Sri Lanka Institute of Information Technology

“Wordly”: Word Guessing App

Report

IT19147024

Kariyawasam K.G.S.S.K

2022_REG_08

Current Trends in Software Engineering – SE4010

B.Sc. (Hons’) in Information Technology

Specializing in Software Engineering

Table of Contents

List of Figures

1. Project Description

"Wordly" is a quiz-based word guessing game developed as a mobile app. The app was developed using Flutter and Firebase.

A normal user can create an account and answer quizzes. In the quizzes, the user needs to select the correct answer(s) for the given definition. User will earn points based on the number of right answers in a quiz and the points will be added to user's total points in the system. There is a leaderboard where the users can see their rank based on their total points among all the other active users. Users can use their profile to view their details and total points they have earned. Users can use this to edit their details too. Also, users can always add reviews to give feedback about the app. Admins can add word definitions and answers to the database through the app. One word definition has four answers, where one or multiple of them can be right. Admins can make normal users Admins or delete them. Admins can also view all the user feedback.

2. Implemented Functionalities

All the code related to the main functionalities implemented by Kariyawasam K.G.S.S.K (IT19147024) is available in the Appendices section of the report, or the complete project is available at https://github.com/rukshan99/wordly .

2.1 Display definition list

This is a "read" operation. An Admin can view all the definitions as a list after logging in to the admin account. The definition list item consists of the definition word, similar words and a delete button. Admin can also use the case-insensitive search bar to search for definitions using the full or partial text of either answers.

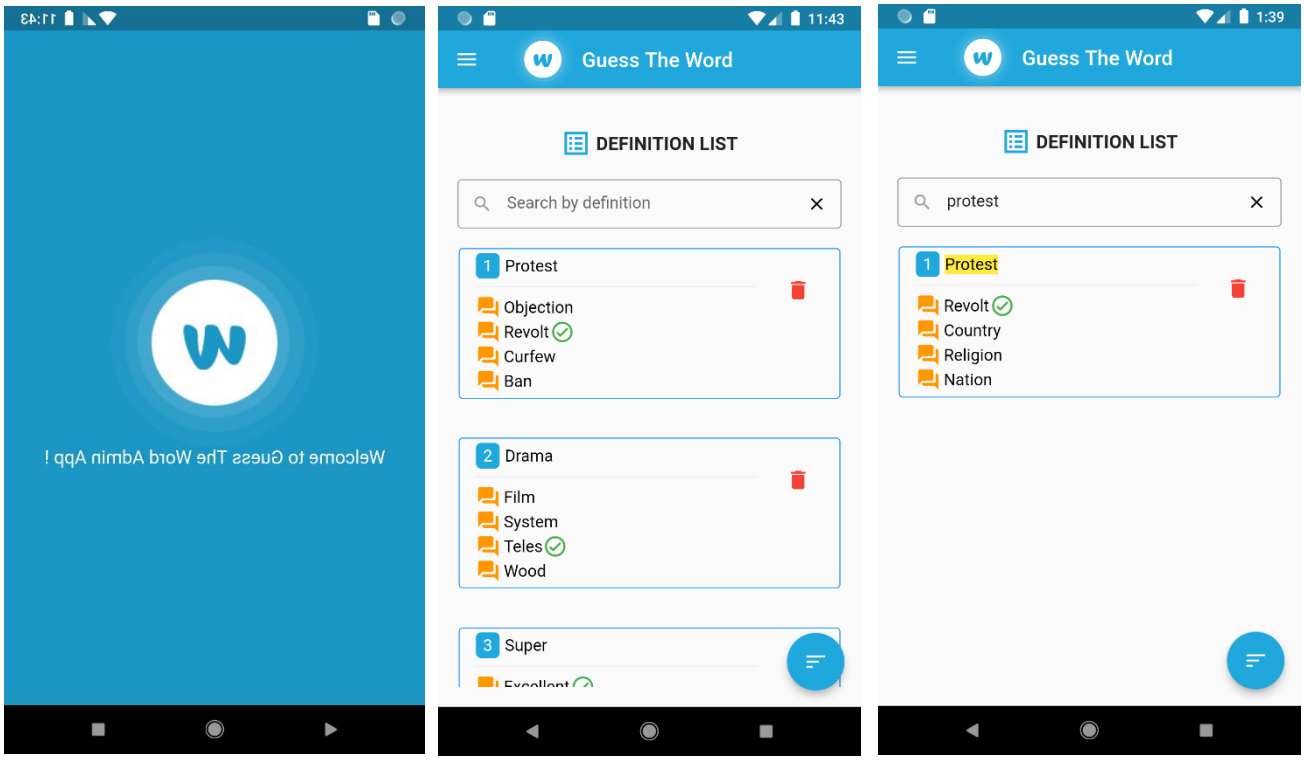

Figure 2.1Definition Splash screen, Definition list screen, Search Screen

2.2 Add definition details

This is a “create” operation. When registering with Firebase Authentication, only the definition and answers of the users are saved to a special “definitions” collection in the Authentication Service.

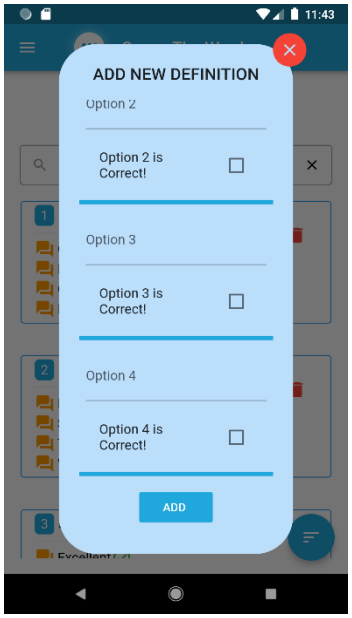

Figure 2.2 Definition document structure

2.3 Add definition details to the “definitions” collection in Cloud Firestore

This is a “create” operation. When registering with Firebase Authentication, only the definition and answers of the users are saved to a special “definitions” collection in the Authentication Service. To handle other information with much ease, a separate collection was created in the Cloud Firestore database with the same name of “definitions” and the data is saved to this collection when a user is registering. As illustrated in *figure 2.2* a user document consists of definition word question (String), options (String), id (null), and answers (Array). craetedDateTime (Timestamp) fields. The field “options” was used to give the answers for definition word. The field “answers” was used to give the correct answer or incorrect answer.

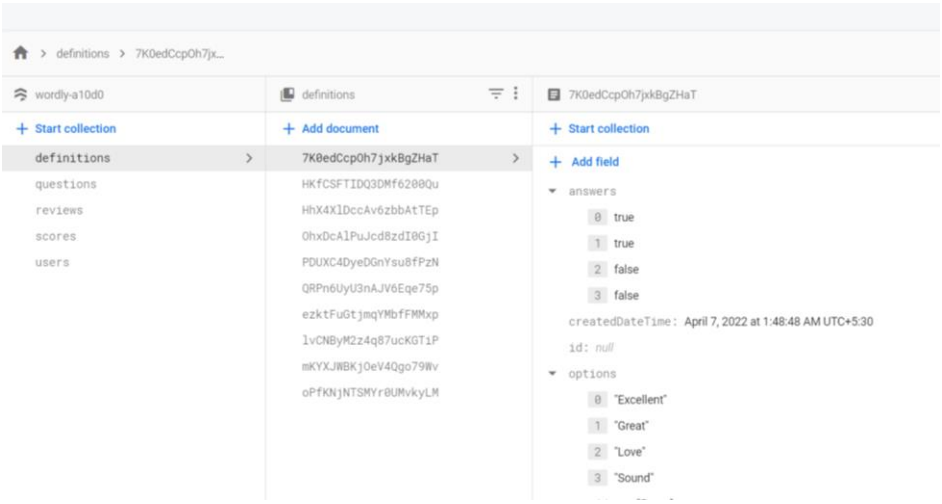

Figure 2.3Definition document structure

2.4 Update definition

This is a “read” operation. An Admin can view all the registered users as a list after logging in to the Admin account. The user list item consists of the name and email and a delete button. A user list item also responds to tapings by opening a user profile card. Admin can also use the case-insensitive search bar to search for registered users using the full or partial text of either name or email.

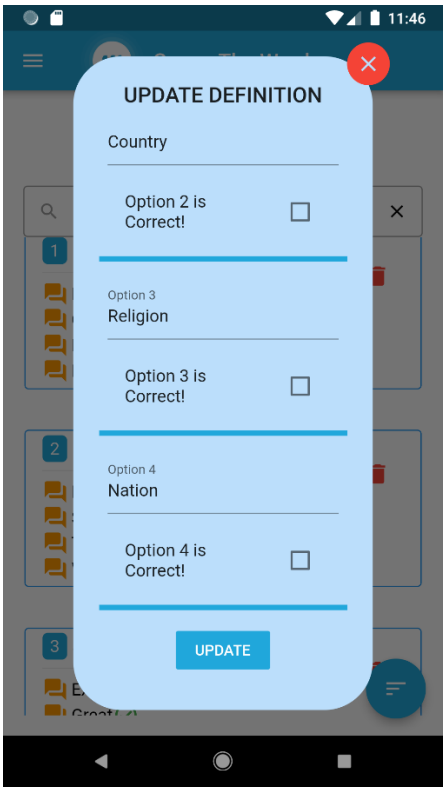

Figure 2.4Update

2.5 Delete definition

This is a “delete” operation. An admin can delete definition from the definition list screen. Two methods were implemented to do the delete operation on definitions. Admin can either use the delete icon button on each definition list item for a normal definition deletion or swipe a definition list item to the left or right for quick deletion. When using normal deletion with the icon button, a warning alert will always pop up to make sure that the icon was not taped accidentally by the Admin. With the quick deletion method, a warning alert will pop up only in the initial successful deletion.

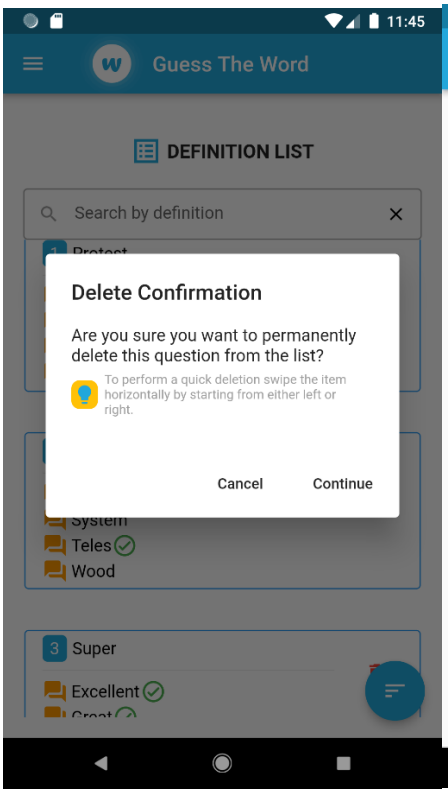

Figure 2.5 Normal delete warning

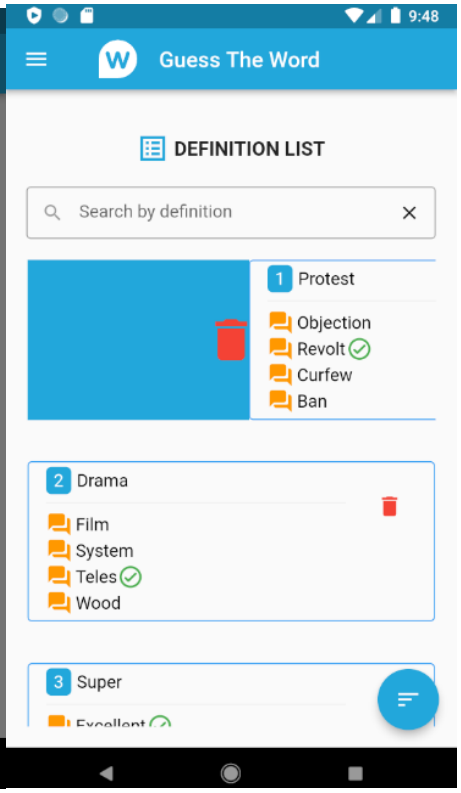

Figure 2.6 Quick delete

3. References

- Flutter documentation: https://docs.flutter.dev/
- Firebase documentation: https://firebase.flutter.dev/

4. Appendices

4.1 Main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:wordly/providers/user_provider.dart';
import 'package:wordly/screens/leaderboard.dart';
import 'package:wordly/screens/login.dart';
import 'package:wordly/screens/register.dart';
import 'package:wordly/screens/splash.dart';
import 'package:wordly/screens/welcome.dart';
import 'package:wordly/screens/admin_users.dart';
import 'package:wordly/screens/definitions.dart';
import 'package:wordly/screens/quiz.dart';
import 'package:wordly/screens/home.dart';
import 'package:wordly/screens/definition_welcomesplash.dart';
import 'package:wordly/screens/review.dart';
import 'package:wordly/screens/reviewList.dart';
import 'package:firebase_auth/firebase_auth.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key key}) : super(key: key);

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

    return ChangeNotifierProvider(
        create: (context) => UserProvider(),
        child: MaterialApp(
          debugShowCheckedModeBanner: false,
          theme: ThemeData(
            primarySwatch: Colors.blue,
          ),
          home: _auth.currentUser == null
```

```
            ? const SplashScreen()
            : (_auth.currentUser.email == 'admin@gmail.com'
                ? const UserList()
                : const HomeScreen()),
          routes: <String, WidgetBuilder>{
            "login": (BuildContext context) => const LoginScreen(),
            "register": (BuildContext context) => const RegisterScreen(),
            "welcome": (BuildContext context) => const WelcomeScreen(),
            "userList": (BuildContext context) => const UserList(),
            "definitionList": (BuildContext context) => definitionList(),
            "home": (BuildContext context) => const HomeScreen(),
            "quiz": (BuildContext context) => QuizScreen(),
            "leaderboard": (BuildContext context) => const LeaderBoard(),
            "definitionAdminWelcome": (BuildContext context) =>
                DefinitionAdminWelcomeSplashScreen(),
            "review": (BuildContext context) => const ReviewScreen(),
            "reviewList": (BuildContext context) => const ReviewListScreen(),
          },
        ));
  }
}
```

4.2 Models/ definition.dart

```
// @dart=2.9
import 'package:cloud_firestore/cloud_firestore.dart';


class Definition {
  final String question;
  final List<String> options;
  final List<bool> answers;
  final DateTime createdDateTime;


  DocumentReference id;
    //Definition(question: "", options: <String>[], answers: <bool>[])
        //as DocumentReference<Definition>;
```

```
Definition(
    {this.question,
    this.options,
    this.answers,
    this.id,
    this.createdDateTime});

factory Definition.fromJson(
    Map<String, dynamic> json, DocumentReference docRef) {
  print(json['question']);
  Timestamp createdTimeStamp = json['createdDateTime'];
  return Definition(
    id: docRef,
    question: json['question'] as String,
    options: List<String>.from(json['options'].map((x) => x)),
    answers: List<bool>.from(json['answers'].map((x) => x)),
    createdDateTime:
        createdTimeStamp != null ? createdTimeStamp.toDate() : null,
  );
}

Map<String, dynamic> toMap() {
  return {
    "createdDateTime": createdDateTime,
    "question": question,
    "options": options,
    "answers": answers,
    "id": id
```

```
    };
  }
}
```

4.3 Controllers/definition_controller.dart

```
// @dart=2.9

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:wordly/models/definitions.dart';



class DefinitionController {
  final FirebaseFirestore _db = FirebaseFirestore.instance;

  // Create a CollectionReference that references the firestore collection
  CollectionReference collectionRef =
      FirebaseFirestore.instance.collection('definitions');

// get all definitions
  Stream<List<Definition>> getDefinitions() {
    return _db.collection('definitions').snapshots().map((snapshot) =>
        snapshot
            .docs
            .map((doc) => Definition.fromJson(doc.data(), doc.reference))
            .toList());
  }

  getAllDefinitions() {
```

```
    return _db.collection('definitions').snapshots();
  }


//add definition , options and correct answers
  addDefinition(Definition definitionObj) async {
    try {
      _db.runTransaction((Transaction transaction) async {
        await _db.collection('definitions').doc().set(definitionObj.toMap());
      });
    } catch (e) {
      // ignore: avoid_print
      print(e.toString());
    }
  }

  // //update definitions, options and correct answers.
  updateDefinition(Definition definitionObj, String question,
      List<String> optionsList, List<bool> answerList) {
    try {
      _db.runTransaction((transaction) async {
        await transaction.update(definitionObj.id, {
          'question': question,
          'options': optionsList,
          'answers': answerList
        });
      });
    } catch (e) {
```

```
      // ignore: avoid_print
      print(e.toString());
    }
  }

  // // delete definition
  deleteQuestion(Definition definition) {
    _db.runTransaction((Transaction transaction) async {
      await transaction.delete(definition.id);
    });
  }


  // ///Deletes All the questions
  deleteAllQuestions(List<Definition> definitionsList) {
    _db.runTransaction((Transaction transaction) async {
      definitionsList.forEach((definition) {
        transaction.delete(definition.id);
      });
    });
  }

}
```

4.4 Widgets/header_container.dart

```
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:wordly/utils/animated_background.dart';
import 'package:wordly/utils/animated_wave.dart';
```

```
import 'package:wordly/utils/color.dart';

// ignore: must_be_immutable
class HeaderContainer extends StatelessWidget {
  var text = "Login";

  HeaderContainer(this.text, {Key? key}) : super(key: key);

  onBottom(Widget child) => Positioned.fill(
        child: Align(
          alignment: Alignment.bottomCenter,
          child: child,
        ),
      );

  @override
  Widget build(BuildContext context) {
    return Container(
      height: MediaQuery.of(context).size.height * 0.38,
      child: Stack(
        alignment: Alignment.center,
        children: <Widget>[
          const Positioned(child: AnimatedBackground()),
          onBottom(const AnimatedWave(
            height: 120,
            speed: 1.0,
          )),
          onBottom(const AnimatedWave(
            height: 60,
            speed: 0.9,
            offset: pi,
          )),
          onBottom(const AnimatedWave(
            height: 160,
            speed: 1.2,
            offset: pi / 2,
          )),
          Positioned(
              bottom: 15,
              child: Text(
                text,
                textAlign: TextAlign.center,
                style: const TextStyle(color: Colors.white, fontSize: 20),
              )),
```

```
          Center(
            child: Image.asset("assets/img/Logo.png"),
          ),
        ],
      ),
    );
  }
}
```

4.5 Screens/definition_welcome.dart

```
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:wordly/utils/animated_background.dart';
import 'package:wordly/utils/animated_wave.dart';
import 'package:wordly/utils/color.dart';

// ignore: must_be_immutable
class HeaderContainer extends StatelessWidget {
  var text = "Login";

  HeaderContainer(this.text, {Key? key}) : super(key: key);

  onBottom(Widget child) => Positioned.fill(
        child: Align(
          alignment: Alignment.bottomCenter,
          child: child,
        ),
      );

  @override
```

```
Widget build(BuildContext context) {
  return Container(
    height: MediaQuery.of(context).size.height * 0.38,
    child: Stack(
      alignment: Alignment.center,
      children: <Widget>[
        const Positioned(child: AnimatedBackground()),
        onBottom(const AnimatedWave(
          height: 120,
          speed: 1.0,
        )),
        onBottom(const AnimatedWave(
          height: 60,
          speed: 0.9,
          offset: pi,
        )),
        onBottom(const AnimatedWave(
          height: 160,
          speed: 1.2,
          offset: pi / 2,
        )),
        Positioned(
            bottom: 15,
            child: Text(
              text,
              textAlign: TextAlign.center,
              style: const TextStyle(color: Colors.white, fontSize: 20),
            )),
```

```
          Center(
            child: Image.asset("assets/img/Logo.png"),
          ),
        ],
      ),
    );
  }
}
```

4.6 Screens/main_drawer.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:wordly/screens/admin_users.dart';
import 'package:wordly/utils/color.dart';
import 'package:wordly/screens/definitions.dart';
import 'package:wordly/screens/definition_welcomesplash.dart';

import '../screens/leaderboard.dart';

class MainDrawer extends StatelessWidget {
  const MainDrawer({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final FirebaseAuth _auth = FirebaseAuth.instance;
    final user = _auth.currentUser;
    // final isAdmin = user!.email == 'admin@gmail.com';
    const isAdmin = true;
```

```
navigateLogin() async {
  Navigator.pushReplacementNamed(context, "login");
}

navigateToReview() async {
  Navigator.pushReplacementNamed(context, "review");
}

navigateToReviewList() async {
  Navigator.pushReplacementNamed(context, "reviewList");
}

return Drawer(
  child: Column(
    children: [
      Container(
        width: double.infinity,
        padding: const EdgeInsets.all(20),
        color: purpleLightColors,
        child: Center(
          child: Column(
            children: [
              Container(
                width: 100,
                height: 100,
                margin: const EdgeInsets.only(top: 30, bottom: 30),
                decoration: const BoxDecoration(
```

```
            shape: BoxShape.circle,
            image: DecorationImage(
              image: AssetImage('assets/img/user.png'),
            ),
          ),
        ),
        user != null
            ? Text(
                user.email!,
                style: TextStyle(
                    color: white,
                    fontSize: 18,
                    fontFamily: 'Righteous'),
              )
            : const Text(''),
      ],
    ),
  ),
),
if (isAdmin)
  ListTile(
    leading: const Icon(Icons.list),
    title: const Text('Definition List'),
    onTap: () async {
      _auth.authStateChanges().listen((event) {
        Navigator.push(
            context,
            MaterialPageRoute(
```

```
            builder: (context) =>
                DefinitionAdminWelcomeSplashScreen()));
      });
    },
  ),
if (isAdmin)
  ListTile(
    leading: const Icon(Icons.supervised_user_circle_outlined),
    title: const Text('User list'),
    onTap: () {
      Navigator.push(context,
          MaterialPageRoute(builder: (context) => const UserList()));
    },
  ),
ListTile(
  leading: const Icon(Icons.leaderboard_outlined),
  title: const Text('Leaderboard'),
  onTap: () {
    Navigator.push(context,
        MaterialPageRoute(builder: (context) => const LeaderBoard()));
  },
),
if (!isAdmin)
  ListTile(
    leading: const Icon(Icons.star_half_rounded),
    title: const Text('Add review'),
    onTap: () => {navigateToReview()},
  ),
```

```
        if (isAdmin)
          ListTile(
            leading: const Icon(Icons.reviews_sharp),
            title: const Text('Review List'),
            onTap: () => {navigateToReviewList()},
          ),
        ListTile(
          leading: const Icon(Icons.exit_to_app),
          title: const Text('Sign out'),
          onTap: () async {
            await _auth.signOut().then((value) => navigateLogin());
          },
        ),
      ],
    ),
  );
 }
}
```