



Sri Lanka Institute of Information Technology

Current Trends in Software Engineering

Semester 01 – Year 04 – 2020

Movie Reviewer

Group Project – Individual Submission

Authentications & Movie Management

Submitted By:

D. Manoj Kumar

IT17050272

April 30th 2020

Table of Contents

List of Figures	iii
1.0 Source Code	1
1.1 Splash Screen & Login Screen	1
1.1.1 Splash Screen – splash.dart	1
1.1.2 Login Screen – login.dart	2
1.2 User Registration Page – register.dart	7
1.3 User Profile Page – profile.dart	13
1.4 Movie Listing Page – movie.dart	21
1.5 Single Movie Detail Page – view_movieDetail.dart	27
1.6 Movie Model – Movie.dart	32
1.7 Services of Movie – movieAPI.dart	33
1.8 Services of User Authentication – auth.dart	34
2.0 Screenshots of UI	36
2.1 Splash Screen / Login Screen	36
2.1.1 User Interface of Login Screen	36
2.1.2 Error Message Displaying	37
2.1.3 Successful System Login	40
2.2 User Registration Page	42
2.2.1 User Interface for Registration Page	42
2.2.2 Error Messages Displaying	43
2.2.3 Successful User Signup	47
2.3 User Profile	48
2.3.1 User Profile Interface	48
2.3.2 Update User Profile	49
2.3.3 Delete User Profile	53
2.4 Movie Page	56
2.4.1 Multiple Movie Listing Page	56
2.4.2 Single Movie Detail Page	57
2.5 App Sign-out	58
3.0 References	60

List of Figures

Figure 2. 1: Login Screen User Interface	36
Figure 2. 2: Error Message type 1: Login Screen	37
Figure 2. 3: Error Message type 2 – Login Screen	38
Figure 2. 4: Error Message type 3 – Login Screen	39
Figure 2. 5: Successful Login	40
Figure 2. 6: Redirect to Home Page After Successful Login	41
Figure 2. 7: User Registration Form Interface	42
Figure 2. 8: Error Message Type 1 - Signup Page	43
Figure 2. 9: Error Message Type 2 – Signup Page	44
Figure 2. 10: Error Message Type 3 – Signup Page	45
Figure 2. 11: Error Message Type 4 – Signup Page	46
Figure 2. 12: Successful User Registration	47
Figure 2. 13: User Profile Interface	48
Figure 2. 14: Click "EDIT PROFILE" button	49
Figure 2. 15: Edit ‘first name’ & ‘surname’	50
Figure 2. 16: Click on “UPDATE” Text	51
Figure 2. 17: Redirect to the User Profile with Toast Message & Updated Details	52
Figure 2. 18: Click on “DELETE PROFILE” button	53
Figure 2. 19: Click on “DELETE” Text in Alert Message to Confirm Delete	54
Figure 2. 20: Redirect to Login Page with Toast Message after Successful Delete	55
Figure 2. 21: Movie Listing Interface	56
Figure 2. 22: Single Movie Detail Page Interface	57
Figure 2. 23: Click on ‘Sign-out’ Icon	58
Figure 2. 24: Click on ‘YES’ Text in Alert Box	59

1.0 Source Code

All the codes mentioned below are not screenshots

1.1 Splash Screen & Login Screen

1.1.1 Splash Screen – splash.dart

```
2  import 'package:flutter/material.dart';
3  import 'package:firebase_auth/firebase_auth.dart';
4  import 'package:cloud_firestore/cloud_firestore.dart';
5  import 'package:movie_corns/constants/constants.dart';
6  import 'home.dart';
7
8  /*
9   * IT17050272 - D. Manoj Kumar
10  *
11  * splash.dart file is the first page of this app. Once the user opens the app,
12  * the user interface which is created by using the source code in this dart file will
13  * be displayed.
14  * According to the below mentioed code, we included login page as our spalsh screen
15  */
16
17  class SplashPage extends StatefulWidget {
18    SplashPage({ Key key }) : super(key: key);
19
20    @override
21    _SplashPageState createState() => _SplashPageState();
22  }
23
24  class _SplashPageState extends State<SplashPage> {
25    @override
26    initState() {
27      FirebaseAuth.instance
28        .currentUser()
29        .then((FirebaseUser currentUser) => {
30          //if there is no current user logged into the app, then the system will redirect to login screen
31          if (currentUser == null)
32            Navigator.pushReplacementNamed(context, "/login")
33          else
34            {
35              //If there is an user already logged into the system, then redirect to home page
36              Firestore.instance
37                .collection("users")
```

```

38         .document(currentUser.toString())
39         .get()
40         .then((DocumentSnapshot result) =>
41             Navigator.pushReplacement(
42                 context,
43                 MaterialPageRoute(
44                     builder: (context) => HomePage(
45                         title: TitleConstants.ALL_MOVIES,
46                     )),
47             ).catchError((err) => print(err))
48     }
49 })
50 .catchError((err) => print(err));
51 super.initState();
52 }
53
54 @override
55 Widget build(BuildContext context) {
56     return Scaffold(
57         body: Center(
58             child: Container(child: CircularProgressIndicator()),
59         ),
60     );
61 }
62 }
63

```

1.1.2 Login Screen – login.dart

```

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:movie_corns/api/services/auth.dart';
import 'package:movie_corns/constants/constants.dart';
import 'package:movie_corns/pages/home.dart';
import 'package:progress_dialog/progress_dialog.dart';
import 'package:fluttertoast/fluttertoast.dart';

/*
 * IT17050272 - D. Manoj Kumar
 *
 * This source code in login.dart file is used to create the user interface
 * as well as the backend tasks of the Login page of the app. In order to

```

- * signin to the app user needs to provide the email address as the username
- * & the password which he/she mentioed while registering the app
- * (mentioned in register.dart). Once the user enters the username & password the
- * system will compare those credentials with database. If creadentials matched,
- * user will be redirected to the next page, movie page with relavant user id.
- * If creadentials not matched then, the system will display relavant error
- * message. Also, user needs to provide both username & password before clicks on
- * 'LOGIN' button, if not the system will provide relavant error messages telling that
- * fields are empty. Apart from these, the user can redirected to register.dart file
- * if user doesn't signup before.
- * These are the tasks which are fulfilled by this dart file.
- */

```

class LoginPage extends StatefulWidget {
  LoginPage({Key key}) : super(key: key);

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

class _LoginPageState extends State<LoginPage> {
  final GlobalKey<FormState> _loginFormKey = GlobalKey<FormState>();
  TextEditingController emailInputController;
  TextEditingController pwdInputController;
  ProgressDialog pr;
  Auth auth;

  Function toast(
    String msg, Toast toast, ToastGravity toastGravity, Color colors) {
    Fluttertoast.showToast(
      msg: msg,
      toastLength: toast,
      gravity: toastGravity,
      backgroundColor: colors,
      textColor: Colors.white,
      fontSize: 16.0);
  }

  @override
  initState() {
    emailInputController = new TextEditingController();
    pwdInputController = new TextEditingController();
    super.initState();
  }

```

```

//Start email & password validation
//Start email validation
//check whether all the necessary points in email are included in user entered email/username
String emailValidator(String value) {
    Pattern pattern =
        r'^([^\>()[]\.,;:\s@\'"]+(\.[^\>()[]\.,;:\s@\'"]+)*)(\."|\.+)"')@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\]|((\[[a-zA-Z\0-9]+\.]|[a-zA-Z]{2,})))$';
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value)) {
        return ValidatorConstants.INVALID_EMAIL_FORMAT;
    } else {
        return null;
    }
}
//End email validation

//Start Password validation
String pwdValidator(String value) {
    //check whether the number of characters in the password less than 8
    if (value.length < 8) {
        return ValidatorConstants.WEAK_PASSWORD;
    } else {
        return null;
    }
}
//End password validation
//End email & password validation

@override
Widget build(BuildContext context) {
    pr = new ProgressDialog(context, type: ProgressDialogType.Normal);
    pr.style(
        message: ProgressDialogMessageConstants.LOGGING_IN,
        borderRadius: 10.0,
        backgroundColor: Colors.white,
        progressWidget: CircularProgressIndicator(),
        elevation: 10.0,
        insetAnimCurve: Curves.easeInOut,
        progress: 0.0,
        maxProgress: 100.0,
        progressTextStyle: TextStyle(
            color: Colors.black, fontSize: 13.0, fontWeight: FontWeight.w400),
        messageTextStyle: TextStyle(
            color: Colors.black, fontSize: 19.0, fontWeight: FontWeight.w600));
    return Scaffold(

```

```

appBar: AppBar(
  title: Text(
    TitleConstants.LOGIN,
    textAlign: TextAlign.justify,
  ),
),
body: Container(
  padding: const EdgeInsets.all(20.0),
  child: SingleChildScrollView(
    child: Form(
      key: _loginFormKey,
      child: Column(
        children: <Widget>[
          TextFormField(
            decoration: InputDecoration(
              labelText: LabelConstants.LABEL_EMAIL,
              hintText: HintTextConstants.HINT_EMAIL),
            controller: emailInputController,
            keyboardType: TextInputType.emailAddress,
            validator: emailValidator,
          ),
          TextFormField(
            decoration: InputDecoration(
              labelText: LabelConstants.LABEL_PASSWORD,
              hintText: HintTextConstants.HINT_PASSWORD),
            controller: pwdInputController,
            obscureText: true,
            validator: pwdValidator,
          ),
          SizedBox(
            height: 20,
          ),
          RaisedButton(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(19)),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                Text(ButtonConstants.LOGIN_BUTTON),
                Icon(Icons.arrow_forward),
              ],
            ),
            color: Theme.of(context).primaryColor,
            textColor: Colors.white,
            onPressed: () async {

```



```

if (_loginFormKey.currentState.validate()) {
  await pr.show();
  FirebaseAuth.instance
    .signInWithEmailAndPassword(
      email: emailInputController.text,
      password: pwdInputController.text)
    .then((currentUser) => Firestore.instance
      .collection("users")
      .document(currentUser.user.uid)
      .get()
      .then((DocumentSnapshot result) =>
        pr.hide().then((isHidden) {
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(
              builder: (context) => HomePage(
                title: TitleConstants
                  .ALL_MOVIES,
                uid: currentUser.user.uid,
              )),
          ));
        }).whenComplete(() {
          toast(
            ToastConstants.WELCOME,
            Toast.LENGTH_LONG,
            ToastGravity.BOTTOM,
            Colors.blueGrey);
          })))
    .catchError((err) => print(err)))
  .catchError((err) {
    pr.hide();
    showAuthError(err.code, context);
  });
},
),
SizedBox(
  height: 300,
),
Text(LabelConstants.LABEL_NEW_HERE),
RaisedButton(
  textColor: Theme.of(context).primaryColor,
  color: Colors.white,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(19)),
  child: Text(ButtonConstants.JOIN_BUTTON),

```

```

        onPressed: () {
          Navigator.pushNamed(context, "/register");
        },
      ),
    ],
  ),
  ));
}

void showAuthError(errorCode, BuildContext context) {
  switch (errorCode) {
    case FirebaseAuthErrorConstants.ERROR_WRONG_PASSWORD:
      toast	ToastConstants.INCORRECT_PASSWORD, Toast.LENGTH_LONG,
            ToastGravity.BOTTOM, Colors.blueGrey);
      break;

    default:
      toast	ToastConstants.UNKNOWN_AUTH_ERROR, Toast.LENGTH_LONG,
            ToastGravity.BOTTOM, Colors.blueGrey);
      break;
  }
}

```

1.2 User Registration Page – register.dart

```

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:movie_corns/constants/constants.dart';
import 'package:movie_corns/pages/home.dart';
import 'package:progress_dialog/progress_dialog.dart';

/*
 * IT17050272 - D. Manoj Kumar
 *
 * register.dart file consists with the source code for registration page user interface
 * & backend services based on user registration
 * Once the user clicks on "Join Movie Corns" button in login page he/she will redirect to
 * this page. In order to register to the system, user needs to provide firstname, surname,
 * username & password as described in below code. An email address used by the user should be
 * as the username as it will be useful for the future development of the app. The user

```

- * needs to provide a password at least 8 character long in order to prevent from unauthorized
- * logins. Also the password has to mention twice to ensure the password entered.
- * After entering all the required information, user can clicks on "Join Movie Corns" button.
- * Soonafter user clicks on it, the user entered data will stored in "users" database in firebase
- * under a newly auto-created user ID/document ID. This user ID will be used under every tasks.
- * Once the user successfully registered to the system, the user will be redirected to the home
- * page (Movie page) of the app
- * As mentioed in below code, the registration page is consisting with validators. It is required
- * to fill all the details mentioned in the form brfore clicks on "Join Movie Corns" button & if
- * user fails to fil one field the system will display an error message. Apart from that, there are
- * validators to check the length of the password as well as the type of the username is "Email" too.
- */

```
class RegisterPage extends StatefulWidget {
  RegisterPage({Key key}) : super(key: key);

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

class _RegisterPageState extends State<RegisterPage> {
  final GlobalKey<FormState> _registerFormKey = GlobalKey<FormState>();
  TextEditingController firstNameInputController;
  TextEditingController lastNameInputController;
  TextEditingController emailInputController;
  TextEditingController pwdInputController;
  TextEditingController confirmPwdInputController;
  ProgressDialog pr1;

  @override
  initState() {
    firstNameInputController = new TextEditingController();
    lastNameInputController = new TextEditingController();
    emailInputController = new TextEditingController();
    pwdInputController = new TextEditingController();
    confirmPwdInputController = new TextEditingController();
    super.initState();
  }

  String emailValidator(String value) {
    Pattern pattern =
      r'^((([^\<>@[\]\\.,:;\\s@\\"]+)(\\.[^\<>@[\]\\.,:;\\s@\\"]+)*)(\\(\\.[^\<>@[\]\\.,:;\\s@\\"]+\\))?)@(([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3})|([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))$';
    RegExp regex = new RegExp(pattern);
    if (!regex.hasMatch(value)) {
```

```

    return ValidatorConstants.INVALID_EMAIL_FORMAT;
  } else {
    return null;
  }
}

String pwdValidator(String value) {
  if (value.length < 8) {
    return ValidatorConstants.WEAK_PASSWORD;
  } else {
    return null;
  }
}

@override
Widget build(BuildContext context) {
  pr1 = new ProgressDialog(context, type: ProgressDialogType.Normal);
  pr1.style(
    message: ProgressDialogMessageConstants.WELCOME_ABOARD,
    borderRadius: 10.0,
    backgroundColor: Colors.white,
    progressWidget: CircularProgressIndicator(),
    elevation: 10.0,
    insetAnimCurve: Curves.easeInOut,
    progress: 0.0,
    maxProgress: 100.0,
    progressTextStyle: TextStyle(
      color: Colors.black, fontSize: 13.0, fontWeight: FontWeight.w400),
    messageTextStyle: TextStyle(
      color: Colors.black, fontSize: 19.0, fontWeight: FontWeight.w600));
  return Scaffold(
    appBar: AppBar(
      title: Text(TitleConstants.REGISTER),
    ),
    body: Container(
      padding: const EdgeInsets.all(20.0),
      child: SingleChildScrollView(
        child: Form(
          key: _registerFormKey,
          child: Column(
            children: <Widget>[
              TextFormField(
                decoration: InputDecoration(
                  labelText: LabelConstants.LABEL_FIRST_NAME,
                  hintText: HintTextConstants.HINT_FIRST_NAME),

```

```

        controller: firstNameInputController,
        validator: (value) {
            if (value.length < 3) {
                return ValidatorConstants.INVALID_FIRST_NAME;
            }
        },
    ),
    TextFormField(
        decoration: InputDecoration(
            labelText: LabelConstants.LABEL_LAST_NAME,
            hintText: HintTextConstants.HINT_LAST_NAME),
        controller: lastNameInputController,
        validator: (value) {
            if (value.length < 3) {
                return ValidatorConstants.INVALID_LAST_NAME;
            }
        },
    ),
    TextFormField(
        decoration: InputDecoration(
            labelText: LabelConstants.LABEL_EMAIL,
            hintText: HintTextConstants.HINT_EMAIL),
        controller: emailInputController,
        keyboardType: TextInputType.emailAddress,
        validator: emailValidator,
    ),
    TextFormField(
        decoration: InputDecoration(
            labelText: LabelConstants.LABEL_PASSWORD,
            hintText: HintTextConstants.HINT_PASSWORD),
        controller: pwdInputController,
        obscureText: true,
        validator: pwdValidator,
    ),
    TextFormField(
        decoration: InputDecoration(
            labelText: LabelConstants.LABEL_CONFIRM_PASSWORD,
            hintText: HintTextConstants.HINT_PASSWORD),
        controller: confirmPwdInputController,
        obscureText: true,
        validator: pwdValidator,
    ),
    SizedBox(
        height: 20,
    ),
    RaisedButton(

```

```

shape: RoundedRectangleBorder(
  borderRadius: BorderRadius.circular(19)),
child: Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    Text(ButtonConstants.JOIN_BUTTON),
    Icon(Icons.arrow_forward),
  ],
),
color: Theme.of(context).primaryColor,
textColor: Colors.white,
onPressed: () async {
  if (_registerFormKey.currentState.validate()) {
    if (pwdInputController.text ==
      confirmPwdInputController.text) {
      await pr1.show();
      FirebaseAuth.instance
        .createUserWithEmailAndPassword(
          email: emailInputController.text,
          password: pwdInputController.text)
        .then((currentUser) => Firestore.instance
          .collection("users")
          .document(currentUser.user.uid)
          .setData({
            "uid": currentUser.user.uid,
            "fname": firstNameInputController.text,
            "surname": lastNameInputController.text,
            "email": emailInputController.text,
          })
        ).then((result) => {
          pr1.hide().then((isHidden) {
            Navigator.pushAndRemoveUntil(
              context,
              MaterialPageRoute(
                builder: (context) =>
                  HomePage(
                    title: TitleConstants
                      .ALL_MOVIES,
                    uid: currentUser
                      .user.uid,
                  ),
                ),
            (_) => false);
          }),
          firstNameInputController.clear(),
          lastNameInputController.clear(),

```

```

        emailInputController.clear(),
        pwdInputController.clear(),
        confirmPwdInputController.clear()
    })
    .catchError((err) => print(err)))
    .catchError((err) => print(err));
} else {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text(TitleConstants.ALERT_ERROR),
        content: Text(ValidatorConstants
          .PASSWORDS_DO_NOT_MATCH),
        actions: <Widget>[
          FlatButton(
            child: Text(ButtonConstants.OPTION_CLOSE),
            onPressed: () {
              Navigator.of(context).pop();
            },
          )
        ],
      );
    });
}
},
),
 SizedBox(
  height: 130,
),
Text(LabelConstants.LABEL_ALREADY_HAVE_ACCOUNT),
RaisedButton(
  textColor: Theme.of(context).primaryColor,
  color: Colors.white,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(19)),
  child: Text(ButtonConstants.LOGIN),
  onPressed: () {
    Navigator.pop(context);
  },
)
],
),
));

```

```
}  
}
```

1.3 User Profile Page – profile.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter/cupertino.dart';  
import 'package:movie_corns/api/services/auth.dart';  
import 'package:movie_corns/constants/constants.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
  
/*  
 * IT17050272 - D. Manoj Kumar  
 *  
 * profile.dart file consists with the source code which used to create User Profile  
 * interfaces & backend services.  
 * Once an user login to the app, the system will fetch the details of the logged user  
 * using the uid/documentID which had created when user registered to the system for first  
 * time. After user login to the app successfully he/she can redirect their user profile  
 * clicking on "My Profile" tab in bottom navigation  
 * User profile UI consists with logged user's firstname, surname & username & also there  
 * are 2 buttons namely "Update Profile" & "Delete Profile" to update or delete the profile.  
 * Once the user successfully login to the system, the system will fetch all the relevant  
 * data from the database according to the uid & place them in the right place in the user  
 * profile as per the below code. Hence, the image of the user is not a compulsory data,  
 * here used a default image from the internet to represent user.  
 * Once the user clicks on "Update Profile" button, the system will display a dialog  
 * box which contain with first name & surname of logged user. We are not allowed to update  
 * username(email) & the password to this point (Hope to develop the task in future). If user  
 * wished to update his/her firstname/surname or both of them, they can give new names & clicks  
 * on the update button. Soon after button clicked the system will redirect to the same user profile  
 * page with updated first name & surname.  
 * Once the user clicks on "Delete Profile" button, the system will display an alert message to confirm  
 * the delete task. If user gives the permission by clicking on the "Yes" button,  
 * then all the user tasks including all the provided reviews under the deleted user ID &  
 * the user profile will be deleted from whole system  
 *  
 * These are the overall tasks complete through this dart file
```



```

*/

class ProfilePage extends StatefulWidget {
  ProfilePage({Key key, this.title, this.uid}) : super(key: key);
  final String title;
  final String uid;
  final Auth auth = new Auth();
  @override
  _ProfilePageState createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage>
  with SingleTickerProviderStateMixin {
  final FocusNode myFocusNode = FocusNode();
  String userId = "";

  //create a function for the toast
  Function toast(
    String msg, Toast toast, ToastGravity toastGravity, Color colors) {
    Fluttertoast.showToast(
      msg: msg,
      toastLength: toast,
      gravity: toastGravity,
      timeInSecForIosWeb: 1,
      backgroundColor: colors,
      textColor: Colors.white,
      fontSize: 16.0);
  }

  @override
  void initState() {
    super.initState();
    widget.auth.getCurrentUser().then((user) {
      setState() {
        if (user != null) {
          userId = user?.uid;
        }
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(

```

```

        automaticallyImplyLeading: false,
        title: Text(TitleConstants.PROFILE),
        actions: <Widget>[
          IconButton(
            icon: Icon(Icons.exit_to_app),
            onPressed: () {
              showDialog(
                context: context,
                builder: (BuildContext context) {
                  return AlertDialog(
                    title: Text(TitleConstants.ALERT_SIGN_OUT),
                    content:
                      Text(PromptConstants.QUESTION_CONFIRM_SIGN_OUT),
                    actions: [
                      FlatButton(
                        child: Text(ButtonConstants.OPTION_CANCEL),
                        onPressed: () {
                          Navigator.pop(context);
                        },
                      ),
                      FlatButton(
                        child: Text(ButtonConstants.OPTION_YES),
                        onPressed: () {
                          widget.auth.signOut();
                          Navigator.pushNamedAndRemoveUntil(
                            context, "/login", (_) => false);
                        },
                      ),
                    ],
                  );
                },
              );
            },
          ),
        ],
        body: _prepareDetailAndBody(context, userId));
  }

Widget _prepareDetailAndBody(BuildContext context, String userId) {
  return StreamBuilder<DocumentSnapshot>(
    stream:
      Firestore.instance.collection('users').document(userId).snapshots(),
    builder: (context, AsyncSnapshot snapshot) {
      if (!snapshot.hasData) return CircularProgressIndicator();
    },
  );
}

```

```

        return _movieDetailBody(context, snapshot.data);
    },
);
}

Widget _movieDetailBody(BuildContext context, DocumentSnapshot snapshot) {
    final rating = Container(
        padding: const EdgeInsets.all(7.0),
        decoration: new BoxDecoration(
            border: new Border.all(color: Colors.white),
            borderRadius: BorderRadius.circular(5.0)),
        child: Text(
            snapshot["email"],
            style: TextStyle(color: Colors.white),
        ),
    );

    final header = Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
            SizedBox(height: 10.0),
            Text(
                snapshot["fname"] + " " + snapshot["surname"],
                style: TextStyle(color: Colors.white, fontSize: 30.0),
            ),
            SizedBox(height: 20.0),
            SizedBox(height: 30.0),
            Row(
                mainAxisAlignment: MainAxisAlignment.start,
                children: <Widget>[Expanded(child: rating)],
            ),
            Row(
                mainAxisAlignment: MainAxisAlignment.start,
            )
        ],
    );

    final headerContent = Stack(
        children: <Widget>[
            Container(
                padding: EdgeInsets.only(left: 10.0),
                height: MediaQuery.of(context).size.height * 0.5,
                decoration: new BoxDecoration(
                    image: new DecorationImage(
                        image: new NetworkImage(NetworkImagesPath.PROFILE_AVATAR),

```

```

        fit: BoxFit.cover),
    )),
    Container(
      height: MediaQuery.of(context).size.height * 0.5,
      padding: EdgeInsets.all(40.0),
      width: MediaQuery.of(context).size.width,
      decoration: BoxDecoration(color: Color.fromRGBO(54, 60, 100, .9)),
      child: Center(
        child: header,
      ),
    ),
  ],
);

final editProfileButton = Container(
  padding: EdgeInsets.symmetric(vertical: 16.0),
  width: MediaQuery.of(context).size.width,
  child: RaisedButton.icon(
    icon: Icon(Icons.edit, color: Colors.white),
    shape:
      RoundedRectangleBorder(borderRadius: BorderRadius.circular(19)),
    onPressed: () => {_displayDialog(context, snapshot)},
    color: Theme.of(context).primaryColor,
    label: Text(ButtonConstants.EDIT_PROFILE,
      style: TextStyle(color: Colors.white)),
  ));

final deleteAccountButton = Container(
  padding: EdgeInsets.symmetric(vertical: 16.0),
  width: MediaQuery.of(context).size.width,
  child: RaisedButton.icon(
    icon: Icon(Icons.delete, color: Colors.white),
    shape:
      RoundedRectangleBorder(borderRadius: BorderRadius.circular(19)),
    onPressed: () => {
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: Text(TitleConstants.ALERT_WARNING),
            content:
              Text(PromptConstants.QUESTION_CONFIRM_ACCOUNT_DELETE),
            actions: <Widget>[
              FlatButton(
                child: Text(ButtonConstants.OPTION_CANCEL),

```

```

onPressed: () {
  Navigator.of(context).pop();
},
),
FlatButton(
  textColor: Colors.red,
  onPressed: () async {
    final Firestore firestore = Firestore.instance;
    final FirebaseAuth firebaseAuth =
      FirebaseAuth.instance;

    FirebaseUser user1 = await firebaseAuth.currentUser();

    firestore
      .collection('reviews')
      .getDocuments()
      .then((snap) {
        for (DocumentSnapshot ds in snap.documents) {
          if (ds.data["uid"] == userId) {
            ds.reference.delete();
          }
        }
      });

    firestore
      .collection('users')
      .getDocuments()
      .then((users) {
        for (DocumentSnapshot us in users.documents) {
          if (us.data["uid"] == userId) {
            us.reference.delete();
          }
        }
      });

    user1.delete();
    widget.auth.signOut();
    Navigator.pushNamedAndRemoveUntil(
      context, "/login", (_) => false);

    toast(
      ToastConstants.PROFILE_DELETED_SUCCESS,
      Toast.LENGTH_LONG,
      ToastGravity.BOTTOM,
      Colors.blueGrey);

```

```

        },
        child: Text(ButtonConstants.OPTION_DELETE),
      ),
    ],
  );
})

},
color: Colors.red,
label: Text(ButtonConstants.DELETE_PROFILE,
  style: TextStyle(color: Colors.white)),
));

final bottomContent = Container(
  width: MediaQuery.of(context).size.width,
  padding: EdgeInsets.all(10.0),
  child: Center(
    child: Column(
      children: <Widget>[editProfileButton, deleteAccountButton],
    ),
  ),
);

return new Scaffold(
  body: Column(
    children: <Widget>[headerContent, bottomContent],
  ),
);
}

String getMovieUrl(String snapUrl) {
  if (snapUrl.isEmpty) return NetworkImagesPath.MOVIE_AVATAR;

  return snapUrl;
}

@override
void dispose() {
  myFocusNode.dispose();
  super.dispose();
}

_displayDialog(BuildContext context, DocumentSnapshot snapshot) async {
  TextEditingController fnameController = new TextEditingController();
  TextEditingController surnameController = new TextEditingController();
  return showDialog(

```

```

context: context,
builder: (BuildContext context) {
  return AlertDialog(
    title: Text(TitleConstants.ALERT_EDIT_PROFILE),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(10.0)), //this right here
    content: Container(
      height: 100,
      child: Container(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            TextField(
              controller: fnameController,
              decoration: InputDecoration(
                border: InputBorder.none,
                hintText: snapshot["fname"]),
            ),
            TextField(
              controller: surnameController,
              decoration: InputDecoration(
                border: InputBorder.none,
                hintText: snapshot["surname"]),
            ),
          ],
        ),
      ),
    ),
    actions: <Widget>[
      FlatButton(
        child: Text(ButtonConstants.OPTION_CANCEL),
        onPressed: () {
          Navigator.of(context).pop();
        },
      ),
      FlatButton(
        onPressed: () {
          //return false;

          String fname = "";
          String surname = "";

          if (fnameController.text.isEmpty)
            fname = snapshot["fname"];

```

```

    else
      fname = fnameController.text;

    if (surnameController.text.isEmpty)
      surname = snapshot["surname"];
    else
      surname = surnameController.text;

    print(widget.uid);

    Firestore.instance
      .collection('users')
      .document(userId)
      .setData({
        "fname": fname,
        "surname": surname,
        "email": snapshot["email"],
        "uid": userId
      });
    Navigator.of(context).pop();
    toast(ToastConstants.PROFILE_UPDATE_SUCCESS,
      Toast.LENGTH_LONG, ToastGravity.BOTTOM, Colors.blueGrey);
    return true;
  },
  child: Text(ButtonConstants.OPTION_UPDATE),
),
],
);
});
}
}

```

1.4 Movie Listing Page – movie.dart

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:movie_corns/api/services/auth.dart';
import 'package:movie_corns/constants/constants.dart';
import 'package:movie_corns/pages/view_movieDetail.dart';

/*
 * IT17050272 - D. Manoj Kumar
 *
 * The main task of this movie.dart file is, providing the source code

```


- * to create the user interface & backend tasks to view a collection of movies in a list
- * By calling firestore get method, the app will fetch all the movies which are already
- * stored in firebase database & view them to user in seperate card views
- * while ordering them according to their release years.
- * Each movie card is containg with the image of the movie, name of the movie,
- * the release year as well as the average of rating. Once the user clicks on
- * a movie card he/she will be abled to redirected to the Movie Details page
- * for the selected movie according to the source code mention below.
- */

```
class MoviePage extends StatefulWidget {
  MoviePage({Key key, this.title, this.uid, this.movieId}) : super(key: key);
  final String title;
  final String uid;
  final String movieId;
  final Auth auth = new Auth();

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

```
class _MoviePageState extends State<MoviePage> {
  String userId = "";

  @override
  void initState() {
    super.initState();
    widget.auth.getCurrentUser().then((user) {
      setState() {
        if (user != null) {
          userId = user?.uid;

          print(userId);
        }
      });
    });
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(TitleConstants.ALL_MOVIES),
      actions: <Widget>[
        IconButton(
```

```

        icon: Icon(Icons.exit_to_app),
        onPressed: () {
          showDialog(
            context: context,
            builder: (BuildContext context) {
              return AlertDialog(
                title: Text(TitleConstants.ALERT_SIGN_OUT),
                content:
                  Text(PromptConstants.QUESTION_CONFIRM_SIGN_OUT),
                actions: [
                  FlatButton(
                    child: Text(ButtonConstants.OPTION_CANCEL),
                    onPressed: () {
                      Navigator.pop(context);
                    },
                  ),
                  FlatButton(
                    child: Text(ButtonConstants.OPTION_YES),
                    onPressed: () {
                      widget.auth.signOut();
                      Navigator.pushNamedAndRemoveUntil(
                        context, "/login", (_) => false);
                    },
                  ),
                ],
              );
            });
        },
      ),
    ],
  ),
  body: _buildMovieBody(context));
}

```

```

Widget _buildMovieBody(BuildContext context) {
  return StreamBuilder<QuerySnapshot>(
    stream: Firestore.instance
      .collection('movies')
      .orderBy("releaseYear", descending: true)
      .snapshots(),
    builder: (context, AsyncSnapshot snapshot) {
      if (!snapshot.hasData) return CircularProgressIndicator();

      return _buildMovieList(context, snapshot.data.documents);
    },
  );
}

```

```

);
}

Widget _buildMovieList(
  BuildContext context, List<DocumentSnapshot> snapshot) {
  return ListView.builder(
    primary: false,
    physics: ScrollPhysics(),
    shrinkWrap: true,
    itemCount: snapshot.isEmpty ? 0 : snapshot.length,
    itemBuilder: (context, index) {
      return _buildMovieItem(context, snapshot[index]);
    });
}

Widget _buildMovieItem(BuildContext context, DocumentSnapshot snapshot) {
  return Padding(
    padding: EdgeInsets.only(top: 5.0, bottom: 5.0),
    child: Container(
      height: MediaQuery.of(context).size.height / 2.8,
      width: MediaQuery.of(context).size.width,
      child: new GestureDetector(
        onTap: () {
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(
              builder: (context) => ViewMovieDetailPage(
                uid: userId,
                movieId: snapshot.documentID,
              )),
        ),
      child: Card(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10.0)),
        elevation: 6.0,
        child: Column(
          children: <Widget>[
            Stack(
              children: <Widget>[
                Container(
                  height: MediaQuery.of(context).size.height / 3.5,
                  width: MediaQuery.of(context).size.width,
                  child: ClipRRect(
                    borderRadius: BorderRadius.only(
                      topLeft: Radius.circular(10),

```

```

        topRight: Radius.circular(10),
      ),
      child: Image.network(
        "${snapshot["movieImageUrl"]}",
        fit: BoxFit.cover,
      ),
    ),
  ),
  Positioned(
    top: 6.0,
    right: 6.0,
    child: Card(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(4.0)),
      child: Padding(
        padding: EdgeInsets.all(2.0),
        child: Row(
          children: <Widget>[
            Icon(
              Icons.star,
              color: Colors.orange,
              size: 10,
            ),
            Text(
              getAverageRating(snapshot["movieId"]),
              style: TextStyle(fontSize: 10),
            ),
          ],
        ),
      ),
    ),
  ),
  Positioned(
    top: 6.0,
    left: 6.0,
    child: Card(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(3.0)),
      child: Padding(
        padding: EdgeInsets.all(4.0),
        child: Text(
          "${snapshot["releaseYear"]}",
          style: TextStyle(
            fontSize: 10,
            color: Colors.green,

```

```

        fontWeight: FontWeight.bold),
      ),
    ),
  ),
],
),
PreferredSize(height: 7.0),
Padding(
  padding: EdgeInsets.only(left: 15.0),
  child: Container(
    width: MediaQuery.of(context).size.width,
    child: Text(
      "${snapshot["movieTitle"]}",
      style:
        TextStyle(fontSize: 20, fontWeight: FontWeight.w800),
      textAlign: TextAlign.left,
    ),
  ),
),
PreferredSize(height: 10.0),
],
),
),
),
),
);
}

String getAverageRating(String movieId) {
  int averagereview;
  int noOfDocs;
  Firestore.instance
    .collection('reviews')
    .where("movieId", isEqualTo: movieId)
    .getDocuments()
    .then((snap) => snap.documents.forEach((doc) => {
      averagereview = averagereview + doc.data["rating"],
      noOfDocs = noOfDocs + snap.documents.length
    }));
  return "5.0";
}
}

```

1.5 Single Movie Detail Page – view_movieDetail.dart

```
import 'dart:ui';

import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:movie_corns/api/services/auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:movie_corns/constants/constants.dart';
import 'package:movie_corns/pages/add_review.dart';
import 'package:movie_corns/pages/home.dart';

/*
 * IT17050272 - D. Manoj Kumar | IT17143950 - G.M.A.S. Bastiansz
 *
 * view_movieDetails.dart file consists with the source code which need to display a single
 * movie detail page with backend services.
 * Once the user selects a movie from the given movie-list in Movie Page, he/she will
 * redirect to this page. According to the below code, the system will get the movie ID when
 * user clicks on a movie from previous page & fetch all the data stored for that movie & retrieve
 * those data to "Movie Details" page. In this page, the system will display the movie image,
 * movie name, movie cast, & small description about the movie relevant to the selected movie ID.
 * At the bottom of the page there is a button called "Add Review" which will redirect to "Add Movie Review"
 * page with the movie ID, movie title & logged user ID
 */

class ViewMovieDetailPage extends StatelessWidget {
  final movieId;
  final uid;
  final movieTitle;

  ViewMovieDetailPage({Key key, this.movieId, this.uid, this.movieTitle})
    : super(key: key);

  final Auth auth = new Auth();
  @override
  Widget build(BuildContext context) {
    return new WillPopScope(
      onWillPop: () async => false,
      child: Scaffold(
        appBar: AppBar(
          automaticallyImplyLeading: false,
          title: Text("Movie Details"),
          leading: IconButton(
            icon: Icon(Icons.arrow_back),
            onPressed: () {
```

```

        Navigator.pushReplacement(
            context,
            MaterialPageRoute(
                builder: (context) => HomePage(
                    title: TitleConstants.ALL_MOVIES,
                    uid: uid,
                )),
        ),
        actions: <Widget>[
            IconButton(
                icon: Icon(Icons.exit_to_app),
                onPressed: () {
                    showDialog(
                        context: context,
                        builder: (BuildContext context) {
                            return AlertDialog(
                                title: Text(TitleConstants.ALERT_SIGN_OUT),
                                content:
                                    Text(PromptConstants.QUESTION_CONFIRM_SIGN_OUT),
                                actions: [
                                    FlatButton(
                                        child: Text(ButtonConstants.OPTION_CANCEL),
                                        onPressed: () {
                                            Navigator.pop(context);
                                        },
                                    ),
                                    FlatButton(
                                        child: Text(ButtonConstants.OPTION_YES),
                                        onPressed: () {
                                            auth.signOut();
                                            Navigator.pushNamedAndRemoveUntil(
                                                context, "/login", (_) => false);
                                        },
                                    ),
                                ],
                            );
                        }
                    ),
                ],
            ),
        ],
        body: _prepareDetailAndBody(context, movieId));
    }

```

```

Widget _prepareDetailAndBody(BuildContext context, String movieId) {

```

```

return StreamBuilder<DocumentSnapshot>(
  stream:
    Firestore.instance.collection('movies').document(movieId).snapshots(),
  builder: (context, AsyncSnapshot snapshot) {
    if (!snapshot.hasData) return CircularProgressIndicator();

    return _movieDetailBody(context, snapshot.data);
  },
);
}

Widget _movieDetailBody(BuildContext context, DocumentSnapshot snapshot) {
  print(uid);
  final rating = Container(
    padding: const EdgeInsets.all(7.0),
    decoration: new BoxDecoration(
      border: new Border.all(color: Colors.white),
      borderRadius: BorderRadius.circular(5.0)),
    child: new Text(
      snapshot["director"],
      style: TextStyle(color: Colors.white),
    ),
  );
  final cast = Container(
    padding: const EdgeInsets.all(7.0),
    decoration: new BoxDecoration(
      border: new Border.all(color: Colors.white),
      borderRadius: BorderRadius.circular(5.0)),
    child: new Text(
      snapshot["cast"],
      style: TextStyle(color: Colors.white),
    ),
  );
  final header = Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      Icon(
        Icons.movie,
        color: Colors.white,
        size: 20.0,
      ),
      Container(
        width: 90.0,
        child: new Divider(
          color: Colors.blue,

```



```

    ),
  ),
  SizedBox(height: 10.0),
  Text(
    snapshot["movieTitle"],
    style: TextStyle(color: Colors.white, fontSize: 30.0),
  ),
  SizedBox(height: 20.0),
  Text('Directed by:', style: TextStyle(color: Colors.white)),
  SizedBox(height: 10.0),
  Row(
    mainAxisAlignment: MainAxisAlignment.start,
    children: <Widget>[Expanded(child: rating)],
  ),
  SizedBox(height: 10.0),
  Text(
    'Cast: ',
    style: TextStyle(color: Colors.white),
  ), //SizedBox(height: 10.0),
  Row(
    mainAxisAlignment: MainAxisAlignment.start,
    children: <Widget>[Expanded(child: cast)],
  )
],
);

final headerContent = Stack(
  children: <Widget>[
    Container(
      padding: EdgeInsets.only(left: 10.0),
      height: MediaQuery.of(context).size.height * 0.52,
      decoration: new BoxDecoration(
        image: new DecorationImage(
          image:
            new NetworkImage(getMovieUrl(snapshot["movieImageUrl"])),
          fit: BoxFit.cover),
      )),
    Container(
      height: MediaQuery.of(context).size.height * 0.512,
      padding: EdgeInsets.all(40.0),
      width: MediaQuery.of(context).size.width,
      decoration: BoxDecoration(color: Color.fromRGBO(54, 60, 100, .9)),
      child: Center(
        child: header,
      ),
    ),
  ],
);

```

```

    ),
  ],
);

final movieDetail = Text(
  snapshot["description"],
  style: TextStyle(fontSize: 15.5),
);

final addReviewButtonFloating = FloatingActionButton.extended(
  label: Text('ADD REVIEW'),
  icon: Icon(Icons.rate_review),
  backgroundColor: Theme.of(context).primaryColor,
  onPressed: () {
    //Amashi start
    Navigator.of(context).push(MaterialPageRoute(
      builder: (context) => AddReviewPage(
        uid: uid,
        movieId: snapshot.documentID,
        movieTitle: snapshot["movieTitle"],
      )),
    //Amashi end
  },
);

final bottomContent = Container(
  width: MediaQuery.of(context).size.width,
  padding: EdgeInsets.all(40.0),
  child: Center(
    child: Column(
      children: <Widget>[movieDetail],
    ),
  ),
);

return new Scaffold(
  body: Column(
    children: <Widget>[headerContent, bottomContent],
  ),
  floatingActionButton: addReviewButtonFloating,
);
}

String getMovieUrl(String snapUrl) {
  if (snapUrl.isEmpty) return NetworkImagesPath.MOVIE_AVATAR;

```

```
    return snapUrl;
  }
}
```

1.6 Movie Model – Movie.dart

```
/*
 * IT17050272 - D. Manoj Kumar
 *
 * This movieAPI.dart file is consisting with services which used to fetch movie details from
 * firebase
 */

class Movie {
  String movieId;
  String movieTitle;
  String director;
  String movieImageUrl;
  String releaseYear;
  String description;

  Movie(
    {this.movieTitle,
    this.director,
    this.movieImageUrl,
    this.releaseYear,
    this.description});

  Movie.fromMap(Map snapshot, String movieId)
    : movieId = movieId ?? "",
      movieTitle = snapshot["movieTitle"],
      director = snapshot["director"],
      movieImageUrl = snapshot["movieImageUrl"],
      releaseYear = snapshot["releaseYear"],
      description = snapshot["description"];

  toJson() {
    return {
      "movieTitle": movieTitle,
      "director": director,
      "movieImageUrl": movieImageUrl,
      "releaseYear": releaseYear,
      "description": description
    };
  }
}
```

```
};  
}  
}
```

1.7 Services of Movie – movieAPI.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:flutter/foundation.dart';  
import 'package:movie_corns/api/models/Movie.dart';  
import 'package:movie_corns/locator.dart';  
  
/*  
 * IT17050272 - D. Manoj Kumar  
 *  
 * This movieAPI.dart file is consisting with services which used to fetch movie details from  
 * firebase  
 */  
  
class MovieApi {  
  final Firestore firestore = Firestore.instance;  
  final String path;  
  CollectionReference collectionReference;  
  
  MovieApi(this.path) {  
    collectionReference = firestore.collection(path);  
  }  
  
  Future<QuerySnapshot> getMovies() {  
    return collectionReference.getDocuments();  
  }  
  
  Stream<QuerySnapshot> streamMovies() {  
    return collectionReference.snapshots();  
  }  
  
  Future<DocumentSnapshot> getMovieByMovieId(String movieId) {  
    return collectionReference.document(movieId).get();  
  }  
}  
  
class MovieService extends ChangeNotifier {  
  MovieApi movieApi = locator<MovieApi>();  
  List<Movie> movies;
```

```

Future<List<Movie>> fetchMovies() async {
  var result = await movieApi.getMovies();
  movies = result.documents
    .map((doc) => Movie.fromMap(doc.data, doc.documentID))
    .toList();
  return movies;
}

Stream<QuerySnapshot> fetchMoviesAsStream() {
  return movieApi.streamMovies();
}

Future<Movie> getMovieById(String movieId) async {
  var doc = await movieApi.getMovieByMovieId(movieId);
  return Movie.fromMap(doc.data, doc.documentID);
}
}

```

1.8 Services of User Authentication – auth.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:shared_preferences/shared_preferences.dart';

/*
 * IT17050272 - D. Manoj Kumar
 *
 * This auth.dart file is consisting with services which used to authenticate logged user
 * & get the user ID of logged user
 * Also here is the source code for Signout function too
 */

class Auth {
  final FirebaseAuth firebaseAuth = FirebaseAuth.instance;

  void signOut() async {
    SharedPreferences preferences = await SharedPreferences.getInstance();
    preferences.clear();
    return firebaseAuth.signOut();
  }

  getCurrentUserId() async {
    FirebaseUser user = await firebaseAuth.currentUser();
    return user.uid;
  }
}

```

```
Future<FirebaseUser> getCurrentUser() async {  
  FirebaseUser user = await firebaseAuth.currentUser();  
  return user;  
}  
}
```

2.0 Screenshots of UI

2.1 Splash Screen / Login Screen

2.1.1 User Interface of Login Screen

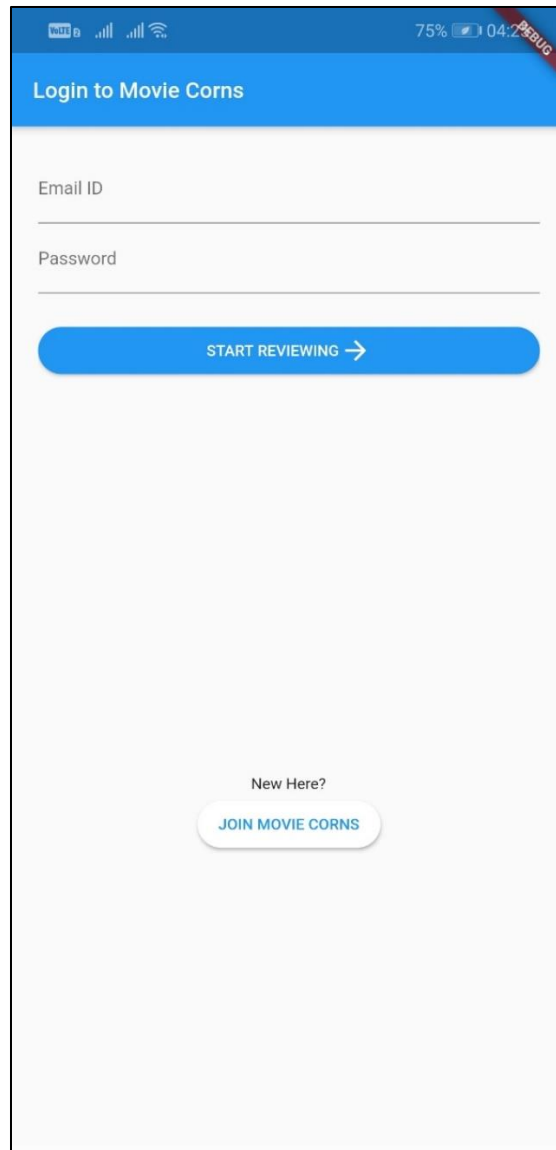


Figure 2. 1: Login Screen User Interface

2.1.2 Error Message Displaying

- 1 If user clicks on “START REVIEWING” button with empty fields in Email ID & Password following error message will be displaying:

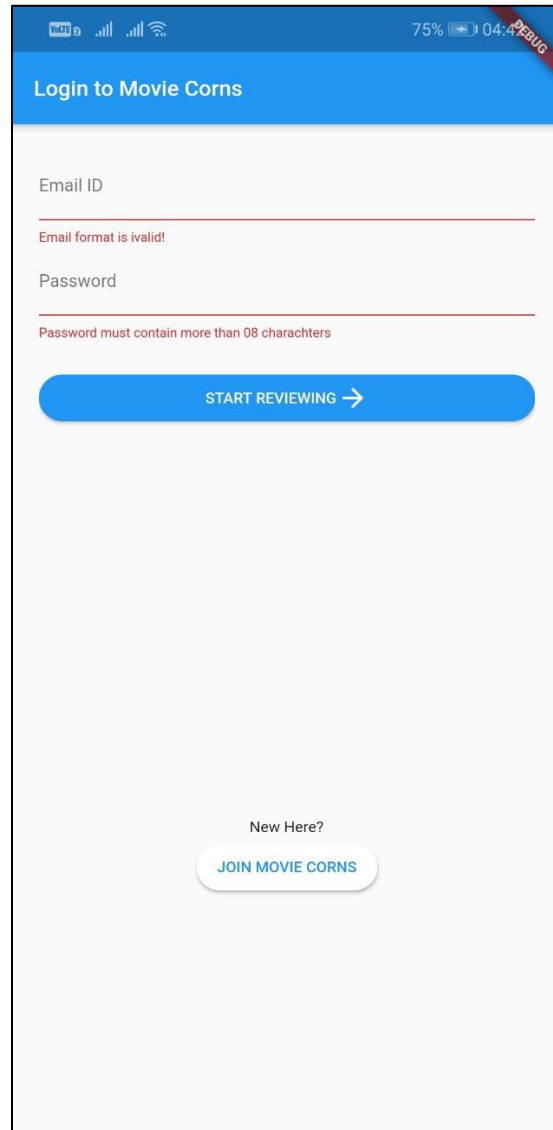


Figure 2. 2: Error Message type 1: Login Screen

- 2 If user enters invalid email ID while login to the system, then following error message will be displaying:

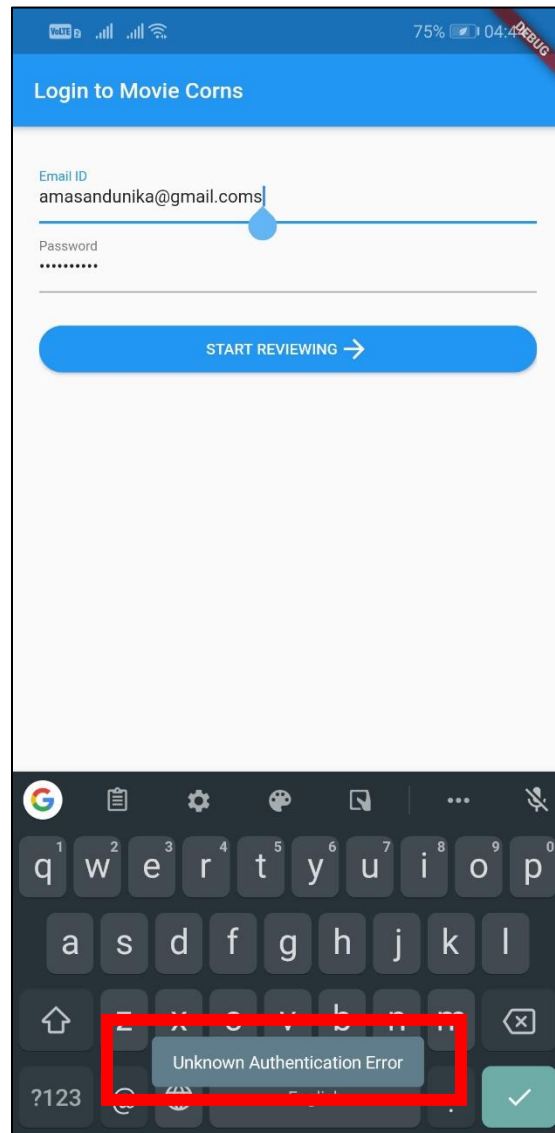


Figure 2. 3: Error Message type 2 – Login Screen

- 3 If user enters invalid password while login to the system, then following error message will be displaying:

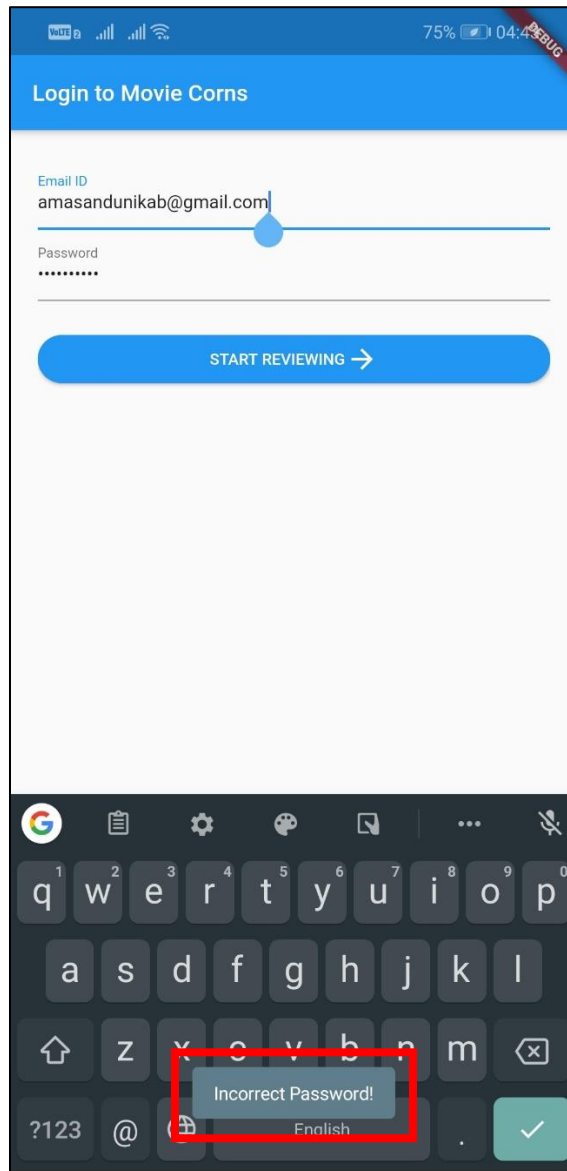


Figure 2. 4: Error Message type 3 – Login Screen

2.1.3 Successful System Login

If user successfully login to the system, then the system will redirect to the home page with the toast message, “Welcome”

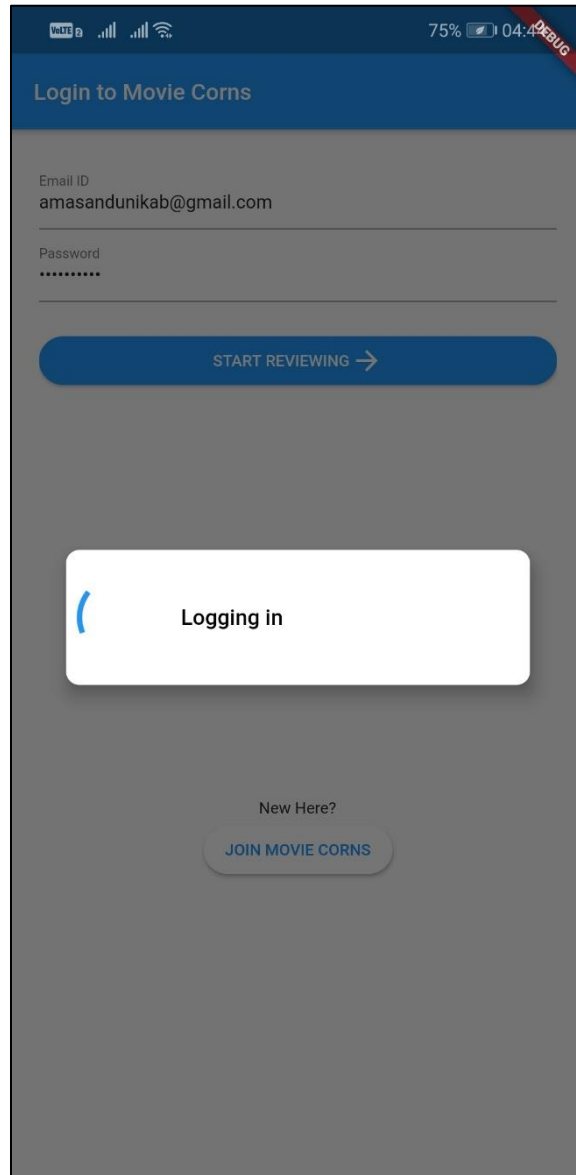


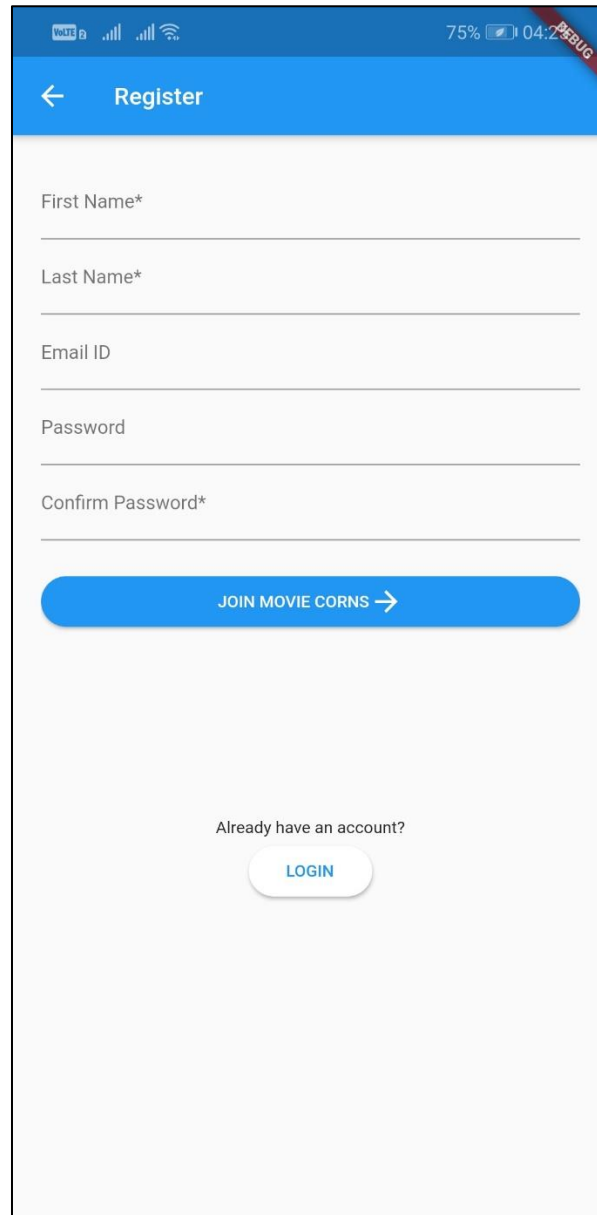
Figure 2. 5: Successful Login



Figure 2. 6: Redirect to Home Page After Successful Login

2.2 User Registration Page

2.2.1 User Interface for Registration Page



The image shows a mobile application interface for user registration. At the top, there is a status bar with icons for signal strength, Wi-Fi, and battery level (75%), along with the time 04:23. Below the status bar is a blue header with a back arrow and the text "Register". The main form area has a light gray background and contains five input fields: "First Name*", "Last Name*", "Email ID", "Password", and "Confirm Password*". Below these fields is a blue button with the text "JOIN MOVIE CORNS" and a right-pointing arrow. At the bottom, there is a link "Already have an account?" followed by a white button with the text "LOGIN". A red "DEBUG" banner is visible in the top right corner of the app screen.

Figure 2. 7: User Registration Form Interface

2.2.2 Error Messages Displaying

1. If user clicks on “JOIN MOVIE CORNS” without filling required fields, then the following error message will be displaying:

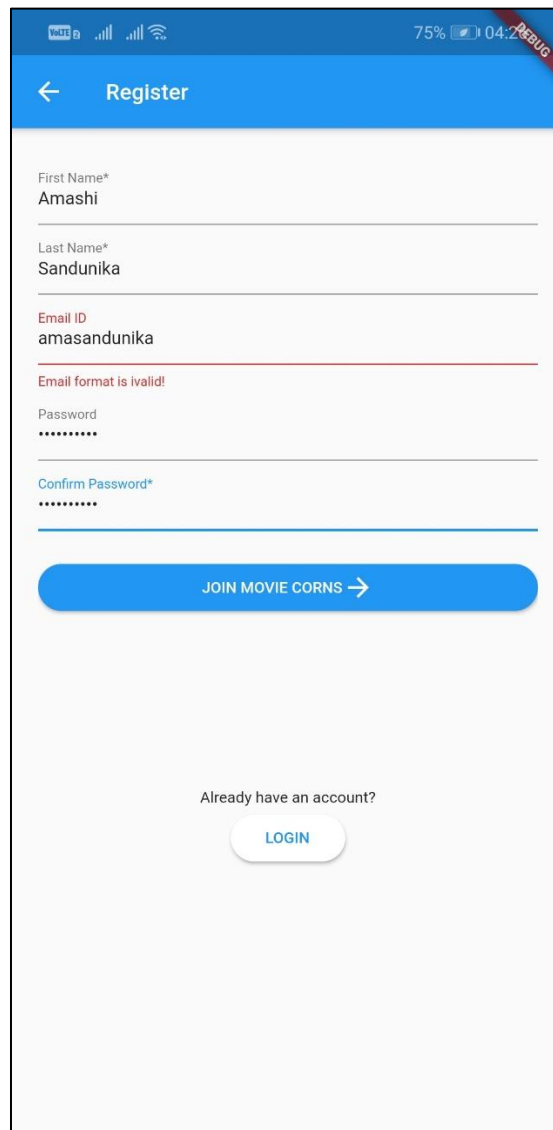
The screenshot shows a mobile application interface for a registration page. At the top, there is a blue header bar with a back arrow and the text "Register". Below the header, the page contains several input fields with associated error messages displayed in red text:

- First Name***: The input field is empty, and the error message "Please enter a valid first name." is displayed below it.
- Last Name***: The input field is empty, and the error message "Please enter a valid last name." is displayed below it.
- Email ID**: The input field is empty, and the error message "Email format is invalid!" is displayed below it.
- Password**: The input field is empty, and the error message "Password must contain more than 08 characters" is displayed below it.
- Confirm Password***: The input field is empty, and the error message "Password must contain more than 08 characters" is displayed below it.

Below the input fields, there is a large blue button with the text "JOIN MOVIE CORNS →". At the bottom of the page, there is a link that says "Already have an account?" followed by a button labeled "LOGIN".

Figure 2. 8: Error Message Type 1 - Signup Page

2. If the user entered Email ID does not compatible with correct email address format, then the following error message will be displaying:



The screenshot shows a mobile application interface for a registration page. At the top, there is a blue header bar with a back arrow and the word "Register". Below the header, the form contains several input fields: "First Name*" with the value "Amashi", "Last Name*" with the value "Sandunika", "Email ID" with the value "amasandunika", "Password" with masked characters "*****", and "Confirm Password*" with masked characters "*****". A red error message "Email format is invalid!" is displayed below the email input field. At the bottom of the form, there is a blue button labeled "JOIN MOVIE CORNS →". Below the button, there is a link "Already have an account?" with a "LOGIN" button underneath it. The status bar at the top of the phone shows 75% battery and the time 04:28.

Figure 2. 9: Error Message Type 2 – Signup Page

3. If the user entered password does not consist with 8 characters, then the following error message will be displaying:

The screenshot shows a mobile application interface for a registration page. At the top, there is a blue header bar with a back arrow and the text "Register". Below the header, the form contains the following fields and messages:

- First Name***: The text "Amashi" is entered.
- Last Name***: The text "Sandunika" is entered.
- Email ID**: The text "amasandunikab@gmail.com" is entered.
- Password**: Five dots are shown, indicating a password of length 5. Below this field, a red error message states: "Password must contain more than 08 charachters".
- Confirm Password***: Five dots are shown, indicating a password of length 5. Below this field, a red error message states: "Password must contain more than 08 charachters".

Below the password fields is a large blue button with the text "JOIN MOVIE CORNS →". At the bottom of the page, there is a link "Already have an account?" followed by a white button with the text "LOGIN". The status bar at the very top shows a battery level of 75% and the time 04:22. A red "DEBUG" label is visible in the top right corner of the app interface.

Figure 2. 10: Error Message Type 3 – Signup Page

4. If “Password” & “Confirm Password” fields do not match with each other, then the following error message will be displaying:

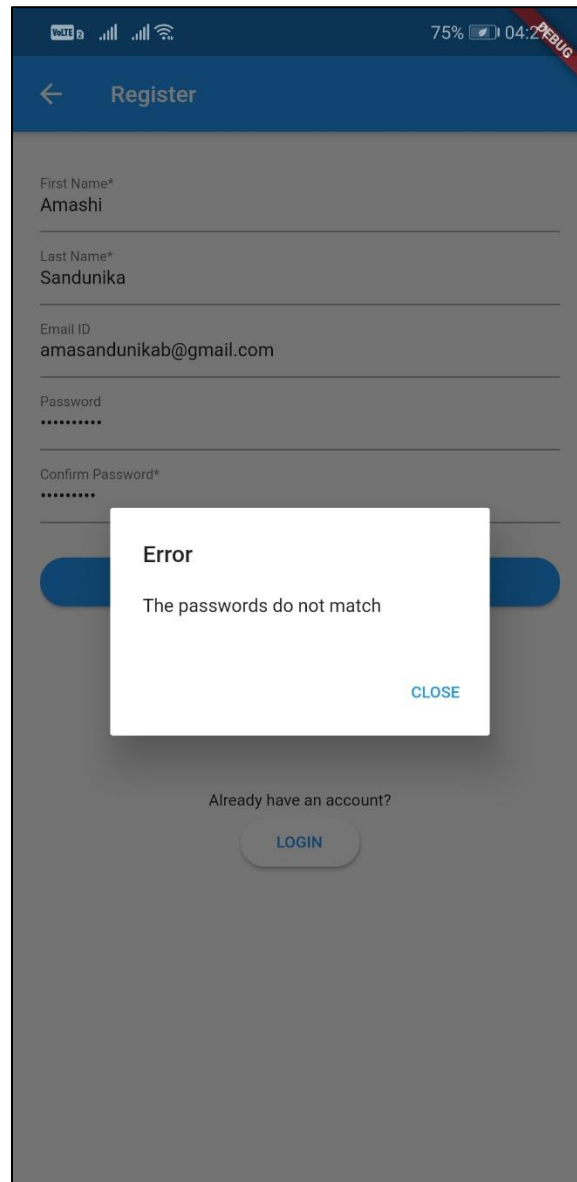


Figure 2. 11: Error Message Type 4 – Signup Page

2.2.3 Successful User Signup

If user successfully register to the system, the system will automatically redirect to home page

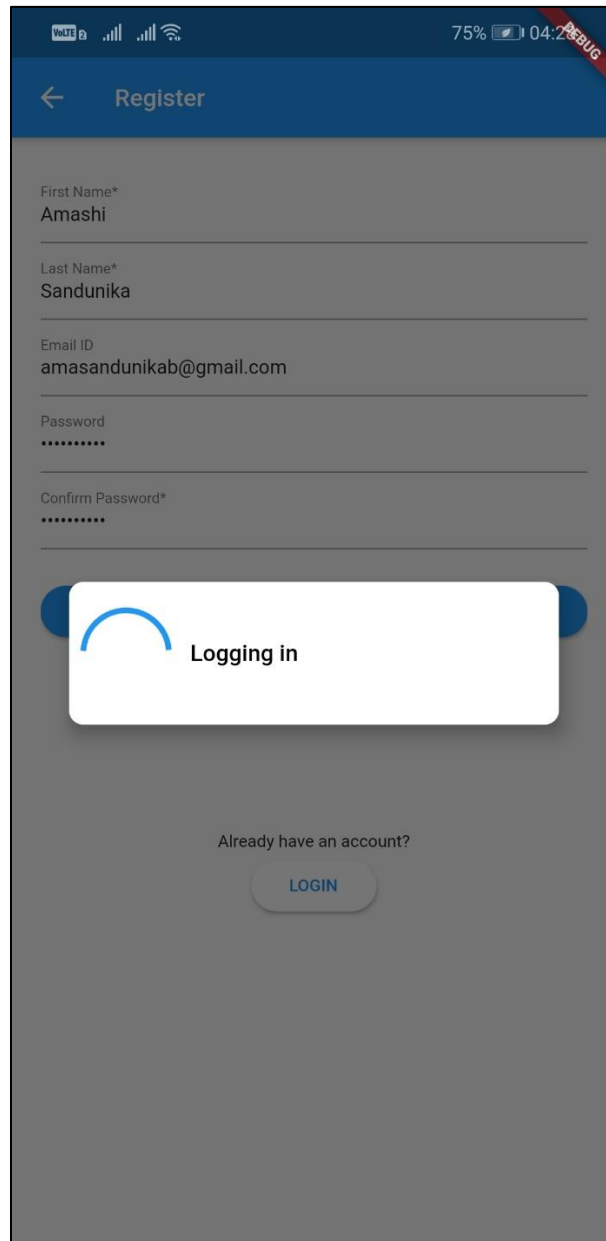


Figure 2. 12: Successful User Registration

2.3 User Profile

2.3.1 User Profile Interface

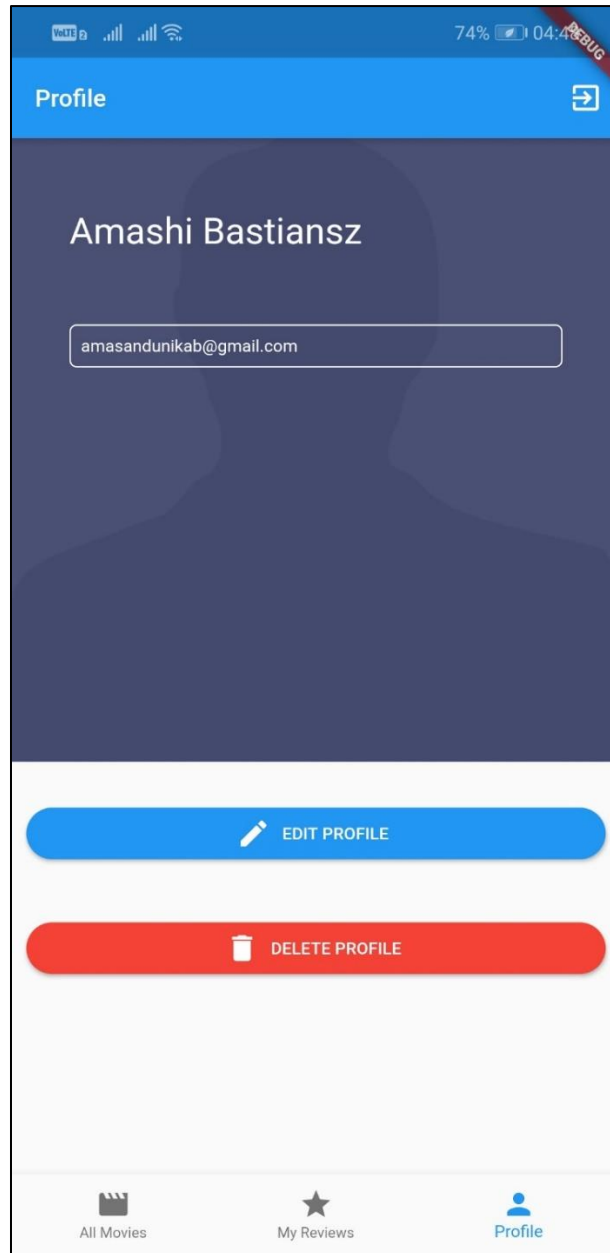


Figure 2. 13: User Profile Interface

2.3.2 Update User Profile

Steps to update user Profile:

1. Click on “Edit Profile” button in User Profile

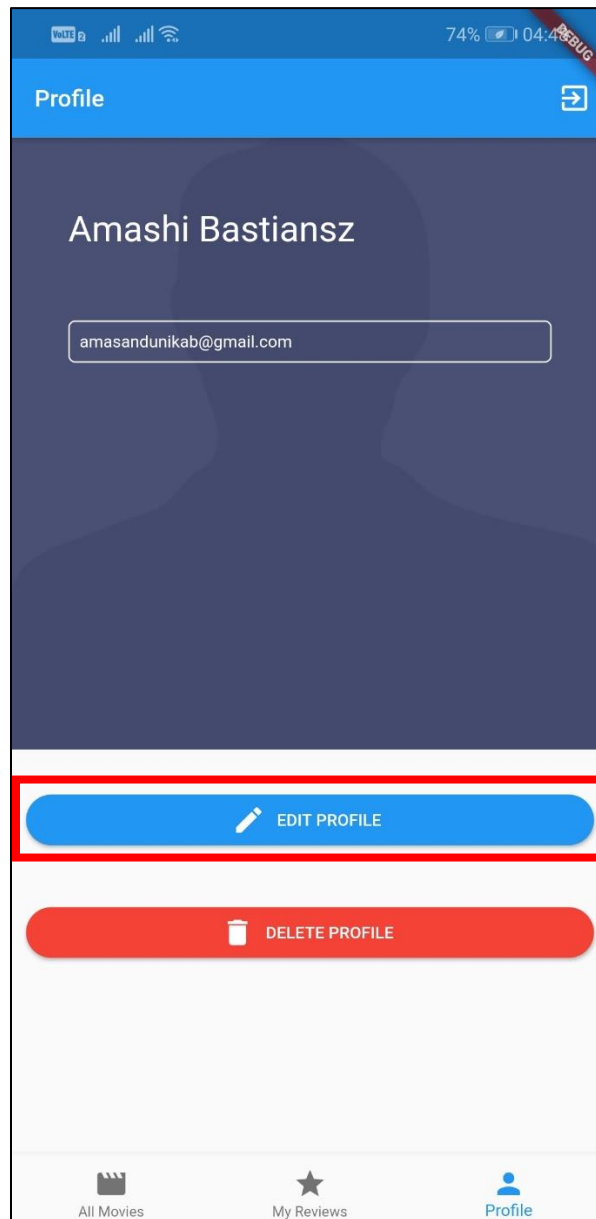


Figure 2. 14: Click "EDIT PROFILE" button

2. In seconds, the system will display a dialog box with previously entered user's first name & surname. Here, user can select the field he/she wants to edit & provide new value

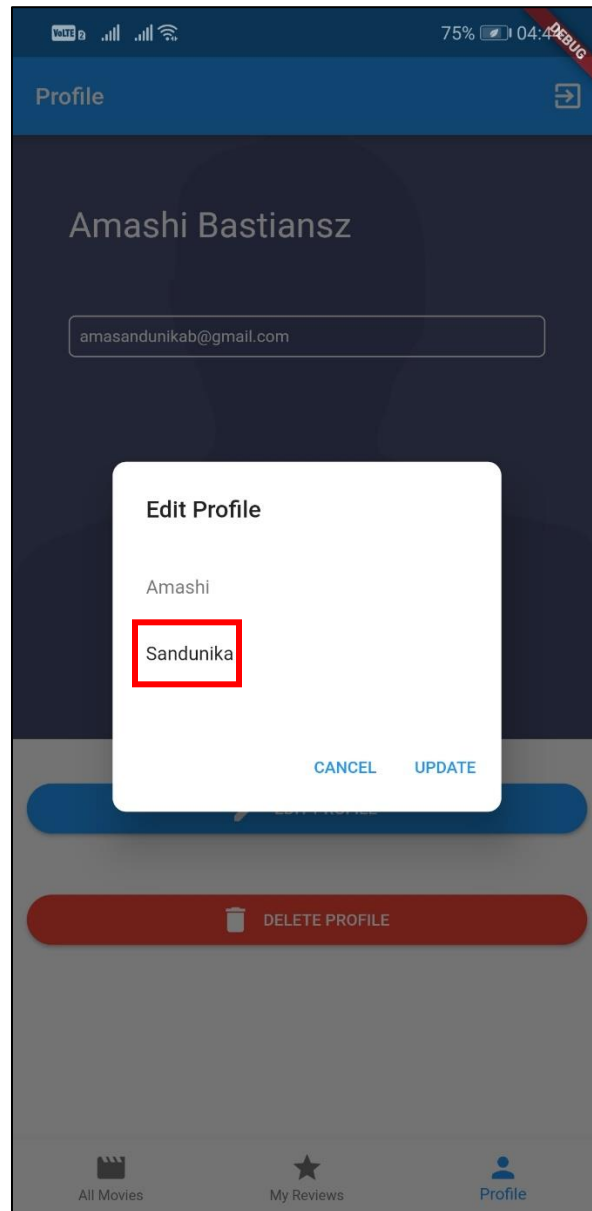


Figure 2. 15: Edit 'first name' & 'surname'

3. Then click on “Update” button. Then user will be redirected to the same user profile page with update value & a toast message telling that, profile updating was successful

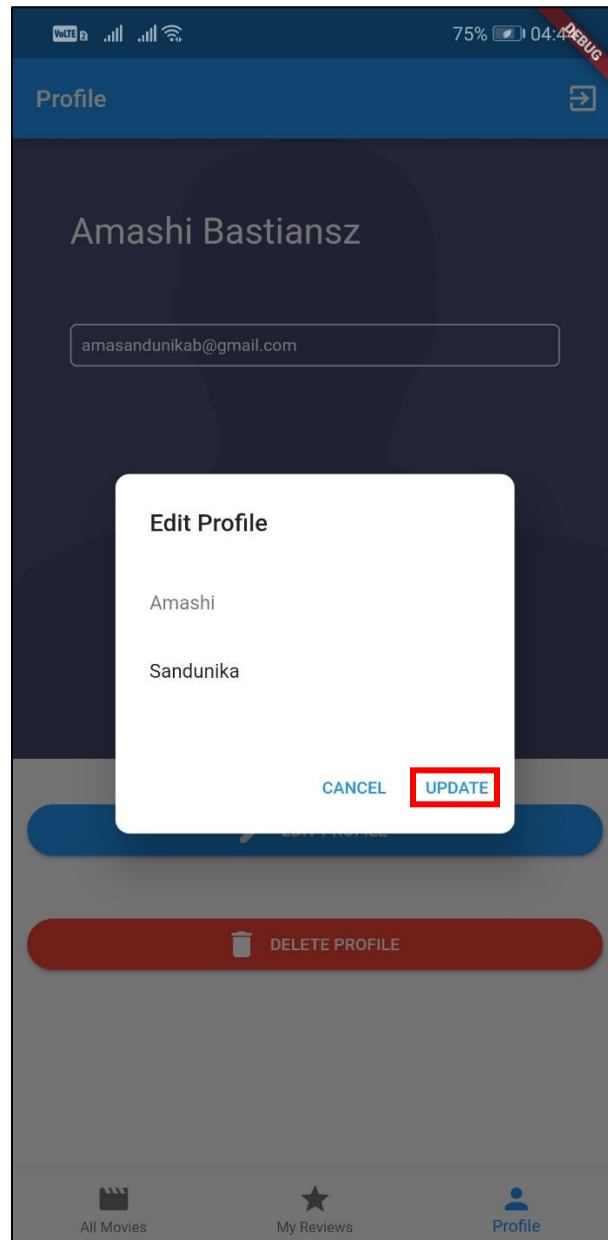


Figure 2. 16: Click on “UPDATE” Text

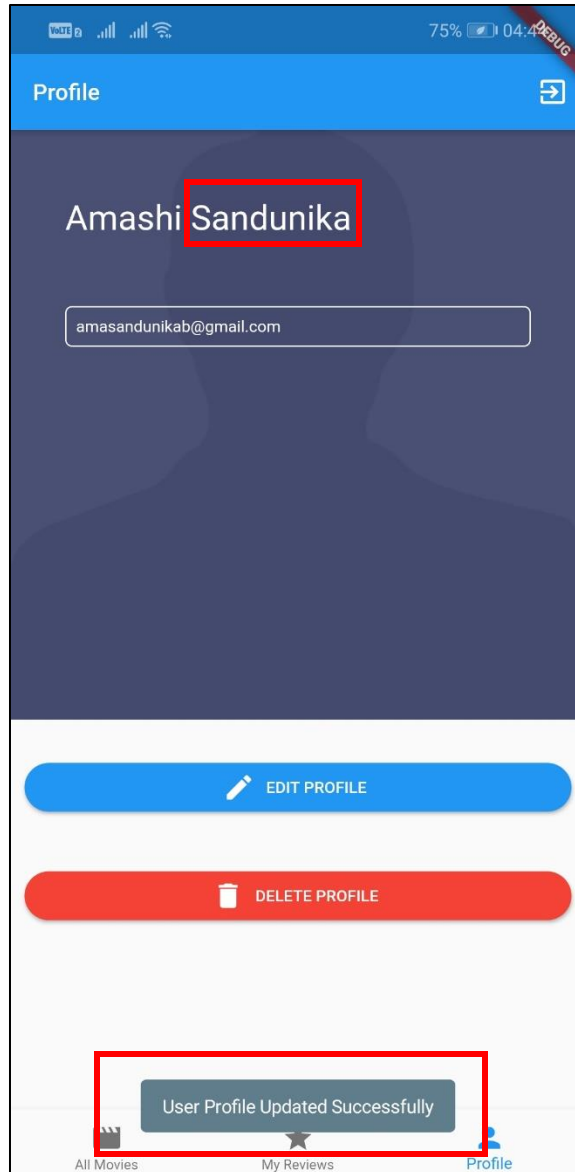


Figure 2. 17: Redirect to the User Profile with Toast Message & Updated Details

2.3.3 Delete User Profile

Steps to delete the User Profile:

1. Click on “DELETE PROFILE” button in User Profile

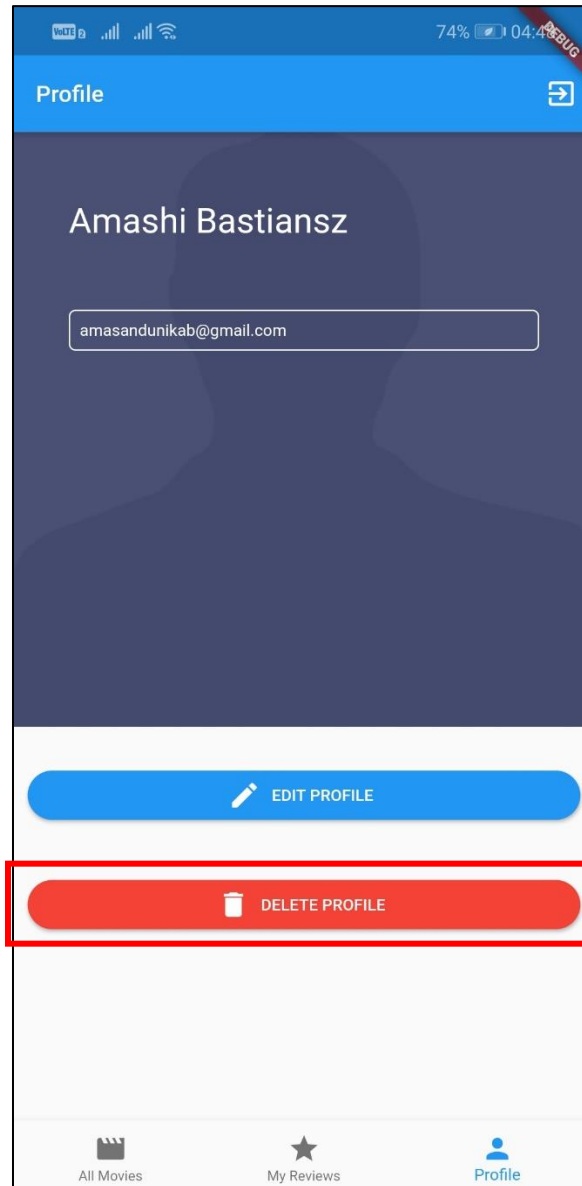


Figure 2. 18: Click on “DELETE PROFILE” button

2. Then the alert message box will be appeared asking to confirm the delete task. Click on “Delete” button in there.

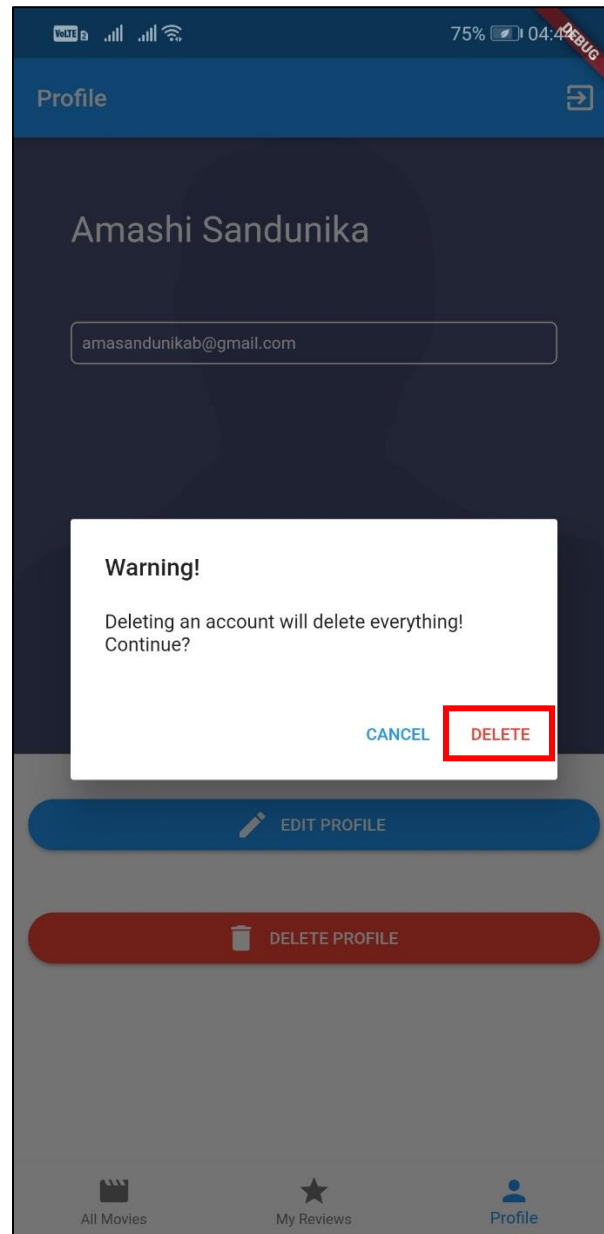


Figure 2. 19: Click on “DELETE” Text in Alert Message to Confirm Delete

3. Then the system will automatically redirect to login page with the toast message, “User Profile Deleted Successfully”

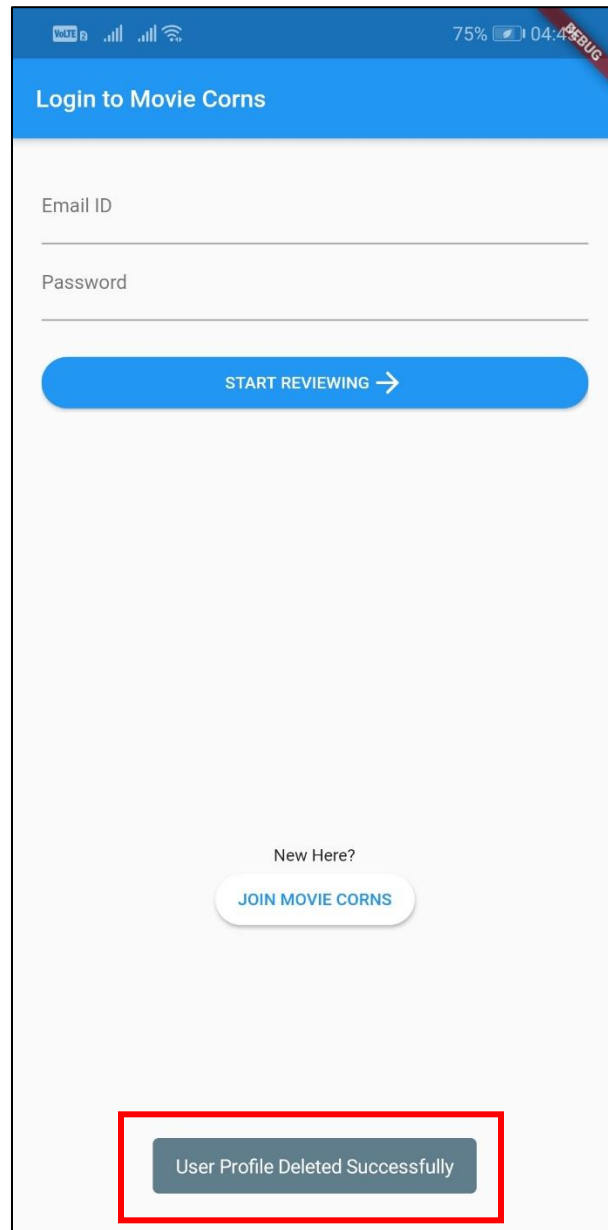


Figure 2. 20: Redirect to Login Page with Toast Message after Successful Delete

2.4 Movie Page

2.4.1 Multiple Movie Listing Page

Below is the user interface which used to display all the movies which are already stored in the system database. Every user will be redirected to this page soon after he/she login as well as sign up to the system successfully. Also, the user will be able to view this page, by clicking on the “All Movies” tab in bottom navigation bar.



Figure 2. 21: Movie Listing Interface

2.4.2 Single Movie Detail Page

After user clicks on any movie card displays in movie list page, he/she can view the single movie detail page. This page is consisting with the details of the movie, which user selected previously. In this page, the user will be able to see the image, the name, the case, a small description about the selected movie.

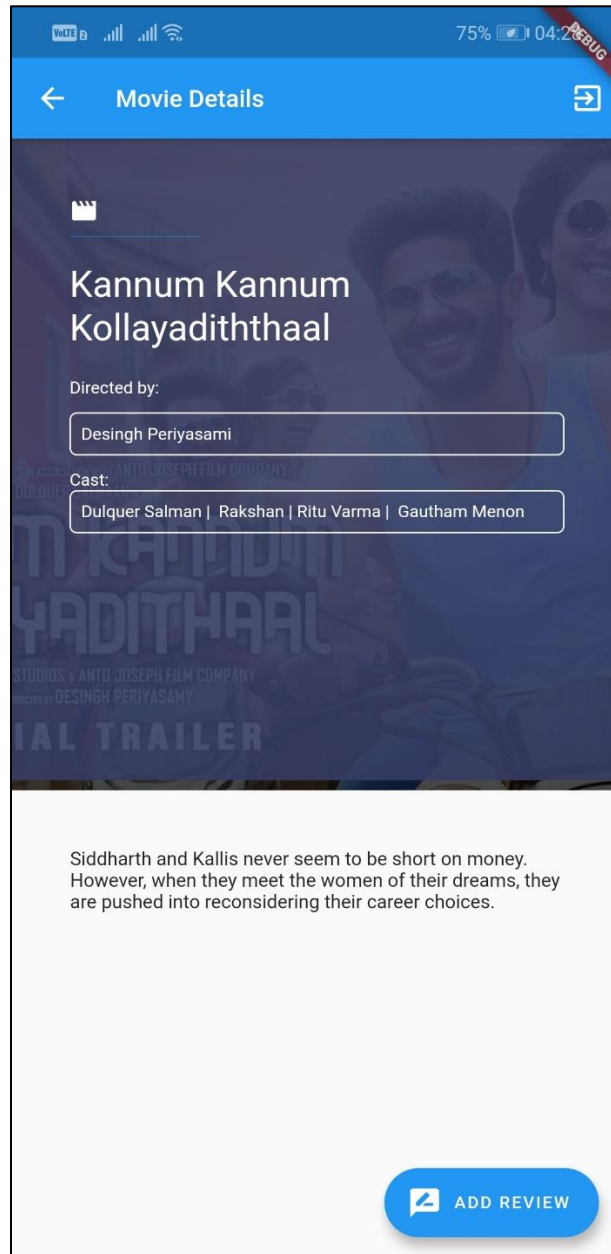


Figure 2. 22: Single Movie Detail Page Interface

2.5 App Sign-out

A logged user can sign-out from the app by clicking on the sign-out icon which is displayed in the right corner of top navigation. Steps to Sign-out:

1. Click on Sign-out icon

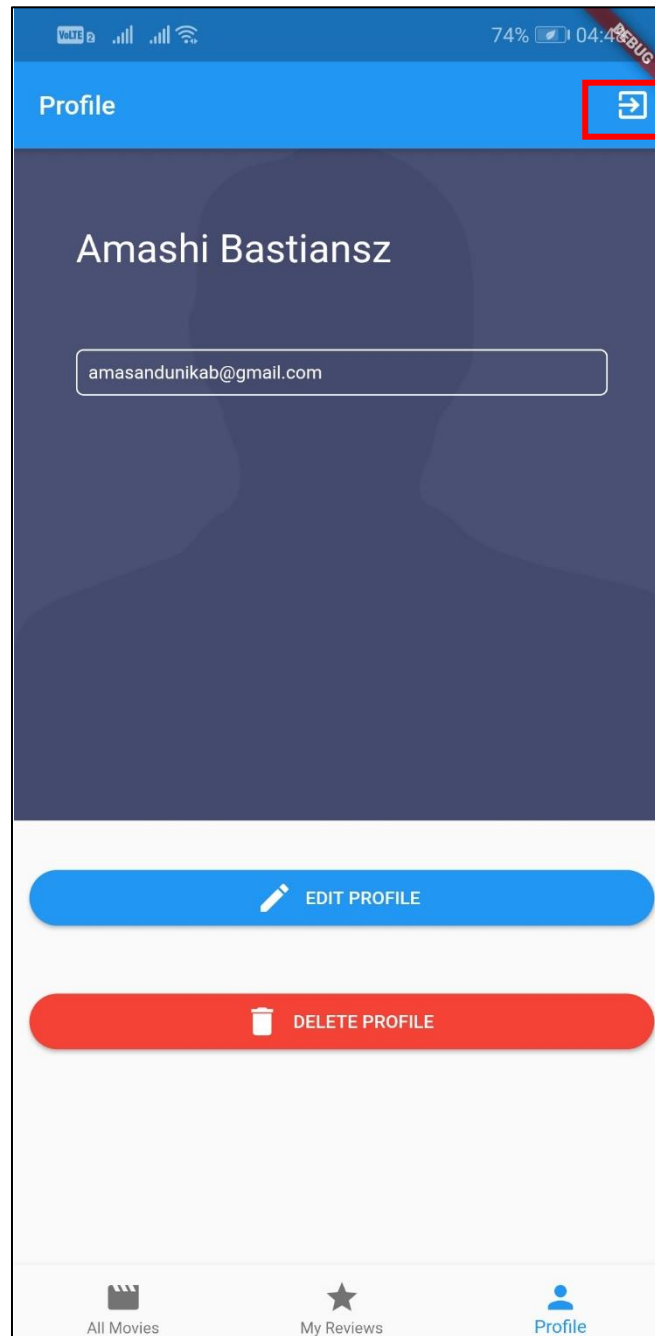


Figure 2. 23: Click on 'Sign-out' Icon

2. Click on “OK” button in alert message which is asked to confirm the sign-out. Then the system will redirected to the login page

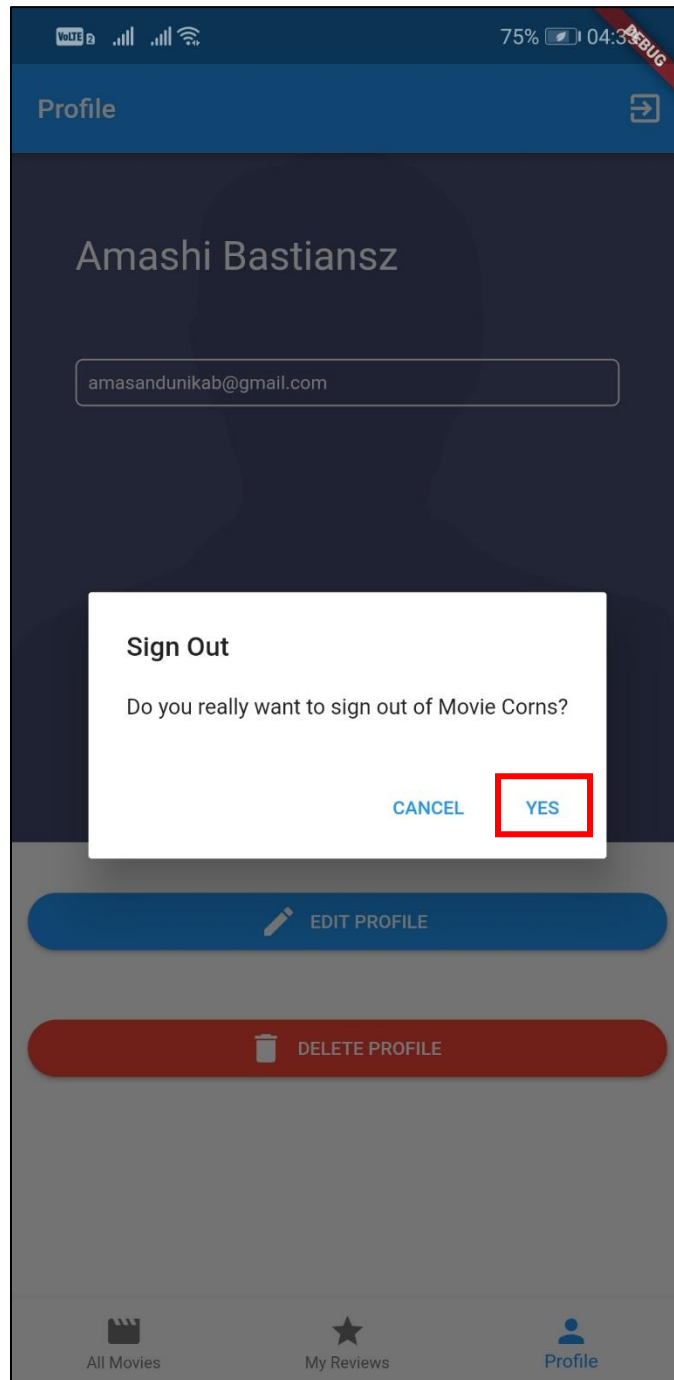


Figure 2. 24:Click on 'YES' Text in Alert Box

3.0 References

- [1] leisim, "A curated list of awesome Flutter packages.," [Online]. Available: <https://github.com/leisim/awesome-flutter-packages>. [Accessed March 2020].
- [2] "Flutter Documentation," [Online]. Available: <https://flutter.dev/docs>. [Accessed 20 March 2020].
- [3] "Adding a splash screen to your mobile app," [Online]. Available: <https://flutter.dev/docs/development/ui/advanced/splash-screen>. [Accessed 15 March 2020].
- [4] cimplesid, "flutter login ui," [Online]. Available: <https://github.com/cimplesid/loginui>. [Accessed March 2020].
- [5] A. Gupta, "Flutter hands on : Building a movie listing app using Flutter (Part 1)," [Online]. Available: <https://blog.usejournal.com/flutter-hands-on-building-a-movie-listing-app-using-flutter-part-1-c2b22d9be6b8>. [Accessed March 2020].

