists',
            style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, color: Colors.white),
          ),
          const SizedBox(height: 12),
          Expanded(
            child: StreamBuilder<QuerySnapshot>(
              stream: listsRef.snapshots(),
              builder: (context, snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                }

                if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
                  return const Center(
                    child: Text('You haven\'t created any lists yet.',
                        style: TextStyle(color: Colors.white70)),
                  );
                }

                final lists = snapshot.data!.docs;

                return ListView.separated(
                  itemCount: lists.length,
                  separatorBuilder: (_, __) => const Divider(indent: 72, endIndent: 16, thickness: 0.4, color: Colors.white24),
                  itemBuilder: (context, index) {
                    final list = lists[index];
                    final data = list.data() as Map<String, dynamic>;

                    return Dismissible(
                      key: Key(list.id),
                      direction: DismissDirection.endToStart,
                      background: Container(
                        color: Colors.red,
                        alignment: Alignment.centerRight,
```

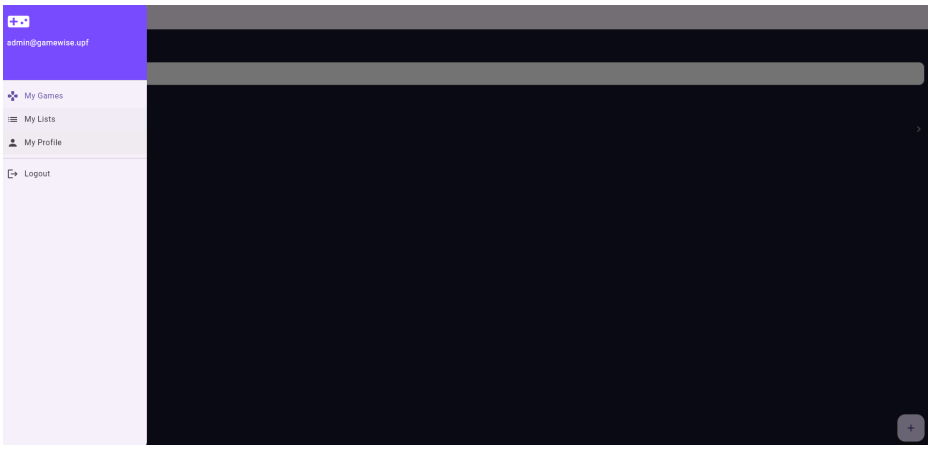
```
                padding: const EdgeInsets.only(right: 20),
                child: const Icon(Icons.delete_forever, color: Colors.white, size: 32),
              ),
              confirmDismiss: (_) async {
                return await showDialog(
                  context: context,
                  builder: (context) => AlertDialog(
                    title: const Text('Delete List?'),
                    content: const Text('Are you sure you want to delete this list?'),
                    actions: [
                      TextButton(onPressed: () => Navigator.of(context).pop(false), child: const Text('Cancel')),
                      TextButton(onPressed: () => Navigator.of(context).pop(true), child: const Text('Delete')),
                    ],
                  ),
                );
              },
              onDismissed: (_) async {
                await FirebaseFirestore.instance
                    .collection('users')
                    .doc(user!.uid)
                    .collection('lists')
                    .doc(list.id)
                    .delete();

                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(content: Text('List deleted')),
                );
              },
              child: ListTile(
                leading: const Icon(Icons.list_alt, color: Colors.deepPurpleAccent),
                title: Text(
                  data['title'] ?? 'No Title',
                  style: const TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                subtitle: data['description'] != null && data['description'].toString().trim().isNotEmpty
                    ? Text(data['description'], style: const TextStyle(color: Colors.white70))
                    : null,
                trailing: const Icon(Icons.chevron_right, color: Colors.white54),
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (_) => ViewListPage(listId: list.id, listData: data),
                    ),
                  );
                },
                contentPadding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
              ),
            );
          },
        );
      },
    ),
  ),
 ],
),
```

```
    ),
  );
}

Widget _buildProfileView(BuildContext context, User? user) {
  return FutureBuilder<DocumentSnapshot>(
    future: FirebaseFirestore.instance.collection('users').doc(user?.uid).get(),
    builder: (context, snapshot) {
      if (!snapshot.hasData) {
        return const Center(child: CircularProgressIndicator());
      }

      final data = snapshot.data!.data() as Map<String, dynamic>? ?? {};

      final gamesRef = FirebaseFirestore.instance
          .collection('users')
          .doc(user?.uid)
          .collection('games')
          .orderBy('createdAt', descending: true);

      final listsRef = FirebaseFirestore.instance
          .collection('users')
          .doc(user?.uid)
          .collection('lists')
          .orderBy('createdAt', descending: true);

      return Container(
        color: const Color(0xFF1E1E2C),
        child: Row(
          children: [
            // LEFT PANEL: Profile Info
            Container(
              width: 300,
              margin: const EdgeInsets.all(16),
              padding: const EdgeInsets.all(16),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.9),
                borderRadius: BorderRadius.circular(12),
              ),
              child: Column(
                children: [
                  const CircleAvatar(radius: 50, backgroundColor: Colors.deepPurpleAccent),
                  const SizedBox(height: 16),
                  Text(data['name'] ?? 'No Name', style: const TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
                  const SizedBox(height: 8),
                  Text(user?.email ?? 'No email', style: const TextStyle(color: Colors.black54)),
                  if ((data['bio'] ?? '').toString().isNotEmpty) ...[
                    const SizedBox(height: 12),
                    Text(data['bio'], textAlign: TextAlign.center),
                  ],
                  if ((data['location'] ?? '').toString().isNotEmpty) ...[
                    const SizedBox(height: 12),
                    Row(mainAxisAlignment: MainAxisAlignment.center, children: [
                      const Icon(Icons.location_on, size: 18),
                      const SizedBox(width: 4),
                      Text(data['location']),
                    ]),
                  ],
                  if ((data['age'] ?? 0) != 0) ...[
```

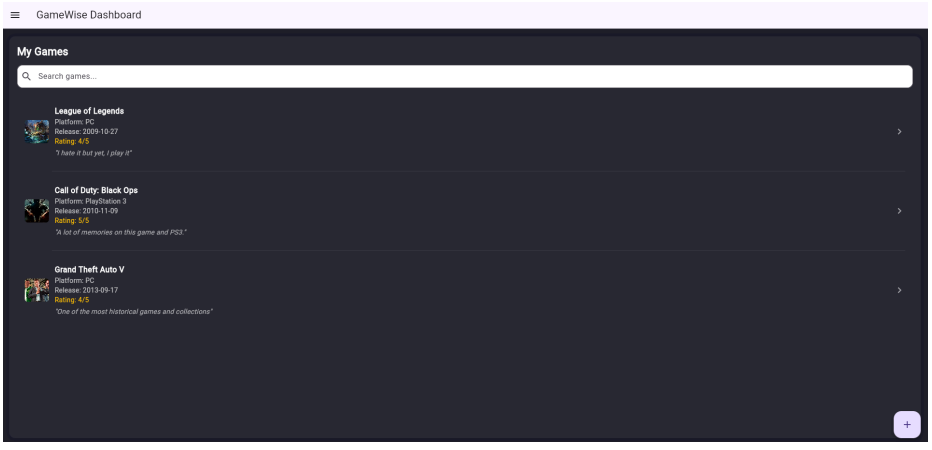
```
          const SizedBox(height: 8),
          Row(mainAxisAlignment: MainAxisAlignment.center, children: [
            const Icon(Icons.cake, size: 18),
            const SizedBox(width: 4),
            Text('Age: ${data['age']}'),
          ]),
        ],
        const SizedBox(height: 16),
        ElevatedButton.icon(
          onPressed: () {
            Navigator.push(context, MaterialPageRoute(builder: (_) => const EditProfilePage()));
          },
          icon: const Icon(Icons.edit, color: Colors.white),
          label: const Text('Edit Profile', style: TextStyle(color: Colors.white)),
          style: ElevatedButton.styleFrom(
            backgroundColor: Color(0xFF2C5364), // same as the dark gradient tone
            minimumSize: const Size.fromHeight(48),
            shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8)),
            elevation: 4,
          ),
        ),
      ],
    ),
  ),
),
// RIGHT PANEL: Recent games and lists with clearer sections
Expanded(
  child: Padding(
    padding: const EdgeInsets.symmetric(vertical: 16, horizontal: 8),
    child: Column(
      children: [
        // Recently Added Games Section
        Container(
          padding: const EdgeInsets.all(16),
          margin: const EdgeInsets.only(bottom: 16),
          decoration: BoxDecoration(
            color: Colors.white.withOpacity(0.08),
            borderRadius: BorderRadius.circular(12),
            boxShadow: [
              BoxShadow(color: Colors.black26, blurRadius: 6, offset: Offset(0, 2)),
            ],
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const Text('Recently Added Games',
                  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color: Colors.white)),
              const SizedBox(height: 12),
              SizedBox(
                height: 180,
                child: StreamBuilder<QuerySnapshot>(
                  stream: gamesRef.limit(5).snapshots(),
                  builder: (context, snapshot) {
                    if (!snapshot.hasData) return const Center(child: CircularProgressIndicator());
                    final docs = snapshot.data!.docs;
                    if (docs.isEmpty) {
                      return const Text('No recent games', style: TextStyle(color: Colors.white70));
                    }
                    return ListView.builder(
                      itemCount: docs.length,
```

```
                itemBuilder: (context, index) {
                  final game = docs[index].data() as Map<String, dynamic>;
                  return ListTile(
                    dense: true,
                    leading: game['coverUrl'] != null && game['coverUrl'] != ''
                        ? ClipRRect(
                            borderRadius: BorderRadius.circular(6),
                            child: Image.network(game['coverUrl'], width: 40, height: 40, fit: BoxFit.cover),
                          )
                        : const Icon(Icons.videogame_asset, color: Colors.deepPurpleAccent),
                    title: Text(game['title'] ?? 'No Title', style: const TextStyle(color: Colors.white)),
                  );
                },
              );
            },
          ),
        ),
      ],
    ),
  ),

  // Recently Created Lists Section
  Container(
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: Colors.white.withOpacity(0.08),
      borderRadius: BorderRadius.circular(12),
      boxShadow: [
        BoxShadow(color: Colors.black26, blurRadius: 6, offset: Offset(0, 2)),
      ],
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const Text('Recently Created Lists',
            style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color: Colors.white)),
        const SizedBox(height: 12),
        SizedBox(
          height: 180,
          child: StreamBuilder<QuerySnapshot>(
            stream: listsRef.limit(5).snapshots(),
            builder: (context, snapshot) {
              if (!snapshot.hasData) return const Center(child: CircularProgressIndicator());
              final docs = snapshot.data!.docs;
              if (docs.isEmpty) {
                return const Text('No recent lists', style: TextStyle(color: Colors.white70));
              }
              return ListView.builder(
                itemCount: docs.length,
                itemBuilder: (context, index) {
                  final list = docs[index].data() as Map<String, dynamic>;
                  return ListTile(
                    dense: true,
                    leading: const Icon(Icons.list_alt, color: Colors.deepPurpleAccent),
                    title: Text(list['title'] ?? 'No Title', style: const TextStyle(color: Colors.white)),
                    subtitle: list['description'] != null
                        ? Text(list['description'], style: const TextStyle(color: Colors.white70))
                        : null,
                  );
```

```
                        },
                      );
                    },
                  ),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
}


}
```

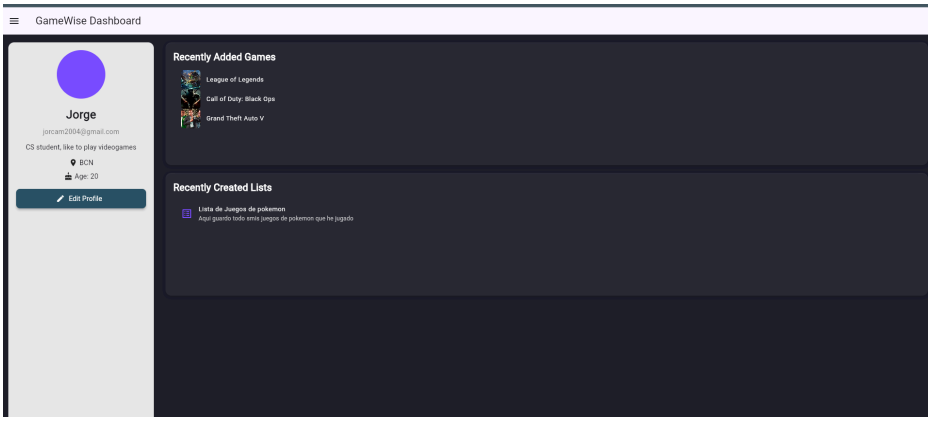


My Games:



My lists

My profile:



Now, the behaviour of the games. A game can be on my games but not on any list. When you delete a game from my games it is deleted from all lists, but when deleted from a list not from my games.

add_game_page.dart

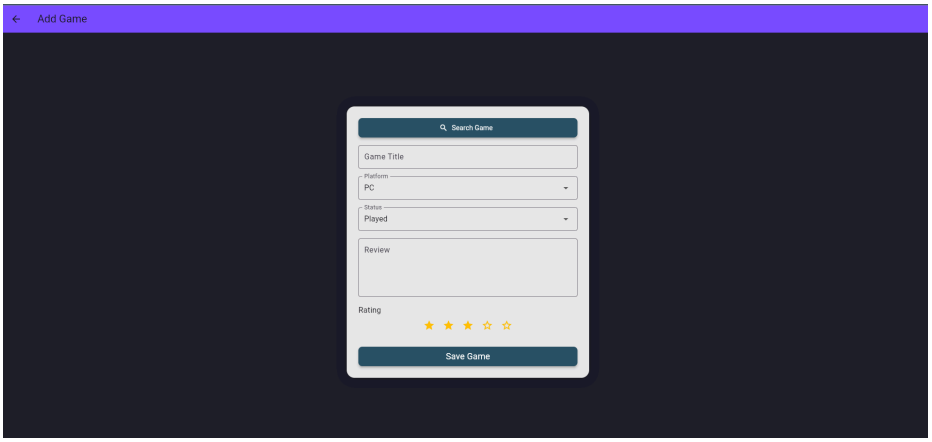
```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'search_game_page.dart';

Future<void> saveGame({
  required String title,
  required String platform,
  required String status,
  required String review,
  required int rating,
  String? coverUrl,
  String? releaseDate,
  List<String>? genres,
  String? listId,
}) async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null) return;

  final baseRef = FirebaseFirestore.instance.collection('users').doc(user.uid);
```

```
  final gameId = baseRef.collection('games').doc().id;

  final gameData = {
    'id': gameId,
    'title': title,
    'platform': platform,
    'status': status,
    'review': review,
    'rating': rating,
    'coverUrl': coverUrl,
    'releaseDate': releaseDate,
    'genres': genres,
    'createdAt': FieldValue.serverTimestamp(),
  };

  await baseRef.collection('games').doc(gameId).set(gameData);

  if (listId != null) {
    await baseRef
        .collection('lists')
        .doc(listId)
        .collection('games')
        .doc(gameId)
        .set(gameData);
  }
}

class AddGamePage extends StatefulWidget {
  final Map<String, dynamic>? initialData;
  final String? listId;

  const AddGamePage({super.key, this.initialData, this.listId});

  @override
  State<AddGamePage> createState() => _AddGamePageState();
}

class _AddGamePageState extends State<AddGamePage> {
  final _formKey = GlobalKey<FormState>();
  final _titleController = TextEditingController();
  final _reviewController = TextEditingController();

  String _selectedPlatform = 'PC';
  String _selectedStatus = 'Played';
  int _rating = 3;
  String? _coverUrl;
  String? _releaseDate;
  List<String> _genres = [];

  @override
  void initState() {
    super.initState();
    if (widget.initialData != null) {
      _titleController.text = widget.initialData!['title'] ?? '';
      final platformsList = widget.initialData!['platforms'] as List<dynamic>? ?? [];
      if (platformsList.isNotEmpty) {
        final platformName = platformsList[0]['platform']['name'] ?? 'PC';
        _selectedPlatform = platformName;
      }
```

```
    _coverUrl = widget.initialData!['background_image'];
    _releaseDate = widget.initialData!['released'];
    final genresList = widget.initialData!['genres'] as List<dynamic>? ?? [];
    _genres = genresList.map((g) => g['name'].toString()).toList();
  }
}

Future<void> _openSearchPage() async {
  final selectedGame = await Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const SearchGamePage()),
  );

  if (selectedGame != null) {
    setState(() {
      _titleController.text = selectedGame['title'] ?? '';
      final platformsList = selectedGame['platforms'] as List<dynamic>? ?? [];
      if (platformsList.isNotEmpty) {
        final platformName = platformsList[0]['platform']['name'] ?? 'PC';
        _selectedPlatform = platformName;
      }
      _coverUrl = selectedGame['background_image'] ?? '';
      _releaseDate = selectedGame['released'];
      final genresList = selectedGame['genres'] as List<dynamic>? ?? [];
      _genres = genresList.map((g) => g['name'].toString()).toList();
    });
  }
}

void _save() async {
  if (!_formKey.currentState!.validate()) return;

  await saveGame(
    title: _titleController.text.trim(),
    platform: _selectedPlatform,
    status: _selectedStatus,
    review: _reviewController.text.trim(),
    rating: _rating,
    coverUrl: _coverUrl ?? '',
    releaseDate: _releaseDate ?? '',
    genres: _genres,
    listId: widget.listId,
  );

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Game saved successfully!')),
  );

  Navigator.pop(context);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Add Game'),
      backgroundColor: Colors.deepPurpleAccent,
    ),
    body: Container(
```

```
      color: const Color(0xFF1E1E2C),
      child: Center(
        child: SingleChildScrollView(
          padding: const EdgeInsets.all(24.0),
          child: Container(
            constraints: const BoxConstraints(maxWidth: 500),
            padding: const EdgeInsets.all(24.0),
            decoration: BoxDecoration(
              color: Colors.white.withOpacity(0.9),
              borderRadius: BorderRadius.circular(16),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.2),
                  blurRadius: 20,
                  spreadRadius: 5,
                ),
              ],
            ),
            child: Form(
              key: _formKey,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  ElevatedButton.icon(
                    onPressed: _openSearchPage,
                    icon: const Icon(Icons.search, color: Colors.white),
                    label: const Text('Search Game', style: TextStyle(color: Colors.white)),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: const Color(0xFF2C5364),
                      minimumSize: const Size.fromHeight(48),
                      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8)),
                      elevation: 4,
                    ),
                  ),
                  const SizedBox(height: 16),
                  TextFormField(
                    controller: _titleController,
                    decoration: const InputDecoration(
                      labelText: 'Game Title',
                      border: OutlineInputBorder(),
                    ),
                    validator: (value) => value == null || value.isEmpty ? 'Please enter the game title' : null,
                  ),
                  const SizedBox(height: 16),
                  DropdownButtonFormField<String>(
                    value: _selectedPlatform,
                    items: [
                      'PC', 'PlayStation 5', 'PlayStation 4', 'PlayStation 3',
                      'Xbox Series S/X', 'Xbox One', 'Xbox 360',
                      'Nintendo Switch', 'Nintendo DS', 'Nintendo 3DS',
                      'Game Boy', 'Game Boy Advance', 'Game Boy Color', 'Mobile'
                    ].map((platform) => DropdownMenuItem(value: platform, child: Text(platform))).toList(),
                    onChanged: (value) => setState(() => _selectedPlatform = value!),
                    decoration: const InputDecoration(
                      labelText: 'Platform',
                      border: OutlineInputBorder(),
                    ),
                  ),
                  const SizedBox(height: 16),
```

```
                DropdownButtonFormField<String>(
                  value: _selectedStatus,
                  items: ['Played', 'Playing', 'Wishlist'].map((status) => DropdownMenuItem(value: status, child: Text(status))).
                  onChanged: (value) => setState(() => _selectedStatus = value!),
                  decoration: const InputDecoration(
                    labelText: 'Status',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 16),
                TextFormField(
                  controller: _reviewController,
                  maxLines: 4,
                  decoration: const InputDecoration(
                    labelText: 'Review',
                    border: OutlineInputBorder(),
                    alignLabelWithHint: true,
                  ),
                ),
                const SizedBox(height: 16),
                const Text('Rating', style: TextStyle(fontSize: 16)),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: List.generate(5, (index) {
                    return IconButton(
                      onPressed: () => setState(() => _rating = index + 1),
                      icon: Icon(
                        _rating > index ? Icons.star : Icons.star_border,
                        color: Colors.amber,
                      ),
                    );
                  }),
                ),
                const SizedBox(height: 24),
                ElevatedButton(
                  onPressed: _save,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color(0xFF2C5364),
                    foregroundColor: Colors.white,
                    minimumSize: const Size.fromHeight(48),
                    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8)),
                    elevation: 4,
                  ),
                  child: const Text('Save Game', style: TextStyle(fontSize: 18)),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  );
 }
}
```

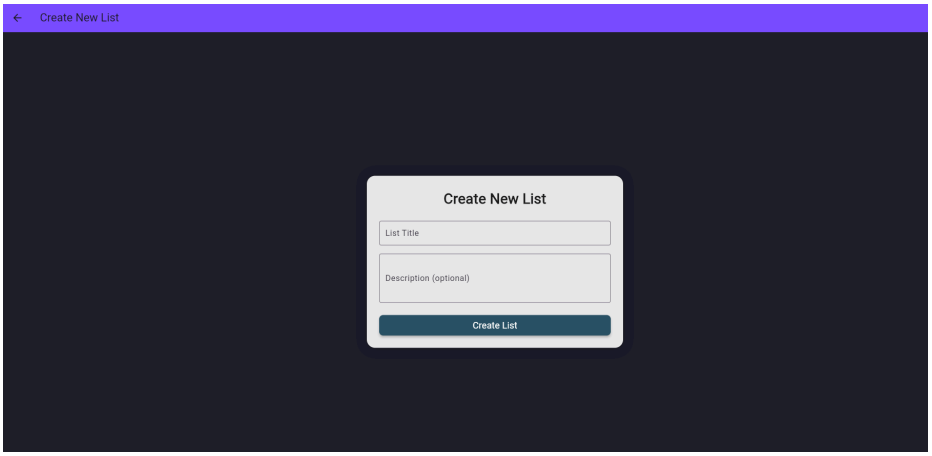


create_list_page.dart

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

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

  @override
  State<CreateListPage> createState() => _CreateListPageState();
}

class _CreateListPageState extends State<CreateListPage> {
  final _formKey = GlobalKey<FormState>();
  final _titleController = TextEditingController();
  final _descriptionController = TextEditingController();

  void _saveList() async {
    if (!_formKey.currentState!.validate()) return;

    final user = FirebaseAuth.instance.currentUser;
    if (user == null) return;

    final listsRef = FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .collection('lists');

    await listsRef.add({
      'title': _titleController.text.trim(),
      'description': _descriptionController.text.trim(),
      'createdAt': FieldValue.serverTimestamp(),
    });

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('List created successfully!')),
    );

    Navigator.pop(context);
  }

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        title: const Text('Create New List'),
        backgroundColor: Colors.deepPurpleAccent,
      ),
      body: Container(
        color: const Color(0xFF1E1E2C),
        child: Center(
          child: SingleChildScrollView(
            padding: const EdgeInsets.all(24.0),
            child: Container(
              constraints: const BoxConstraints(maxWidth: 500),
              padding: const EdgeInsets.all(24.0),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.9),
                borderRadius: BorderRadius.circular(16),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withOpacity(0.2),
                    blurRadius: 20,
                    spreadRadius: 5,
                  ),
                ],
              ),
              child: Form(
                key: _formKey,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    const Text(
                      'Create New List',
                      style: TextStyle(
                        fontSize: 28,
                        fontWeight: FontWeight.bold,
                        color: Colors.black87,
                      ),
                      textAlign: TextAlign.center,
                    ),
                    const SizedBox(height: 24),
                    TextFormField(
                      controller: _titleController,
                      decoration: const InputDecoration(
                        labelText: 'List Title',
                        border: OutlineInputBorder(),
                      ),
                      validator: (value) {
                        if (value == null || value.isEmpty) {
                          return 'Please enter a list title';
                        }
                        return null;
                      },
                    ),
                    const SizedBox(height: 16),
                    TextFormField(
                      controller: _descriptionController,
                      maxLines: 3,
                      decoration: const InputDecoration(
                        labelText: 'Description (optional)',
                        border: OutlineInputBorder(),
```

```
                ),
              ),
              const SizedBox(height: 24),
              ElevatedButton(
                onPressed: _saveList,
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFF2C5364),
                  foregroundColor: Colors.white,
                  minimumSize: const Size.fromHeight(48),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                  elevation: 4,
                ),
                child: const Text('Create List', style: TextStyle(fontSize: 18)),
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

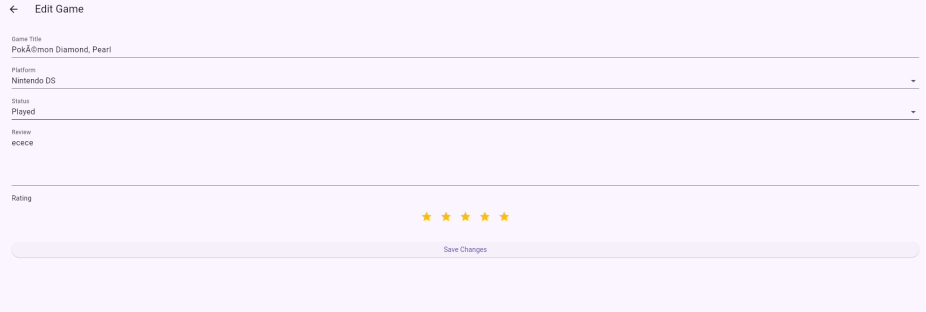


edit_game_page.dart

```
// Flutter UI library
import 'package:flutter/material.dart';
// Firebase Authentication to access the current user
import 'package:firebase_auth/firebase_auth.dart';
// Firebase Firestore to save lists
import 'package:cloud_firestore/cloud_firestore.dart';

/// Page that allows the user to create a new list (for organizing games).
class CreateListPage extends StatefulWidget {
 const CreateListPage({super.key});

 @override
 State<CreateListPage> createState() => _CreateListPageState();
```

```
}

class _CreateListPageState extends State<CreateListPage> {
  // Form key to validate the form fields
  final _formKey = GlobalKey<FormState>();

  // Controllers to capture user input
  final _titleController = TextEditingController();
  final _descriptionController = TextEditingController();

  /// Saves the new list to Firestore under the current user's document.
  void _saveList() async {
    if (!_formKey.currentState!.validate()) return; // Validate the form first

    final user = FirebaseAuth.instance.currentUser;
    if (user == null) return; // Ensure user is authenticated

    final listsRef = FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .collection('lists');

    await listsRef.add({
      'title': _titleController.text.trim(),
      'description': _descriptionController.text.trim(),
      'createdAt': FieldValue.serverTimestamp(), // Timestamp for sorting
    });

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('List created successfully!')), // Show success message
    );

    Navigator.pop(context); // Return to the previous page
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Create New List'), // Page title
      ),
      body: Padding(
        padding: const EdgeInsets.all(24.0),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              // Title input field
              TextFormField(
                controller: _titleController,
                decoration: const InputDecoration(labelText: 'List Title'),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter a list title'; // Validate title is not empty
                  }
                  return null;
                },
              ),
              const SizedBox(height: 16),
```

```
            // Description input field (optional)
            TextFormField(
              controller: _descriptionController,
              maxLines: 3,
              decoration: const InputDecoration(labelText: 'Description (optional)'),
            ),
            const SizedBox(height: 32),

            // Save button
            ElevatedButton(
              onPressed: _saveList,
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.deepPurpleAccent,
                minimumSize: const Size.fromHeight(50),
              ),
              child: const Text('Create List'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```



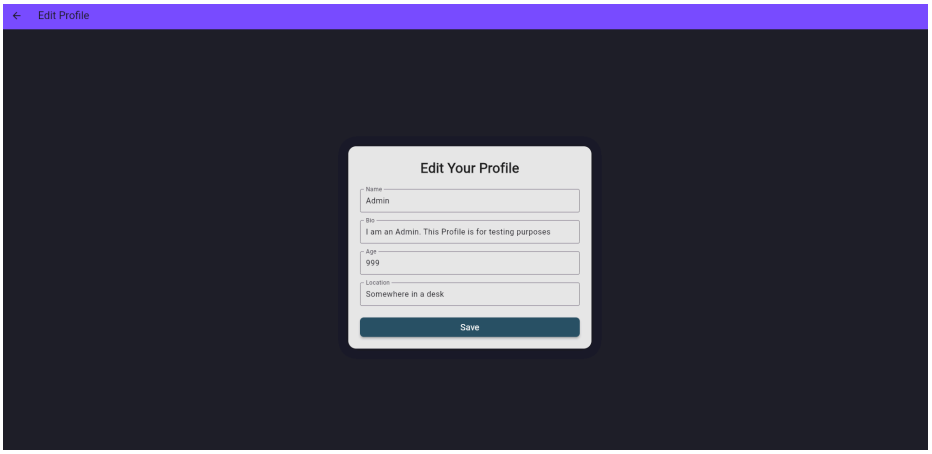
edit_profile_page.dart

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

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

 @override
 State<EditProfilePage> createState() => _EditProfilePageState();
}

class _EditProfilePageState extends State<EditProfilePage> {
 final _nameController = TextEditingController();
 final _bioController = TextEditingController();
 final _ageController = TextEditingController();
 final _locationController = TextEditingController();

 @override
 void initState() {
```

```
  super.initState();
  _loadProfile();
 }

 Future<void> _loadProfile() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null) return;

  final doc = await FirebaseFirestore.instance.collection('users').doc(user.uid).get();

  if (doc.exists) {
   final data = doc.data()!;
   _nameController.text = data['name'] ?? '';
   _bioController.text = data['bio'] ?? '';
   _ageController.text = (data['age'] ?? '').toString();
   _locationController.text = data['location'] ?? '';
  }
 }

 Future<void> _saveProfile() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user == null) return;

  final data = {
   'name': _nameController.text.trim(),
   'bio': _bioController.text.trim(),
   'age': int.tryParse(_ageController.text.trim()) ?? 0,
   'location': _locationController.text.trim(),
  };

  await FirebaseFirestore.instance
     .collection('users')
     .doc(user.uid)
     .set(data, SetOptions(merge: true));

  if (!mounted) return;

  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(content: Text('Profile updated successfully!')),
  );

  Navigator.pop(context);
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text('Edit Profile'),
    backgroundColor: Colors.deepPurpleAccent,
   ),
   body: Container(
    color: const Color(0xFF1E1E2C),
    child: Center(
     child: SingleChildScrollView(
      padding: const EdgeInsets.all(24.0),
      child: Container(
       constraints: const BoxConstraints(maxWidth: 500),
       padding: const EdgeInsets.all(24.0),
```

```
            decoration: BoxDecoration(
              color: Colors.white.withOpacity(0.9),
              borderRadius: BorderRadius.circular(16),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.2),
                  blurRadius: 20,
                  spreadRadius: 5,
                ),
              ],
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                const Text(
                  'Edit Your Profile',
                  style: TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                    color: Colors.black87,
                  ),
                  textAlign: TextAlign.center,
                ),
                const SizedBox(height: 24),
                TextField(
                  controller: _nameController,
                  decoration: const InputDecoration(
                    labelText: 'Name',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 16),
                TextField(
                  controller: _bioController,
                  decoration: const InputDecoration(
                    labelText: 'Bio',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 16),
                TextField(
                  controller: _ageController,
                  keyboardType: TextInputType.number,
                  decoration: const InputDecoration(
                    labelText: 'Age',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 16),
                TextField(
                  controller: _locationController,
                  decoration: const InputDecoration(
                    labelText: 'Location',
                    border: OutlineInputBorder(),
                  ),
                ),
                const SizedBox(height: 24),
                ElevatedButton(
                  onPressed: _saveProfile,
```

```
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFF2C5364),
                  foregroundColor: Colors.white,
                  minimumSize: const Size.fromHeight(48),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(8),
                  ),
                  elevation: 4,
                ),
                child: const Text('Save', style: TextStyle(fontSize: 18)),
              ),
            ],
          ),
        ),
      ),
    ),
  ),
 );
 }
}
```



search_game_page.dart

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

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

 @override
 State<SearchGamePage> createState() => _SearchGamePageState();
}

class _SearchGamePageState extends State<SearchGamePage> {
 final TextEditingController _controller = TextEditingController();
 Timer? _debounce;
 List<dynamic> _results = [];

 void _searchGames(String query) async {
  if (query.isEmpty) {
```

```
    setState(() => _results = []);
    return;
  }

  final url = Uri.parse('https://api.rawg.io/api/games?key=6469bcadbd654cd790bf47c07943e8ab&search=$query');
  final response = await http.get(url);

  if (response.statusCode == 200) {
    final data = json.decode(response.body);
    setState(() {
      _results = data['results'];
    });
  } else {
    setState(() => _results = []);
  }
}

void _onSearchChanged(String query) {
  if (_debounce?.isActive ?? false) _debounce!.cancel();
  _debounce = Timer(const Duration(milliseconds: 500), () {
    _searchGames(query.trim());
  });
}

@override
void dispose() {
  _debounce?.cancel();
  _controller.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFF1E1E2C),
    appBar: AppBar(
      title: const Text('Search Game'),
      backgroundColor: Colors.deepPurpleAccent,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          TextField(
            controller: _controller,
            onChanged: _onSearchChanged,
            onSubmitted: _searchGames,
            decoration: InputDecoration(
              hintText: 'Search for a game...',
              filled: true,
              fillColor: Colors.white,
              prefixIcon: const Icon(Icons.search),
              border: OutlineInputBorder(borderRadius: BorderRadius.circular(12)),
            ),
          ),
          const SizedBox(height: 16),
          Expanded(
            child: _results.isEmpty
                ? const Center(child: Text('No results', style: TextStyle(color: Colors.white70)))
```

```
            : ListView.builder(
                itemCount: _results.length,
                itemBuilder: (context, index) {
                 final game = _results[index];
                 return Card(
                  color: Colors.white.withOpacity(0.08),
                  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
                  margin: const EdgeInsets.symmetric(vertical: 8),
                  child: ListTile(
                   leading: game['background_image'] != null
                      ? ClipRRect(
                        borderRadius: BorderRadius.circular(6),
                        child: Image.network(
                         game['background_image'],
                         width: 50,
                         height: 50,
                         fit: BoxFit.cover,
                        ),
                       )
                      : const Icon(Icons.videogame_asset, color: Colors.deepPurpleAccent),
                   title: Text(game['name'], style: const TextStyle(color: Colors.white)),
                   subtitle: game['released'] != null
                      ? Text('Released: ${game['released']}', style: const TextStyle(color: Colors.white70))
                      : null,
                   onTap: () => Navigator.pop(context, {
                    'title': game['name'],
                    'platforms': game['platforms'],
                    'background_image': game['background_image'],
                    'released': game['released'],
                    'genres': game['genres'],
                   }),
                  ),
                 );
                },
               ),
          ),
         ],
        ),
       ),
      );
     }
    }
```

firebase_options.dart

```
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint

// Import necessary classes
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart' show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// This class provides Firebase configuration options for different platforms (Web, Android, iOS, macOS, Windows).
/// It's automatically generated by FlutterFire CLI, and is used during Firebase.initializeApp().
class DefaultFirebaseOptions {

  /// Returns the correct FirebaseOptions based on the current platform.
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      // If the app is running on the web
      return web;
    }
    // Determine platform based on defaultTargetPlatform
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        return windows;
      case TargetPlatform.linux:
        // Linux is not configured; throw an error
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        // If the platform is unknown, throw an error
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  /// Configuration for Web
  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyB8CJtbmr7Ezt1l13PtPbj5S_pl3tjNDIY',
    appId: '1:602783046019:web:752cdc2b993126c9ae3eb6',
    messagingSenderId: '602783046019',
    projectId: 'gamewise-6d228',
    authDomain: 'gamewise-6d228.firebaseapp.com',
    storageBucket: 'gamewise-6d228.firebasestorage.app',
  );

  /// Configuration for Android
  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyCEjcrF1ZoMYgsGw2ObJA84Orht5ETGpe0',
    appId: '1:602783046019:android:a643747cf18db284ae3eb6',
    messagingSenderId: '602783046019',
    projectId: 'gamewise-6d228',
```

```
    storageBucket: 'gamewise-6d228.firebasestorage.app',
  );

  /// Configuration for iOS
  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyC73TOxhRwHjlwZEL-SL9H79DaEUpbwh4A',
    appId: '1:602783046019:ios:416b50979dad9fa5ae3eb6',
    messagingSenderId: '602783046019',
    projectId: 'gamewise-6d228',
    storageBucket: 'gamewise-6d228.firebasestorage.app',
    iosClientId: '602783046019-vkdgoem30bgnqvgkc5c3pjtvm53kadru.apps.googleusercontent.com',
    iosBundleId: 'com.example.gamewiseWeb',
  );

  /// Configuration for macOS
  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyC73TOxhRwHjlwZEL-SL9H79DaEUpbwh4A',
    appId: '1:602783046019:ios:416b50979dad9fa5ae3eb6',
    messagingSenderId: '602783046019',
    projectId: 'gamewise-6d228',
    storageBucket: 'gamewise-6d228.firebasestorage.app',
    iosClientId: '602783046019-vkdgoem30bgnqvgkc5c3pjtvm53kadru.apps.googleusercontent.com',
    iosBundleId: 'com.example.gamewiseWeb',
  );

  /// Configuration for Windows
  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyB8CJtbmr7Ezt1l13PtPbj5S_pl3tjNDIY',
    appId: '1:602783046019:web:bc1dd29aae974abdae3eb6',
    messagingSenderId: '602783046019',
    projectId: 'gamewise-6d228',
    authDomain: 'gamewise-6d228.firebaseapp.com',
    storageBucket: 'gamewise-6d228.firebasestorage.app',
  );
}
```