Final Project Documentation

Application Name: War Card Game

Author: Kody Owens

Date Finished:

---

1. Project Overview

1.1 Problem Statement

Provide a 2-3 sentence description of the problem your application solves or the need it fulfills.

My app provides a way to play the card game “War” on the go, without having to carry around a physical deck of cards everywhere. It pits two players against each other via one phone.

1.2 Target Audience

Describe the intended user of your application.

This app is designed to be used by those who enjoy card games, but might not always have the ability to carry physical cards around.

1.3 Core Features

Provide a bulleted list of the main features of your application.

- Feature 1: The user can play the card game “War”
- Feature 2: The app can store and display the game statistics (which player wins, how many times they’ve won, etc)
- Feature 3: The app saves the game data locally via shared_preferences
- Feature 4:

---

2. Technical Design & Architecture

2.1 State Management Strategy

Describe the state management approach you chose and why it was appropriate for your application's complexity.

My application utilizes the Provider package for application-wide state management. The reason I chose this package was due to its simplicity and scalability. It also helped keep the UI components nice and neat.

The Provider package is appropriate for the following reasons:

Includes a shared state.

Allows the state to be updated while also notifying the listening widgets.

2.2 Data Model

Describe the main data structures or classes in your application. Include a code block for your primary model(s).

The core data model is the CardModel class, which represents a playing card for the game. Each card has a suit (i.e., hearts, spades) and a rank (2-14, where 11-14 are Jack, Queen, King, Ace). It also includes the path necessary for rendering card images in the app.

//card_model.dart

```dart
class CardModel {
  final String suit;
  final int rank;

  CardModel({required this.suit, required this.rank});

  String get asset => 'assets/cards/${rank}_of_$suit.svg';
}
```

2.3 Persistence / API Strategy

Explain how your application handles data. If you used local persistence, describe the method and data format.

This application uses local persistence via the shared_preferences package to store game statistics between sessions.

Data stored includes:

Total number of games played

Player 1 win count

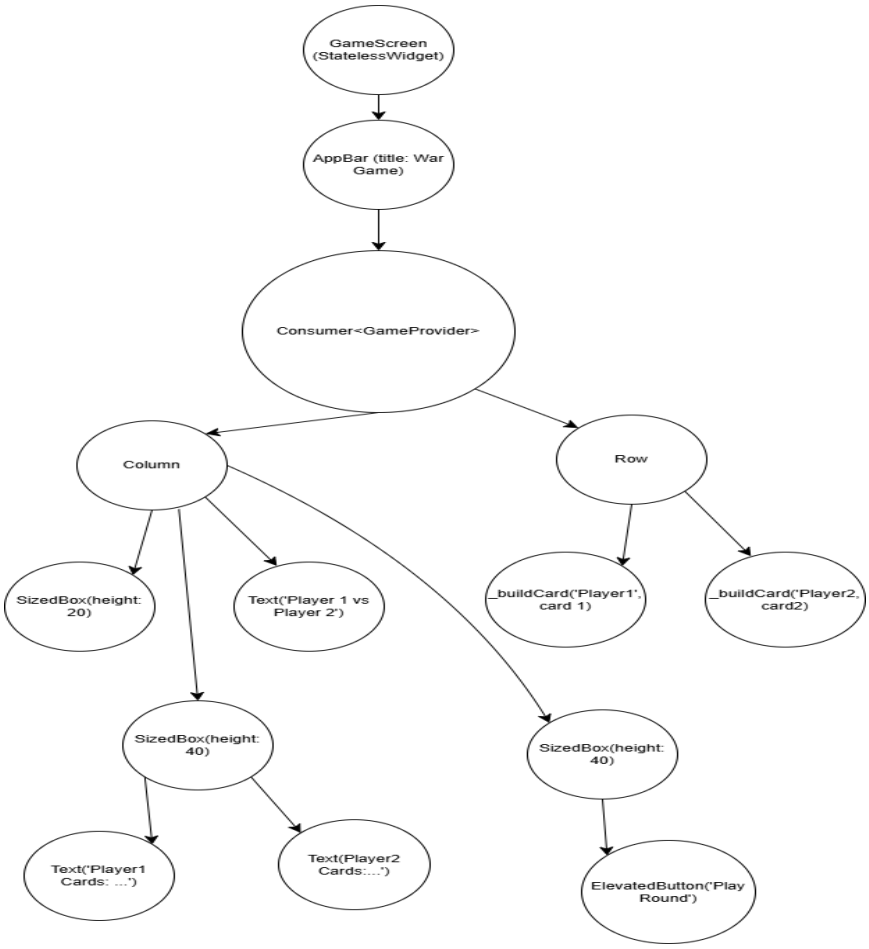
Player 2 win count

How it works:

When the game ends, the app will retrieve the results and update the winner's win count.

2.4 Widget Tree Diagram

Provide a draw.io diagram of the widget tree for one of your main screens.




---

3. Setup & Installation

3.1 Prerequisites

- Firebase Studio and Flutter SDK

3.2 Installation Steps

1. Clone the public repository you created: git clone <your-repo-url>
2. Navigate into the project directory: cd <your-project-directory>
3. Set up your Flutter files: flutter create
4. After adding dependencies: flutter pub get
5. Run the application: flutter run

---

4. Automated Testing

4.1 Testing Strategy

Briefly describe your approach to testing.

Testing to make sure the widgets work.

4.2 Test Case Examples

Provide at least one unit test and one widget test example from your project. For each, describe its purpose, the AAA steps, and include the code snippet.

Tests to see if the Home screen functions as intended.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:myapp/main.dart';
import 'package:myapp/game_screen.dart';

void main() {
  testWidgets('Home screen renders and navigates', (tester) async {
    await tester.pumpWidget(MyApp());

    expect(find.text('Start Game'), findsOneWidget);
    await tester.tap(find.text('Start Game'));
```

```
    await tester.pumpAndSettle();

    expect(find.byType(GameScreen), findsOneWidget);
  });
}
```

Test Case 2: Widget Test

Tests to see if the Widget for the Play Round button functions as intended.

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:myapp/game_provider.dart';
import 'package:myapp/game_screen.dart';
import 'package:provider/provider.dart';

void main() {
  testWidgets('Game screen shows intial UI elements', (tester) async {
    await tester.pumpWidget(
      ChangeNotifierProvider(
        create: (_) => GameProvider(),
        child: MaterialApp(home: GameScreen()),
      ),
    );
    // Looks for the Play Round button
    expect(find.text('Play Round'), findsOneWidget);
    // Looks for the Player1 vs Player2 title
    expect(find.text('Player1 vs Player2'), findsOneWidget);
    // Looks for when there are no cards
    expect(find.text('No Card'), findsNWidgets(2));
  });
}
```

---

5. User Guide

Provide simple, step-by-step instructions for using the core features of your app.

1. Click Start New Game.

2. Click Play Round until you are satisfied.
3. Once done playing, click on the upper left side to return to the home screen.