Project

Mobile Application Development

Name:M. Qamer Hassan
ID: F2021376057
Section: A1

Tic-Tac-Toe App

Code:

Main.dart:

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

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  final themeMode = await ThemePreferences.getThemeMode();
  runApp(MyApp(themeMode: themeMode));
}
```

```dart
class MyApp extends StatefulWidget {
  final ThemeMode themeMode;
  const MyApp({Key? key, required this.themeMode}) : super(key: key);
```

```dart
  @override
  State<MyApp> createState() => _MyAppState();
}
```

```dart
class _MyAppState extends State<MyApp> {
  late ThemeMode _themeMode;
```

```dart
  @override
  void initState() {
    super.initState();
    _themeMode = widget.themeMode;
  }
```

```dart
  void toggleTheme() async {
    final newTheme =
        _themeMode == ThemeMode.dark ? ThemeMode.light : ThemeMode.dark;
    setState(() {
      _themeMode = newTheme;
    });
    await ThemePreferences.setThemeMode(newTheme);
  }
```

```dart
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Tic Tac Toe',
      debugShowCheckedModeBanner: false,
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      themeMode: _themeMode,
```

```
      home: HomeScreen(toggleTheme: toggleTheme),
    );
  }
}
```

```
/// Handles theme persistence using SharedPreferences
class ThemePreferences {
  static const String key = "theme_mode";
```

```
  static Future<void> setThemeMode(ThemeMode mode) async {
    final prefs = await SharedPreferences.getInstance();
    prefs.setInt(key, mode == ThemeMode.dark ? 1 : 0);
  }
```

```
  static Future<ThemeMode> getThemeMode() async {
    final prefs = await SharedPreferences.getInstance();
    final mode = prefs.getInt(key) ?? 0;
    return mode == 1 ? ThemeMode.dark : ThemeMode.light;
  }
}
```

Home_screen.dart:

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

class HomeScreen extends StatefulWidget {
  final VoidCallback toggleTheme;
  const HomeScreen({Key? key, required this.toggleTheme}) : super(key: key);
```

```
  @override
  State<HomeScreen> createState() => _HomeScreenState();
}
```

```
class _HomeScreenState extends State<HomeScreen> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController player1Controller = TextEditingController();
  final TextEditingController player2Controller = TextEditingController();
```

```
  @override
  Widget build(BuildContext context) {
    bool isDark = Theme.of(context).brightness == Brightness.dark;
```

```
    return Scaffold(
      backgroundColor: isDark ? Colors.black : Colors.blue.shade900,
      appBar: AppBar(
        title: Text("Tic Tac Toe"),
        actions: [
          IconButton(
            icon: Icon(isDark ? Icons.light_mode : Icons.dark_mode),
            onPressed: widget.toggleTheme,
```

```
        )
      ],
    ),
    body: Stack(
      children: [
        Container(
          decoration: BoxDecoration(
            image: DecorationImage(
              image: AssetImage("assets/background.jpg"), // Add the image
              fit: BoxFit.cover,
            ),
          ),
        ),
        Center(
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  "Enter Players' Names",
                  style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.bold,
                      color: Colors.white),
                ),
                SizedBox(height: 20),
                _buildPlayerInput("Player 1 Name", player1Controller),
                _buildPlayerInput("Player 2 Name", player2Controller),
                SizedBox(height: 20),
                ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState!.validate()) {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => GameScreen(
                            player1: player1Controller.text,
                            player2: player2Controller.text,
                          ),
                        ),
                      );
                    }
                  },
                  child: Text("Start Game", style: TextStyle(fontSize: 24)),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
```

```
    );
  }
```

```
  Widget _buildPlayerInput(String hint, TextEditingController controller) {
    return Padding(
      padding: EdgeInsets.all(15),
      child: TextFormField(
        controller: controller,
        style: TextStyle(color: Colors.white),
        decoration: InputDecoration(
          focusedBorder:
              OutlineInputBorder(borderSide: BorderSide(color: Colors.white)),
          hintText: hint,
          hintStyle: TextStyle(color: Colors.white),
        ),
        validator: (value) =>
            value == null || value.isEmpty ? "Please enter $hint" : null,
      ),
    );
  }
}
```

Game_screen.dart:

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

class GameScreen extends StatefulWidget {
  final String player1;
  final String player2;
```

```
  GameScreen({required this.player1, required this.player2});
```

```
  @override
  _GameScreenState createState() => _GameScreenState();
}
```

```
class _GameScreenState extends State<GameScreen> {
  late List<List<String>> board;
  late String currentPlayer;
  late bool gameOver;
  late String winner;
```

```
  @override
  void initState() {
    super.initState();
    _resetGame();
  }
```

```
  void _resetGame() {
    setState(() {
```

```
      board = List.generate(3, (_) => List.generate(3, (_) => ""));
      currentPlayer = "X";
      gameOver = false;
      winner = "";
    });
  }
```

```
  void _makeMove(int row, int col) {
    if (board[row][col] != "" || gameOver) return;
```

```
    setState(() {
      board[row][col] = currentPlayer;
```

```
      if (Utils.checkWinner(board, currentPlayer)) {
        winner = currentPlayer == "X" ? widget.player1 : widget.player2;
        gameOver = true;
      } else if (Utils.isBoardFull(board)) {
        winner = "It's a tie!";
        gameOver = true;
      } else {
        currentPlayer = currentPlayer == "X" ? "O" : "X";
      }
    });
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          Positioned.fill(
            child: Image.asset(
              'assets/background2.jpg',
              fit: BoxFit.cover,
            ),
          ),
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                gameOver
                    ? "Winner: $winner"
                    : "Turn: ${currentPlayer == 'X' ? widget.player1 :
widget.player2}",
                style: TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                    color: Colors.white),
              ),
              SizedBox(height: 20),
              _buildBoard(),
              SizedBox(height: 20),
              ElevatedButton(
```

```
              onPressed: _resetGame,
              style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
              child: Text("Restart Game",
                  style: TextStyle(fontSize: 22, color: Colors.white)),
            ),
          ],
        ),
      ],
    ),
  );
 }
```

```
 Widget _buildBoard() {
   return Column(
     children: List.generate(3, (row) {
       return Row(
         mainAxisAlignment: MainAxisAlignment.center,
         children: List.generate(3, (col) {
           return GestureDetector(
             onTap: () => _makeMove(row, col),
             child: Container(
               width: 80,
               height: 80,
               margin: EdgeInsets.all(5),
               decoration: BoxDecoration(
                 color: Colors.white.withOpacity(0.8),
                 borderRadius: BorderRadius.circular(10),
               ),
               alignment: Alignment.center,
               child: Text(
                 board[row][col],
                 style: TextStyle(
                     fontSize: 40,
                     fontWeight: FontWeight.bold,
                     color: Colors.black),
               ),
             ),
           );
         }),
       );
     }),
   );
 }
}
```

Utils.dart:

```
class Utils {
  static bool checkWinner(List<List<String>> board, String player) {
    for (int i = 0; i < 3; i++) {
```

```
      if (board[i][0] == player &&
          board[i][1] == player &&
          board[i][2] == player) return true;
      if (board[0][i] == player &&
          board[1][i] == player &&
          board[2][i] == player) return true;
    }
    if (board[0][0] == player && board[1][1] == player && board[2][2] == player)
      return true;
    if (board[0][2] == player && board[1][1] == player && board[2][0] == player)
      return true;
    return false;
  }

  static bool isBoardFull(List<List<String>> board) {
    return board.every((row) => row.every((cell) => cell.isNotEmpty));
  }
}
```

Assets:

Pubspec.yaml:

```
name: tic_tac_toe
description: A simple Tic-Tac-Toe game in Flutter.
publish_to: 'none'

version: 1.0.0+1
```

```
environment:
  sdk: '>=2.17.0 <3.0.0'
```

```
dependencies:
  flutter:
    sdk: flutter
```

```
  cupertino_icons: ^1.0.2
  shared_preferences: ^2.3.1
```

```
dev_dependencies:
  flutter_test:
```

```
    sdk: flutter

  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true

  assets:
    - assets/background.jpg
    - assets/background2.jpg
```

Screenshots: