Sri Lanka Institute of Information Technology

“Wordly”: Word Guessing App

Report

IT19129204

Jayasekara R.T.R

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 Jayasekara R.T.R (IT19129204) is available in the Appendices section of the report, or the complete project is available at https://github.com/rukshan99/wordly .

2.1 Login and register with Firebase Authentication

The login and registration functionalities were implemented using the Firebase Authentication service which leverages industry standards like OAuth 2.0 and OpenID Connect. Among the main authentication methods available, email and password-based authentication was used given the simplicity of the mobile app. Instead of default Firebase UIs custom UIs were implemented for login and user registration screens with a complimentary welcome screen. After a successful login, a function was implemented to check if the user's email is "admin@gmail.com" which is the default Admin email address or the "isAdmin" field of the user is true, to decide whether to give the logged-in user Admin privileges or not.

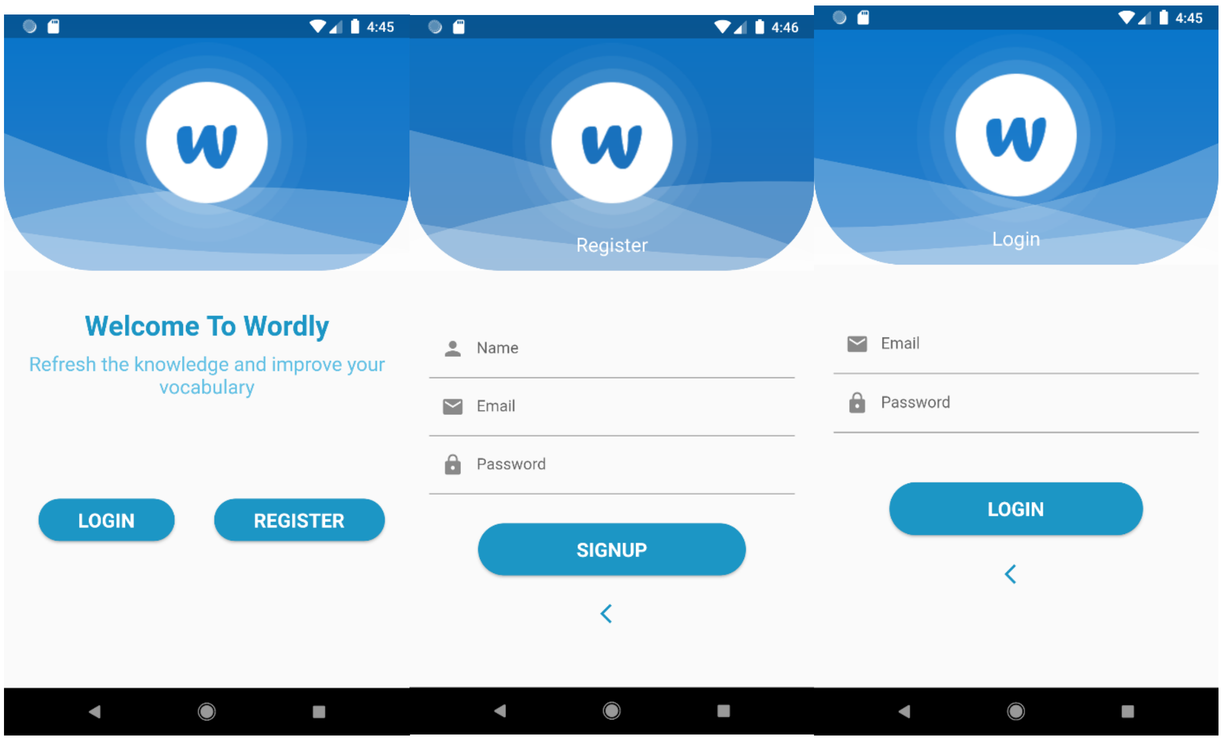
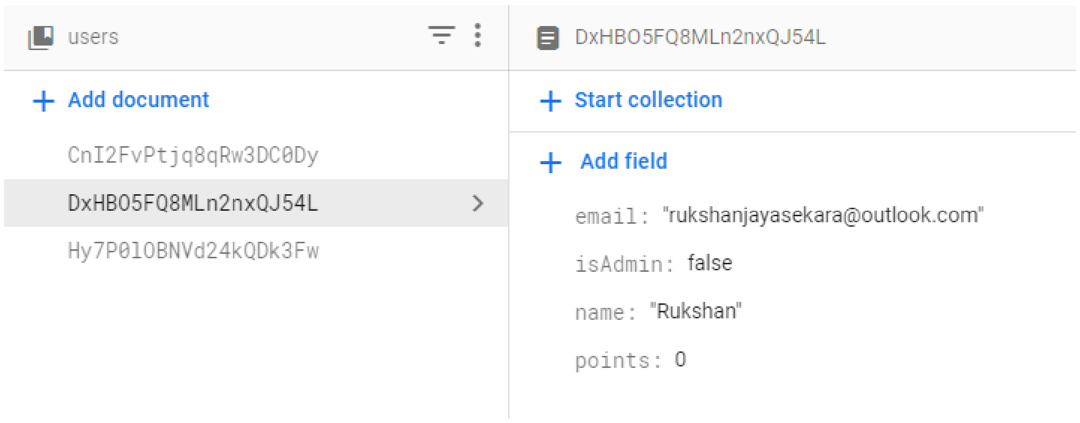


Figure 2.1 Welcome screen, Register screen and Login screen

2.2 Add user details to the “users” collection in Cloud Firestore

This is a “create” operation. When registering with Firebase Authentication, only the email and password of the users are saved to a special “users” 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 “users” and the data is saved to this collection when a user is registering. As illustrated in *figure 2.2* a user document consists of email (String), isAdmin (Boolean), name (String), and points (Number) fields. The field “isAdmin” was used to handle the role of the users with the “change a normal user to an Admin user” functionality of the application which is discussed later in this report. The default or initial value of this field is false which means a normal user. The field “points” was used to store the number of points earned by a user when attempting the quizzes. The default or initial value of this field is set to be 0. The field “email” has a unique value for each user, which is ensured by the Firebase Authentication service in the user registration process.



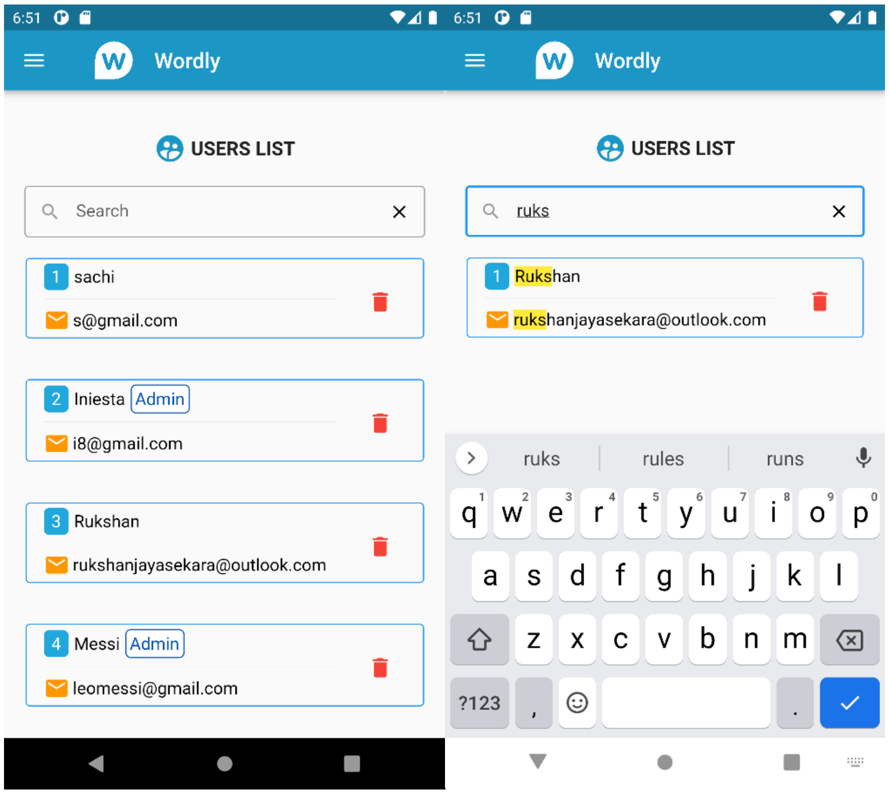
CnI2FvPtjq8qRw3DC0Dy
DxHBO5FQ8MLn2nxQJ54L
Hy7P01OBNVd24kQDk3Fw

email: "rukshanjayasekara@outlook.com"
isAdmin: false
name: "Rukshan"
points: 0

Figure 2.2 User document structure

2.3 Display registered users list

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.



1 sachi
s@gmail.com

2 Iniesta Admin
i8@gmail.com

3 Rukshan
rukshanjayasekara@outlook.com

4 Messi Admin
leomessi@gmail.com

1 Rukshan
rukshanjayasekara@outlook.com

Figure 2.3 User list

2.4 Change a normal user to an Admin user

This is an "update" operation. An admin has the option to change the user roles as needed. To do this the Admin needs to go to the user list screen and select a user. Then, the user profile card will be opened, and in that there is a toggle switch that can be used to switch the user roles between the Admin user role and the normal user role. According to the value of the toggle switch, the value of the "isAdmin" field of the correspondent user will be updated.

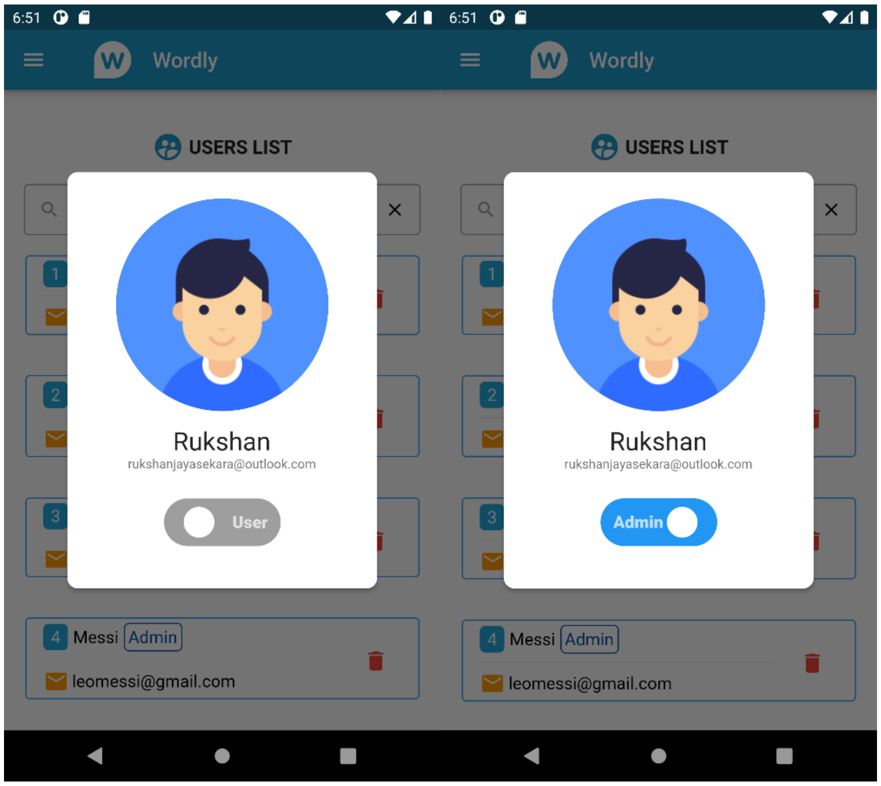
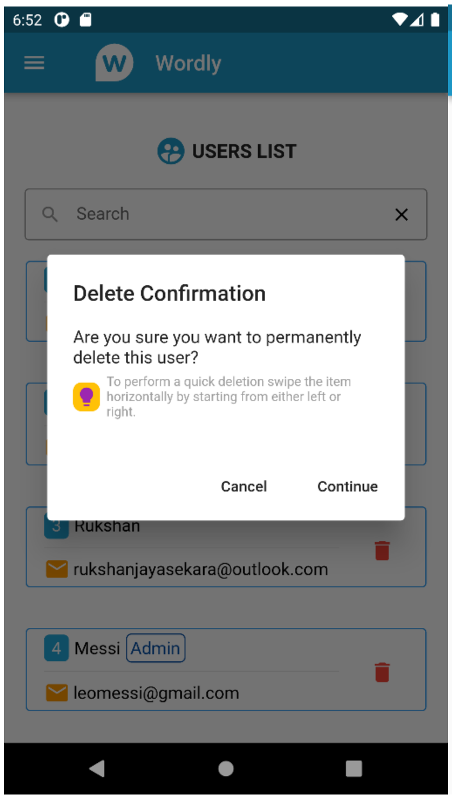


Figure 2.4 Update Admin status

2.5 Delete user

This is a "delete" operation. An admin can delete users from the user list screen. Two methods were implemented to do the delete operation on users. Admin can either use the delete icon button on each user list item for a normal user deletion or swipe a user 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.



Are you sure you want to permanently delete this user?

To perform a quick deletion swipe the item horizontally by starting from either left or right.

Figure 2.5 Normal delete warning

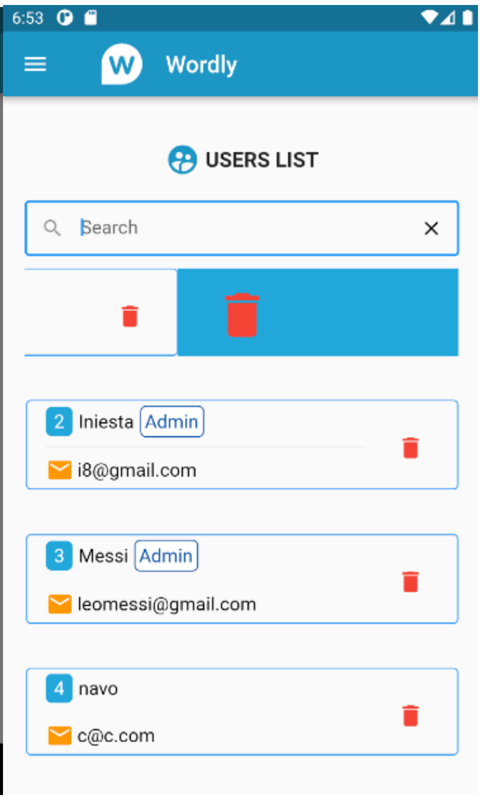
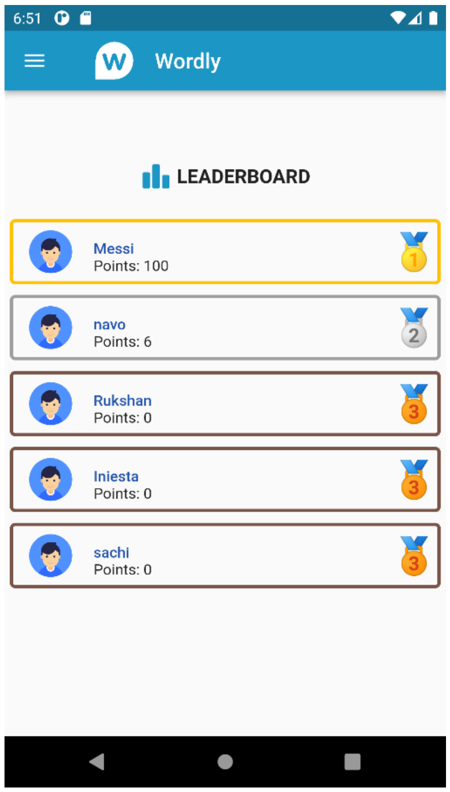


Figure 2.6 Quick delete

2.6 Display a user leaderboard

This is a “read” operation. Wordly users can see the rankings of all the users as a leaderboard. The ranks are calculated based on the total points scored on the quizzes. This was implemented by descendingly ordering users by the value of the “points” field of the user documents in the “users” collection (*see figure 2.2*).



Messi
Points: 100

navo
Points: 6

Rukshan
Points: 0

Iniesta
Points: 0

sachi
Points: 0

Figure 2.7 Leaderboard

3. References

- Flutter documentation: https://docs.flutter.dev/
- Firebase documentation: https://firebase.flutter.dev/
- Background animations: https://felixblaschke.medium.com/fancy-background-animations-in-flutter-4163d50f5c37

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/user.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';

class User {
  final String name;
  final String email;
  final int points;
  final bool isAdmin;

  User({
    required this.name,
    required this.email,
    required this.points,
    required this.isAdmin
  });
  factory User.fromJson(Map<String, dynamic> json, DocumentReference docRef) {
    // ignore: avoid_print
    print(json['user']);
    return User(
        name: json['name'] as String,
        email: json['email'] as String,
        points: json['points'] as int,
        isAdmin: json['isAdmin'] as bool);
  }

  Map<String, dynamic> toMap() {
    return {"name": name, "email": email, "points": points, "isAdmin": isAdmin};
  }
}
```

4.3 Providers/user_provider.dart

```
import 'package:flutter/material.dart';
import '../controllers/user_controller.dart';

class UserProvider extends ChangeNotifier {
  bool _isAdmin = false;

  bool get isAdmin => _isAdmin;

  final userController = UserController();

  void update(bool val) {
    _isAdmin = val;
    notifyListeners();
  }
}
```

4.4 Controllers/user_controller.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:wordly/models/user.dart';

class UserController {
  final FirebaseFirestore _db = FirebaseFirestore.instance;

  // Create a CollectionReference that references the firestore collection
```

```
CollectionReference collectionRef =
    FirebaseFirestore.instance.collection('users');

// Get all users
Stream<List<User>> getUsers() {
  return _db.collection('users').snapshots().map((snapshot) => snapshot.docs
      .map((doc) => User.fromJson(doc.data(), doc.reference))
      .toList());
}

getAllUsers() {
  return _db.collection('users').snapshots();
}

// Add a user
addUser(User userObj) async {
  try {
    _db.runTransaction((Transaction transaction) async {
      await _db.collection('users').doc().set(userObj.toMap());
    });
  } catch (e) {
    // ignore: avoid_print
    print(e.toString());
  }
}

// Delete a user
// --> need to handle the deleted user from firebase auth
deleteUser(String email) {
  collectionRef.where('email', isEqualTo: email).get().then((snapshot) => {
        _db.runTransaction((Transaction transaction) async {
          snapshot.docs.forEach(
              (DocumentSnapshot doc) => {transaction.delete(doc.reference)});
        })
      });
}

updateIsAdmin(DocumentReference docRef, bool isAdmin) {
  return docRef
      .update({'isAdmin': isAdmin})
      .then((value) => print("User was given Admin privileges"))
      .catchError((error) => print("Failed to update user to Admin: $error"));
}
```

```
  Future checkIsAdmin(String? email) async {
    if (email == null || email == 'admin@gmail.com') return false;

    bool isAdmin = false;
    var snapshots = await collectionRef.where('email', isEqualTo: email).get();
    snapshots.docs.forEach((doc) {
      //print('$doc.data()');
      isAdmin = doc.get('isAdmin');
    });
    return isAdmin;
  }
}
```

4.5 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.jpg"),
          ),
        ],
      ),
    );
  }
}
```

4.6 Screens/welcome.dart

```
import 'package:flutter/material.dart';
import 'package:wordly/utils/color.dart';
import 'package:wordly/widgets/header_container.dart';

class WelcomeScreen extends StatefulWidget {
  const WelcomeScreen({Key? key}) : super(key: key);

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

```
class _WelcomeScreenState extends State<WelcomeScreen> {
  navigateToLogin() async {
    Navigator.pushReplacementNamed(context, "login");
  }

  navigateToRegister() async {
    Navigator.pushReplacementNamed(context, "register");
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: Container(
        padding: const EdgeInsets.only(bottom: 30),
        child: Column(
          children: <Widget>[
            HeaderContainer(''),
            Expanded(
              flex: 1,
              child: Container(
                margin: const EdgeInsets.only(left: 20, right: 20, top: 40),
                child: Column(
                  children: <Widget>[
                    RichText(
                      text: TextSpan(
                        text: 'Welcome To ',
                        style: TextStyle(
                            fontSize: 28.0,
                            fontWeight: FontWeight.bold,
                            color: purpleColors),
                        children: <TextSpan>[
                          TextSpan(
                            text: 'Wordly',
                            style: TextStyle(
                                fontSize: 28.0,
                                fontWeight: FontWeight.bold,
                                color: purpleColors),
                          ),
                        ],
                      ),
                    ),
                    const SizedBox(
                      height: 10.0,
                    ),
                    Text(
                      'Refresh the knowledge and improve your vocabulary',
                      textAlign: TextAlign.center,
                      style:
                          TextStyle(fontSize: 20.0, color: purpleLightColors),
                    ),
                    const SizedBox(
                      height: 100.0,
                    ),
                    Row(
                      children: <Widget>[
                        ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            primary: purpleColors, // background
                            onPrimary: Colors.white,
                            padding: const EdgeInsets.fromLTRB(40, 10, 40, 10),
                            shape: RoundedRectangleBorder(
                                borderRadius:
                                    BorderRadius.circular(100.0)), // foreground
                          ),
                          onPressed: navigateToLogin,
                          child: const Text(
                            'LOGIN',
                            style: TextStyle(
                                fontSize: 20.0, fontWeight: FontWeight.bold),
                          ),
                        ),
                        const SizedBox(
                          width: 30.0,
                        ),
                        ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            primary: purpleColors, // background
                            onPrimary: Colors.white,
                            padding: const EdgeInsets.fromLTRB(40, 10, 40, 10),
                            shape: RoundedRectangleBorder(
                                borderRadius:
                                    BorderRadius.circular(100.0)), // foreground
                          ),
                          onPressed: navigateToRegister,
                                            child: const Text(
                                              'REGISTER',
                                              style: TextStyle(
                                                  fontSize: 20.0, fontWeight: FontWeight.bold),
                                            ),
                                          ),
                                        ],
                                      ),
                                    ],
                                  ),
                                ),
                              )
                            ],
                          ),
                        ),
                      );
                     }
                    }
```

4.7 Screens/register.dart

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

import 'package:wordly/screens/home.dart';
import 'package:wordly/utils/color.dart';
import 'package:wordly/widgets/header_container.dart';
import 'package:wordly/controllers/user_controller.dart';
import 'package:wordly/models/user.dart';
import 'package:wordly/screens/admin_users.dart';

class RegisterScreen extends StatefulWidget {
  const RegisterScreen({Key? key}) : super(key: key);

  @override
  _RegisterScreenState createState() => _RegisterScreenState();
}

class _RegisterScreenState extends State<RegisterScreen> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final userController = UserController();
```

```
late String _name, _email, _password;
final int _points = 0;

checkAuthentification() async {
  _auth.authStateChanges().listen((user) async {
    bool isAdmin = await userController.checkIsAdmin(user?.email);
    if (user != null) {
      if (user.email == 'admin@gmail.com' || isAdmin) {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) => const UserList()));
      } else {
        Navigator.push(context,
            MaterialPageRoute(builder: (context) => const HomeScreen()));
      }
    }
  });
}

@override
void initState() {
  super.initState();
  this.checkAuthentification();
}

signUp() async {
  if (_formKey.currentState!.validate()) {
    _formKey.currentState!.save();

    try {
      UserCredential user = await _auth.createUserWithEmailAndPassword(
          email: _email, password: _password);
      if (user != null) {
        await _auth.currentUser!.updateProfile(displayName: _name);
        User userObj =
            User(name: _name, email: _email, points: _points, isAdmin: false);
        await userController.addUser(userObj);
      }
    } catch (e) {
      showError(e.toString());
    }
  }
}
```

```
showError(String errormessage) {
  showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('ERROR'),
          content: Text(errormessage),
          actions: <Widget>[
            TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
                child: const Text('OK')),
          ],
        );
      });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    body: Container(
      padding: const EdgeInsets.only(bottom: 30),
      child: Column(
        children: <Widget>[
          HeaderContainer("Register"),
          Expanded(
            flex: 1,
            child: Container(
              padding: const EdgeInsets.all(20.0),
              child: Form(
                key: _formKey,
                child: Column(
                  children: <Widget>[
                    const SizedBox(
                      height: 30.0,
                    ),
                    TextFormField(
                      // ignore: missing_return
                      validator: (input) {
                        if (input!.isEmpty) return 'Enter Name';
  },
  decoration: const InputDecoration(
      labelText: 'Name', prefixIcon: Icon(Icons.person)),
  onSaved: (input) => _name = input!,
),
TextFormField(
  // ignore: missing_return
  validator: (input) {
    if (input!.isEmpty) return 'Enter Email';
  },
  decoration: const InputDecoration(
      labelText: 'Email', prefixIcon: Icon(Icons.email)),
  onSaved: (input) => _email = input!,
),
TextFormField(
  // ignore: missing_return
  validator: (input) {
    if (input!.length < 6) {
      return 'Provide Minimum 6 Character Password';
    }
  },
  decoration: const InputDecoration(
      labelText: 'Password',
      prefixIcon: Icon(Icons.lock)),
  obscureText: true,
  onSaved: (input) => _password = input!,
),
const SizedBox(
  height: 50.0,
),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    primary: purpleColors, // background
    onPrimary: Colors.white,
    padding: const EdgeInsets.fromLTRB(100, 15, 100, 15),
    shape: RoundedRectangleBorder(
        borderRadius:
            BorderRadius.circular(100.0)), // foreground
  ),
  onPressed: signUp,
  child: const Text(
    'SIGNUP',
    style: TextStyle(
                              fontSize: 20.0,
                              fontWeight: FontWeight.bold,
                              color: Colors.white),
                        ),
                      ),
                      const SizedBox(
                        height: 15.0,
                      ),
                     IconButton(
                        icon: const Icon(Icons.arrow_back_ios),
                        color: purpleColors,
                        tooltip: 'Go Back',
                        onPressed: () {
                          Navigator.pushReplacementNamed(context, 'welcome');
                        },
                      ),
                    ],
                  ),
                ),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

4.8 Screens/login.dart

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

import 'package:wordly/screens/home.dart';
import 'package:wordly/utils/color.dart';
import 'package:wordly/widgets/header_container.dart';
import 'package:wordly/controllers/user_controller.dart';
import 'package:wordly/screens/admin_users.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final userController = UserController();
  late String _email, _password;

  checkAuthentification() async {
    _auth.authStateChanges().listen((user) async {
      bool isAdmin = await userController.checkIsAdmin(user?.email);
      if (user != null) {
        if (user.email == 'admin@gmail.com' || isAdmin) {
          Navigator.push(context,
              MaterialPageRoute(builder: (context) => const UserList()));
        } else {
          Navigator.push(context,
              MaterialPageRoute(builder: (context) => const HomeScreen()));
        }
      }
    });
  }

  @override
  void initState() {
    super.initState();
    this.checkAuthentification();
  }

  login() async {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();
      try {
        UserCredential user = await _auth.signInWithEmailAndPassword(
            email: _email, password: _password);
        bool isAdmin = await userController.checkIsAdmin(user.user?.email);
        if (user.user?.email == 'admin@gmail.com' || isAdmin) {
          Navigator.pushReplacementNamed(context, 'userList');
        } else {
          Navigator.pushReplacementNamed(context, 'home');
        }
      } catch (e) {
        showError(e.toString());
      }
    }
  }

  showError(String errormessage) {
    showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text('ERROR'),
            content: Text(errormessage),
            actions: <Widget>[
              TextButton(
                  onPressed: () {
                    Navigator.of(context).pop();
                  },
                  child: const Text('OK')),
            ],
          );
        });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: Container(
        padding: const EdgeInsets.only(bottom: 30),
        child: Column(
          children: <Widget>[
            HeaderContainer("Login"),
            Expanded(
              flex: 1,
              child: Container(
                padding: const EdgeInsets.all(20.0),
                child: Form(
                  key: _formKey,
                  child: Column(
                    children: <Widget>[
                      const SizedBox(
                        height: 30.0,
                      ),
                      TextFormField(
                        // ignore: missing_return
                        validator: (input) {
                          if (input == null || input.isEmpty) {
                            return 'Enter Email';
                          }
                        },
                        decoration: const InputDecoration(
                            labelText: 'Email', prefixIcon: Icon(Icons.email)),
                        onSaved: (input) => _email = input!,
                      ),
                      TextFormField(
                        // ignore: missing_return
                        validator: (input) {
                          if (input != null && input.length < 6) {
                            return 'Provide Minimum 6 Character Password';
                          }
                        },
                        decoration: const InputDecoration(
                            labelText: 'Password',
                            prefixIcon: Icon(Icons.lock)),
                        obscureText: true,
                        onSaved: (input) => _password = input!,
                      ),
                      const SizedBox(
                        height: 50.0,
                      ),
                      ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          primary: purpleColors, // background
                          onPrimary: Colors.white,
                          padding: const EdgeInsets.fromLTRB(100, 15, 100, 15),
                          shape: RoundedRectangleBorder(
                              borderRadius:
                                  BorderRadius.circular(100.0)), // foreground
                        ),
                        onPressed: login,
                        child: const Text(
                          'LOGIN',
                          style: TextStyle(
                              fontSize: 20.0,
                              fontWeight: FontWeight.bold,
                              color: Colors.white),
                        ),
                      ),
                      const SizedBox(
                        height: 15.0,
                      ),
                      IconButton(
                        icon: const Icon(Icons.arrow_back_ios),
                        color: purpleColors,
                        tooltip: 'Go Back',
                        onPressed: () {
                          Navigator.pushReplacementNamed(context, 'welcome');
                        },
                      ),
                    ],
                  ),
                ),
              ),
            )
          ],
        ),
        ),
        );
       }
      }
```

4.9 Screens/admin_users.dart

```
import 'dart:math';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_speed_dial/flutter_speed_dial.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:provider/provider.dart';
import 'package:wordly/providers/user_provider.dart';
import 'package:wordly/widgets/main_drawer.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:substring_highlight/substring_highlight.dart';
import 'package:flutter_switch/flutter_switch.dart';

import '../controllers/user_controller.dart';
import '../models/user.dart';

class UserList extends StatefulWidget {
  const UserList({Key? key}) : super(key: key);

  @override
  _UserListState createState() => _UserListState();
}

class _UserListState extends State<UserList> {
  final userController = UserController();
  bool isAdmin = false;

  final TextEditingController _searchController = TextEditingController();

  //Stores the complete users list
  List<QueryDocumentSnapshot> _resultsList = [];

  //Stores the filtered users based on the search criteria
  List<QueryDocumentSnapshot> _searchResultsList = [];

  //To notify when a revert has been made for a first time deletion
  bool _isRevertDeletion = false;

  @override
  void initState() {
    super.initState();
    _searchController.addListener(_onSearchChanged);
  }

  @override
  void dispose() {
    _searchController.removeListener(_onSearchChanged);
    _searchController.dispose();
    super.dispose();
  }

  /// Initializes the necessary state changes needed after performing a search operation
  _onSearchChanged() {
    List<QueryDocumentSnapshot> filteredResultsList = [];
    _resultsList.forEach((element) {
      User currentUser = User.fromJson(
          element.data() as Map<String, dynamic>, element.reference);
      String formattedSearchText = _searchController.text.toLowerCase();

      if (currentUser.name.toLowerCase().contains(formattedSearchText) ||
          currentUser.email.toLowerCase().contains(formattedSearchText)) {
        filteredResultsList.add(element);
      }
    });
    setState(() {
      _searchResultsList = filteredResultsList;
    });
  }

  // Load user profile
  _showUserProfile(BuildContext context, snapshot) {
    DocumentReference docRef = snapshot.reference;

    User userObj = User(
        name: snapshot.data()['name'],
        email: snapshot.data()['email'],
        points: snapshot.data()['points'],
        isAdmin: snapshot.data()['isAdmin']);
    bool isAdmin_ = userObj.isAdmin;
    showDialog(
        context: context,
        builder: (context) {
          return StatefulBuilder(builder: (context, setState) {
            return Center(
              child: SizedBox(
                width: 300,
                height: 400,
                child: Card(
                  elevation: 3,
                  shape: RoundedRectangleBorder(
                    side: const BorderSide(color: Colors.white70, width: 1),
                    borderRadius: BorderRadius.circular(10),
                  ),
                  child: Column(
                    children: [
                      const SizedBox(
                        height: 25,
                      ),
                      const CircleAvatar(
                        backgroundImage: AssetImage('assets/img/user.png'),
                        minRadius: 60,
                        maxRadius: 100,
                      ),
                      const SizedBox(
                        height: 15,
                      ),
                      Text(userObj.name,
                          style: Theme.of(context).textTheme.headline5),
                      Text(userObj.email,
                          style: Theme.of(context).textTheme.caption),
                      const SizedBox(
                        height: 25,
                      ),
                      FlutterSwitch(
                        width: 110.0,
                        height: 45.0,
                        toggleSize: 50.0,
                        value: isAdmin_,
                        borderRadius: 25.0,
                        padding: 8.0,
                        showOnOff: true,
                        activeText: 'Admin',
                        inactiveText: 'User  ',
                        onToggle: (val) {
                          setState(() {
                            isAdmin = val;
                            isAdmin_ = val;
                          });
                          _updateIsAdmin(context, docRef, val);
                        },
                      ),
                    ],
                  ),
                ),
              ),
            );
          });
        });
  }

  _updateIsAdmin(BuildContext context, DocumentReference docRef, bool isAdmin) {
    userController.updateIsAdmin(docRef, isAdmin);
    Provider.of<UserProvider>(context, listen: false).update(isAdmin);
  }

  // Load all the users to the build body as a widget
  Widget buildBody(BuildContext context) {
    return StreamBuilder<QuerySnapshot>(
      stream: userController.getAllUsers(),
      builder: (context, snapshot) {
        if (snapshot.hasError) {
          return Text('Error ${snapshot.error}');
        }
        if (snapshot.hasData) {
          // ignore: avoid_print
          print("Document -> ${snapshot.data!.docs.length}");
          _resultsList = snapshot.data!.docs;
          //Renders the user list based on the search criteria
          if (_searchController.text.isEmpty) {
            return buildList(context, _resultsList);
          } else {
            return buildList(context, _searchResultsList);
          }
        }
        return buildList(context, []);
      },
    );
  }

//Load list and convert to a list view
  Widget buildList(BuildContext context, List<DocumentSnapshot> snapshot) {
    int _currentUserNumber = 0;

    return ListView(
        children: snapshot
            .map((data) => listItemBuild(context, data, ++_currentUserNumber))
            .toList());
  }

//Load Single User Object a single item
  Widget listItemBuild(
      BuildContext context, DocumentSnapshot data, int userNumber) {
    final userObj =
        User.fromJson(data.data() as Map<String, dynamic>, data.reference);
    final String formattedUserNumberText = " " + userNumber.toString() + " ";
    isAdmin = userObj.isAdmin;
```

```
return Padding(
  key: ValueKey(userObj.name),
  padding: const EdgeInsets.symmetric(vertical: 19, horizontal: 1),
  child: Dismissible(
    key: Key(userObj.email.toString() +
      Random().nextInt(10000).toString()), //--> userObj.email
    background: Container(
     color: const Color.fromARGB(255, 32,167,219),
     child: const Padding(
      padding: EdgeInsets.all(15),
      child: Icon(Icons.delete, color: Colors.red, size: 50),
     ),
    ),
    child: Container(
     decoration: BoxDecoration(
      border: Border.all(color: Colors.blue),
      borderRadius: BorderRadius.circular(4),
     ),
     child: SingleChildScrollView(
      child: ListTile(
       title: InkWell(
         child: Column(children: <Widget>[
          Row(children: <Widget>[
           Container(
            child: Text(formattedUserNumberText,
              style: const TextStyle(color: Colors.white)),
            decoration: const BoxDecoration(
              borderRadius:
                BorderRadius.all(Radius.circular(5)),
              color: Color.fromARGB(255, 32,167,219)),
            padding: const EdgeInsets.all(3.0),
            margin: const EdgeInsets.only(right: 5.0),
           ),
           Flexible(
             child: SubstringHighlight(
            text: userObj.name,
            term: _searchController.text,
            textStyle: const TextStyle(
              // non-highlight style
              color: Colors.black,
              fontSize: 16),
            textStyleHighlight: const TextStyle(
             // highlight style
    color: Colors.black,
    backgroundColor: Colors.yellow,
  ),
 )),
 Container(
    child: isAdmin
        ? Container(
            child: const Text('Admin',
                style: TextStyle(
                    color: Color.fromARGB(
                        255, 2, 79, 167))),
            decoration: const BoxDecoration(
                border: Border(
                  top: BorderSide(
                      width: 1.0,
                      color: Color.fromARGB(
                          255, 2, 79, 167)),
                  left: BorderSide(
                      width: 1.0,
                      color: Color.fromARGB(
                          255, 2, 79, 167)),
                  right: BorderSide(
                      width: 1.0,
                      color: Color.fromARGB(
                          255, 2, 79, 167)),
                  bottom: BorderSide(
                      width: 1.0,
                      color: Color.fromARGB(
                          255, 2, 79, 167)),
                ),
                borderRadius: BorderRadius.all(
                    Radius.circular(5)),
                color:
                    Color.fromARGB(255, 255, 255, 255)),
            padding: const EdgeInsets.all(3.0),
            margin: const EdgeInsets.only(left: 5.0),
          )
        : null),
]),
const Divider(),
Row(children: <Widget>[
  Container(
    child: const Icon(Icons.email, color: Colors.orange),
                        margin: const EdgeInsets.only(right: 3.0),
                      ),
                      Flexible(
                          child: SubstringHighlight(
                        text: userObj.email,
                        term: _searchController.text,
                        textStyle: const TextStyle(
                            // non-highlight style
                            color: Colors.black,
                            fontSize: 16),
                        textStyleHighlight: const TextStyle(
                          // highlight style
                          color: Colors.black,
                          backgroundColor: Colors.yellow,
                        ),
                      )),
                    ]),
                  ]),
                  onTap: () => {_showUserProfile(context, data)},
                ),
                trailing: IconButton(
                    icon: const Icon(Icons.delete, color: Colors.red),
                    onPressed: () {
                      _showDeleteAlertDialogBox(context, userObj);
                    }),
              ),
            ),
          ),
          onDismissed: (DismissDirection direction) {
            var value = _isFirstTimeQuickDelete();
            value.then((result) =>
                {_performQuickDeletion(direction, result, userObj)});
          }));
  }

//Build Widget
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: Row(
          mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Image.asset(
            'assets/img/logo.jpg',
            fit: BoxFit.cover,
            height: 60.0,
          ),
          Container(
            padding: const EdgeInsets.all(8.0),
            child: const Text(
              'Wordly',
              style: TextStyle(fontFamily: 'Righteous', fontSize: 20.0),
            ),
          )
        ],
      ),
      backgroundColor: const Color.fromARGB(255, 28,150,197),
    ),
    drawer: const MainDrawer(),
    body: Container(
      padding: const EdgeInsets.all(19),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.start,
        children: <Widget>[
          Container(),
          const SizedBox(
            height: 20,
          ),
          Wrap(
              crossAxisAlignment: WrapCrossAlignment.center,
              children: const [
                Icon(Icons.supervised_user_circle,
                    color: Color.fromARGB(255, 28,150,197), size: 30),
                Text(
                  " USERS LIST",
                  style: TextStyle(fontSize: 18, fontWeight: FontWeight.w800),
                )
              ]),
          const SizedBox(
            height: 20,
          ),
          TextField(
            controller: _searchController,
              decoration: InputDecoration(
                prefixIcon: const IconButton(
                  color: Colors.black,
                  icon: Icon(Icons.search),
                  iconSize: 20.0,
                  onPressed: null,
                ),
                suffixIcon: IconButton(
                    color: Colors.black,
                    icon: const Icon(Icons.clear),
                    iconSize: 20.0,
                    onPressed: () => _searchController.clear()),
                contentPadding: const EdgeInsets.only(left: 25.0),
                hintText: 'Search',
                border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(4.0))),
            ),
            Flexible(child: buildBody(context))
          ],
        ),
      ),
    );
  }
}
```

4.10 Screens/leaderboard.dart

```
// ignore_for_file: avoid_print

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:wordly/widgets/main_drawer.dart';

class LeaderBoard extends StatefulWidget {
  const LeaderBoard({Key? key}) : super(key: key);

  @override
  _LeaderBoardState createState() => _LeaderBoardState();
}

class _LeaderBoardState extends State<LeaderBoard> {
  int i = 0;
  // ignore: non_constant_identifier_names
```

```
Color my = Colors.brown, CheckMyColor = Colors.white;

@override
Widget build(BuildContext context) {
  var r = const TextStyle(color: Colors.purpleAccent, fontSize: 34);
  return Stack(
    children: <Widget>[
      Scaffold(
          resizeToAvoidBottomInset: false,
          appBar: AppBar(
            title: Row(
              mainAxisAlignment: MainAxisAlignment.start,
              children: [
                Image.asset(
                  'assets/img/logo.jpg',
                  fit: BoxFit.cover,
                  height: 60.0,
                ),
                Container(
                  padding: const EdgeInsets.all(8.0),
                  child: const Text(
                    'Wordly',
                    style: TextStyle(fontFamily: 'Righteous', fontSize: 20.0),
                  ),
                )
              ],
            ),
            backgroundColor: const Color.fromARGB(255, 28, 150, 197),
          ),
          drawer: const MainDrawer(),

          body: Container(

            margin: const EdgeInsets.only(top: 65.0),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.center,

              children: <Widget>[

                Wrap(

                    crossAxisAlignment: WrapCrossAlignment.center,

                    children: const [
      Icon(Icons.leaderboard_rounded,
          color: Color.fromARGB(255, 28, 150, 197), size: 30),
      Text(
        " LEADERBOARD",
        style: TextStyle(
            fontSize: 18, fontWeight: FontWeight.w800),
      )
    ]),
const SizedBox(
  height: 20,
),
Flexible(
    child: StreamBuilder<QuerySnapshot>(
        stream: FirebaseFirestore.instance
            .collection('users')
            .orderBy('points', descending: true)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            i = 0;
            return ListView.builder(
                itemCount: snapshot.data!.docs.length,
                itemBuilder: (context, index) {
                  print(index);
                  if (index >= 1) {
                    print('Greater than 1');
                    if (snapshot.data!.docs[index]
                            .get('points') ==
      snapshot.data!.docs[index - 1]
          .get('points')) {
    print('Same');
  } else {
    i++;
  }
}

return Padding(
  padding: const EdgeInsets.symmetric(
      horizontal: 5.0, vertical: 5.0),
  child: InkWell(
    child: Container(
      decoration: BoxDecoration(
          border: Border.all(
              color: i == 0
                  ? Colors.amber
                  : i == 1
                      ? Colors.grey
                      : i == 2
                          ? Colors.brown
                          : Colors.white,
              width: 3.0,
              style: BorderStyle.solid),
          borderRadius:
              BorderRadius.circular(5.0)),
      width:
          MediaQuery.of(context).size.width,
child: Column(
  children: <Widget>[
    Row(
      children: <Widget>[
        Padding(
          padding:
              const EdgeInsets.only(
                  top: 7.5,
                  bottom: 7.5,
                  left: 15.0),
          child: Row(
            children: <Widget>[
              CircleAvatar(
                  child: Container(
                      decoration: const BoxDecoration(
                          shape: BoxShape
                              .circle,
                          image: DecorationImage(
                              image: AssetImage(
                                  'assets/img/user.png'),
                              fit: BoxFit
                                  .fill)))),
            ],
          ),
        ),
        Padding(
          padding:
              const EdgeInsets.only(
            left: 20.0,
            top: 10.0),
    child: Column(
      mainAxisAlignment:
          MainAxisAlignment
            .center,
      crossAxisAlignment:
          CrossAxisAlignment
            .start,
      children: <Widget>[
        Container(
            alignment: Alignment
              .centerLeft,
            child: Text(
              snapshot.data!
                  .docs[index]
                  .get('name'),
              style: const TextStyle(
                  color: Color.fromARGB(255, 37, 85, 173),
                  fontWeight:
                      FontWeight
                          .w500),
              maxLines: 6,
            )),
        Text("Points: " +
            snapshot.data!
                .docs[index]
                .get('points')
```

```
                  .toString()),
             ],
            ),
           ),
           Flexible(child: Container()),
           i == 0
             ? Text("□", style: r)
             : i == 1
               ? Text(
                 "□",
                 style: r,
                )
               : i == 2
                 ? Text(
                   "□",
                   style: r,
                  )
                 : const Text(''),
          ],
         ),
        ],
       ),
      ),
     ),
    );
   });
} else {
 return const Center(
                        child: CircularProgressIndicator(),
                      );
                    }
                  }))
            ],
          ),
        )),
      ],
    );
  }
}
```

4.1 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());
         },
        ),
       ],
      ),
     );
    }
   }
```