



Sri Lanka Institute of Information Technology

Current Trends in Software Engineering

Semester 01 – Year 04 – 2020

Movie Reviewer

Group Project – Individual Submission

Managing CRUD operations for Movie Reviews

Submitted By:

G.M.A.S. Bastiansz

IT17143950

April 30th 2020

Table of Contents

List of Figures	iii
1.0 Source Code	4
1.1 Model for Review – Review.dart	4
42.1 Add Movie Page – add_review.dart	5
42.2 My Reviews Page – my_reviews.dart	10
2.0 Screenshots of UI	21
2.1 Add Movie Review Page	21
2.1.1 Add Movie Review Form User Interface	21
2.1.2 Steps to Add a Review	22
2.2 My Reviews Page	26
2.2.1 My Reviews Page User Interface	26
2.2.2 View Reviews	27
2.2.2 Update a Selected Movie Review	28
2.2.3 Delete a Selected Movie Review	33
3.0 References	36

List of Figures

Figure 2. 1: Add Movie Reviews Form	21
Figure 2. 2: Click on “ADD REVIEW” Button	22
Figure 2. 3: Enter Comment & Rating	23
Figure 2. 4: Click on “SUBMIT REVIEW” Button	24
Figure 2. 5: Display Toast Message	25
Figure 2. 6: My Reviews User Interface	26
Figure 2. 7: Sections in My Reviews Page	27
Figure 2. 8: Click on “Edit” Icon	28
Figure 2. 9: View Dialog Box with Previously Entered Review	29
Figure 2. 10: Enter new Review in the Dialog Box	30
Figure 2. 11: Click on “UPDATE” Text	31
Figure 2. 12: Display the Appropriate Toast Message	32
Figure 2. 13: Click on “Delete” Icon	33
Figure 2. 14: Click on “DELETE” Text in Alert Box	34
Figure 2. 15: View Appropriate Toast Message After Successful Delete Task	35

1.0 Source Code

All the source codes mentioned below are not screenshots

1.1 Model for Review – Review.dart

```
2  /*
3   * IT17050272 - D. Manoj Kumar
4   *
5   * This movieAPI.dart file is consisting with services which used to fetch mo
  vie details from
6   * firebase
7   */
8
9  class Movie {
10   String movieId;
11   String movieTitle;
12   String director;
13   String movieImageUrl;
14   String releaseYear;
15   String description;
16
17   Movie(
18     {this.movieTitle,
19     this.director,
20     this.movieImageUrl,
21     this.releaseYear,
22     this.description});
23
24   Movie.fromMap(Map snapshot, String movieId)
25     : movieId = movieId ?? '',
26       movieTitle = snapshot["movieTitle"],
27       director = snapshot["director"],
28       movieImageUrl = snapshot["movieImageUrl"],
29       releaseYear = snapshot["releaseYear"],
30       description = snapshot["description"];
31
32   toJson() {
33     return {
34       "movieTitle": movieTitle,
35       "director": director,
36       "movieImageUrl": movieImageUrl,
37       "releaseYear": releaseYear,
38       "description": description
```

```

39     };
40   }
41 }
42

```

42.1 Add Movie Page – add_review.dart

```

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

/*
 * IT17143950 - G.M.A.S. Bastiansz
 *
 * The task of this add_review.dart file is providing a form to submit
 * the review of the user for a selected movie. User is able to provide
 * a comment & a rate (1-5) for the selected movie. Once the user submit
 * the Add Movie Review form, the entered comment & rating store in Firebase
 * database with the user ID, movie ID & the name of the selected movie
 */

class AddReviewPage extends StatefulWidget {
  final movieId;
  final uid;
  final movieTitle;

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

  @override
  ReviewFormBodyState createState() => new ReviewFormBodyState();
}

class ReviewFormBodyState extends State<AddReviewPage> {
  final _formKey = GlobalKey<FormState>();
  final reference = Firestore.instance.collection('reviews').reference();
  double rating = 0.0;
  String review = "";

  //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
Widget build(BuildContext context) {
    const edgeInsetValue = 8.0;
    final Auth auth = new Auth();

    print("Movie id:");
    print(widget.movieId.toString());

    print("User id:");
    print(widget.uid);

    //start 'Add Movie Review' page
    return Container(
        child: Scaffold(
            appBar: AppBar(
                title: Text(TitleConstants.ADD_MOVIE_REVIEW),
                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);
        },
      ),
    ],
  );
});

},
),
],
),

//start 'Add Movie Review' form
body: Container(
  padding: const EdgeInsets.fromLTRB(
    edgeInsetValue, 50.0, edgeInsetValue, edgeInsetValue),
  child: ListView(
    children: <Widget>[
      Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            SizedBox(
              height: 30.0,
            ),

            //Start 'Comment' field
            Padding(
              padding: EdgeInsets.all(3.0),
              child: Text(
                LabelConstants.LABEL_COMMENT_FIELD,
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.black,
                  fontSize: 20),
              ),
            ),
            TextFormField(
              maxLines: 4,
              validator: (comment) {
                setState(() {

```

```

        this.review = comment;
    });

    //Validating the 'Comment' field for empty values
    if (comment.isEmpty) {
        return ValidatorConstants.COMMENT_CANNOT_NULL;
    }
    return null;
},
decoration: InputDecoration(
    contentPadding:
        EdgeInsets.fromLTRB(10.0, 35.0, 10.0, 5.0),
    filled: true,
    fillColor: Colors.white,
    hintText: HintTextConstants.HINT_COMMENT_FIELD,
    border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(10.0))),
),
//End 'Comment' field
//Start 'Rate Us' field
Padding(
    padding: EdgeInsets.all(12.0),
    child: Text(
        LabelConstants.LABEL_RATING_FIELD,
        textAlign: TextAlign.right,
        style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.black,
            fontSize: 20),
    ),
),
Padding(
    padding: EdgeInsets.all(12.0),
    child: SmoothStarRating(
        allowHalfRating: true,
        onRatingChanged: (rateVal) {
            setState(() {
                rating = rateVal;
                //print('The rating is $rating');
            });
        },
        starCount: 5,
        rating: this.rating,
        size: 40,
        filledIconData: Icons.star,

```



```

        halfFilledIconData: Icons.star_half,
        color: Colors.blueAccent,
        borderColor: Colors.blue,
        spacing: 0.0,
      ),
    ),
    //End 'Rate Us' field
    //Start button row
    Padding(
      padding: const EdgeInsets.fromLTRB(50.0, 20.0, 10.0, 5.0),
      child: Row(
        children: <Widget>[
          //Start 'Submit Review' button
          RaisedButton(
            onPressed: () {
              if (_formKey.currentState.validate()) {
                addReviewFirebase(
                  review,
                  rating,
                  widget.movieId,
                  widget.uid,
                  widget.movieTitle);
              }
              Navigator.pushNamedAndRemoveUntil(
                context, "/home", (_) => false);

              toast(
                ToastConstants.ADD_REVIEW_SUCCESS,
                Toast.LENGTH_LONG,
                ToastGravity.BOTTOM,
                Colors.blueGrey);
            },
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(19)),
            color: Colors.blue,
            child: Text(
              ButtonConstants.ADD_REVIEW,
              style: TextStyle(color: Colors.white),
            ),
          ),
        ],
      ),
    ),
    SizedBox(
      height: 10.0,
      width: 50.0,
    ),
    //End 'Submit Review' button

```

```

        //Start 'Cancel Review' button
        RaisedButton(
          onPressed: () {
            Navigator.pushNamedAndRemoveUntil(
              context, "/movie", (_) => false);
          },
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(19)),
          color: Colors.red,
          child: Text(
            ButtonConstants.CANCEL_REVIEW,
            style: TextStyle(color: Colors.white),
          ),
        )
        //End 'Cancel Review' button
      ],
    )),
  //End Button row
],
),
),
),
),
)),
}

//this method is used to add the entered review to the system database
void addReviewFirebase(String review, double rating, String movieId,
  String uid, String movieTitle) {
  Map<String, dynamic> data = Map();
  data['review'] = "$movieTitle - $review";
  data['rating'] = rating;
  data['movieId'] = movieId;
  data['uid'] = uid;

  reference.add(data);
}
}

```

42.2 My Reviews Page – my_reviews.dart

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

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:smooth_star_rating/smooth_star_rating.dart';
import 'package:movie_corns/api/services/auth.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:movie_corns/constants/constants.dart';

/*
 * IT17143950 - G.M.A.S. Bastiansz
 *
 * The tasks of my_review.dart file are providing relevant user interfaces as well as
 * backend codes to view all the reviews provided by the logged user, update a selected review
 * & delete reviews. According to the below source code the logged user can view all the reviews he/she
 * had provided to movies in a list view. The list with separate cards. Each card displays
 * separate review. One card view consists with the name of the movie which user had reviewed,
 * the review comment & also the rating. Also each card has "Edit" icon & "Delete" icon which
 * can be used to fulfill update & delete tasks respectively.
 * According to the below code when the user clicks on Edit Icon, he/she will be able to update the
 * review which already provided. In order to firestore query, the user can only update the comment he/she
 * provided before. Once the "edit" icon clicks, the system will prompt a dialog box with the old comment.
 * If user wishes to edit it, he/she can give the new comment & clicks on "Update" button. The app will
 * automatically redirect to the My Reviews page while replacing the new comment instead of old comment.
 * When the user clicks on "Delete" icon, the system will prompt an alert message to confirm the delete task.
 * After user approves it, the system will erase the review from the whole database while refreshing to the
 * My Reviews page without the deleted review
 * These are the overall tasks fulfilled through this dart file
 */

class MyReviewsPage extends StatefulWidget {
  final String title;
  final dynamic uid;
  final movieId;
  final reviewId;

```

```

MyReviewsPage({Key key, this.title, this.uid, this.movieId, this.reviewId})
    : super(key: key);

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

class _MyReviewsPageState extends State<MyReviewsPage> {
    final Auth auth = new 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);
    }

    static final fullStar = Icon(
        Icons.star,
        color: Colors.blueAccent,
    );

    static final halfStar = Icon(
        Icons.star_half,
        color: Colors.blueAccent,
    );

    static final starBorder = Icon(
        Icons.star_border,
        color: Colors.blueAccent,
    );

    static final starRow = Row(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[fullStar, fullStar, fullStar, halfStar, starBorder],
    );
    String uid = "";

    @override

```

```

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

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(TitleConstants.MY_REVIEWS),
      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: Container(child: _buildReviewBody(context));
}

Widget _buildReviewBody(BuildContext context) {
  return StreamBuilder<QuerySnapshot>(
    stream: Firestore.instance
      .collection('reviews')
      .where('uid', isEqualTo: uid)
      .snapshots(),
    builder: (context, snapshot) {
      if (!snapshot.hasData || snapshot.data.documents.length < 1) {
        return Text(TitleConstants.NO_REVIEWS);
      } else {
        return _buildReviewList(context, snapshot.data.documents);
      }
    },
  );
}

Widget _buildReviewList(
  BuildContext context, List<DocumentSnapshot> snapshot) {
  return ListView(
    padding: const EdgeInsets.only(top: 20.0),
    children: <Widget>[
      Row(
        children: <Widget>[
          Padding(
            padding: EdgeInsets.all(5.0),
            child: _displayStars(snapshot),
          ),
          Padding(
            padding: EdgeInsets.all(5.0),
            child: Text('${snapshot.length} reviews'),
          ),
        ],
      ),
      Row(
        children: <Widget>[
          SizedBox(
            height: 30.0,
          ),
        ],
      ),
    ],
  );
}

```

```

        Padding(
          padding: EdgeInsets.all(5.0),
          child: Text(TitleConstants.ALL_YOUR_REVIEWS),
        )
      ],
    ),
    Column(
      children: _buildReviewTiles(context, snapshot),
    )
  ],
);
}

List<Widget> _buildReviewTiles(
  BuildContext context, List<DocumentSnapshot> snapshot) {
  return (snapshot.map((data) => _buildReviewItem(context, data)).toList());
}

Widget _buildReviewItem(BuildContext context, DocumentSnapshot data) {
  final record = Record.fromSnapshot(data);
  return _buildReviewCard('imagePath', record);
}

Widget _buildReviewCard(String path, Record record) {
  return Padding(
    padding: EdgeInsets.only(),
    child: Container(
      //height: MediaQuery.of(context).size.height / 2.8,
      width: MediaQuery.of(context).size.width,
      child: new GestureDetector(
        child: new Card(
          child: ListTile(
            leading: CircleAvatar(
              child: Icon(Icons.rate_review),
            ),
            title: Row(
              children: <Widget>[
                SmoothStarRating(
                  allowHalfRating: false,
                  starCount: 5,
                  rating: record.rating.toDouble(),
                  size: 20.0,
                  filledIconData: Icons.star,
                  halfFilledIconData: Icons.star_border,
                  color: Colors.blueAccent,

```

```

        borderColor: Colors.blue,
        spacing: 0.0,
      ),
    ],
  ),
  subtitle: Padding(
    padding: EdgeInsets.all(12.0),
    child: Text("${record.review}"),
  ),
  trailing: Row(
    mainAxisAlignment: MainAxisAlignment.min,
    children: <Widget>[
      IconButton(
        icon: new Icon(Icons.edit),
        color: Colors.green,
        onPressed: () {
          _displayReviewUpdateDialog(context, record);
        },
      ),
      IconButton(
        icon: new Icon(Icons.delete),
        color: Colors.red,
        onPressed: () {
          showDialog(
            context: context,
            builder: (context) {
              return AlertDialog(
                title: Text(TitleConstants.ALERT_WARNING),
                content: Text(PromptConstants
                  .QUESTION_CONFIRM_REVIEW_DELETE),
                actions: <Widget>[
                  FlatButton(
                    child:
                      Text(ButtonConstants.OPTION_CANCEL),
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                  ),
                  FlatButton(
                    child:
                      Text(ButtonConstants.OPTION_DELETE),
                    onPressed: () {
                      final Firestore firestore =
                        Firestore.instance;

```



```

        firestore
            .collection('reviews')
            .getDocuments()
            .then((snap) {
                for (DocumentSnapshot snapshot
                    in snap.documents) {
                    if (snapshot.documentID ==
                        record.reference.documentID) {
                        snapshot.reference.delete();
                    }
                }
            });

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

        Navigator.of(context).pop();
    },
    ),
    ],
    );
    });
    })
    ],
    ),
    isThreeLine: true,
    ),
    ))));
}

```

```

_displayReviewUpdateDialog(BuildContext context, Record record) async {
    TextEditingController commentController = new TextEditingController();

    return showDialog(
        context: context,
        builder: (BuildContext context) {
            return AlertDialog(
                title: Text(TitleConstants.UPDATE_MOVIE_REVIEW),
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10.0)),
                content: Container(

```

```

height: 100,
child: Container(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      TextField(
        controller: commentController,
        decoration: InputDecoration(
          border: InputBorder.none, hintText: record.review),
      ),
    ],
  ),
),
),
actions: <Widget>[
  FlatButton(
    child: Text(ButtonConstants.OPTION_CANCEL),
    onPressed: () {
      Navigator.of(context).pop();
    },
  ),
  FlatButton(
    child: Text(ButtonConstants.OPTION_UPDATE),
    onPressed: () {
      String commentText = "";

      if (commentController.text.isEmpty) {
        commentText = record.review;
      } else {
        commentText = record.review.split("-")[0] +
          " - " +
          commentController.text;
      }

      print(widget.uid);

      Firestore.instance
        .collection('reviews')
        .document(record.reference.documentID)
        .setData({
          "review": commentText,
          "rating": record.rating,
          "movieId": record.movieId,
          "uid": uid

```

```

        });
        Navigator.of(context).pop();

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

        return true;
    })
    ],
    );
});
}

double _calAvgReview(List<DocumentSnapshot> snapshot) {
    double sum = 0.0;

    for (var i = 0; i < snapshot.length; i++) {
        Record record = Record.fromSnapshot(snapshot[i]);
        sum += record.rating.toDouble();
    }

    return sum / snapshot.length;
}

Widget _displayStars(List<DocumentSnapshot> snapshot) {
    return SmoothStarRating(
        allowHalfRating: true,
        starCount: 5,
        rating: _calAvgReview(snapshot),
        size: 40.0,
        filledIconData: Icons.star,
        halfFilledIconData: Icons.star_half,
        color: Colors.blueAccent,
        borderColor: Colors.blue,
        spacing: 0.0,
    );
}

class Record {
    String review;
    num rating;
}

```

```

String movieId;
final DocumentReference reference;

Record.fromMap(Map<String, dynamic> map, {this.reference})
    : assert(map['review'] != null),
      assert(map['rating'] != null),
      assert(map['movieId'] != null),
      review = map['review'],
      rating = map['rating'],
      movieId = map['movieId'];

Record.fromSnapshot(DocumentSnapshot snapshot)
    : this.fromMap(snapshot.data, reference: snapshot.reference);

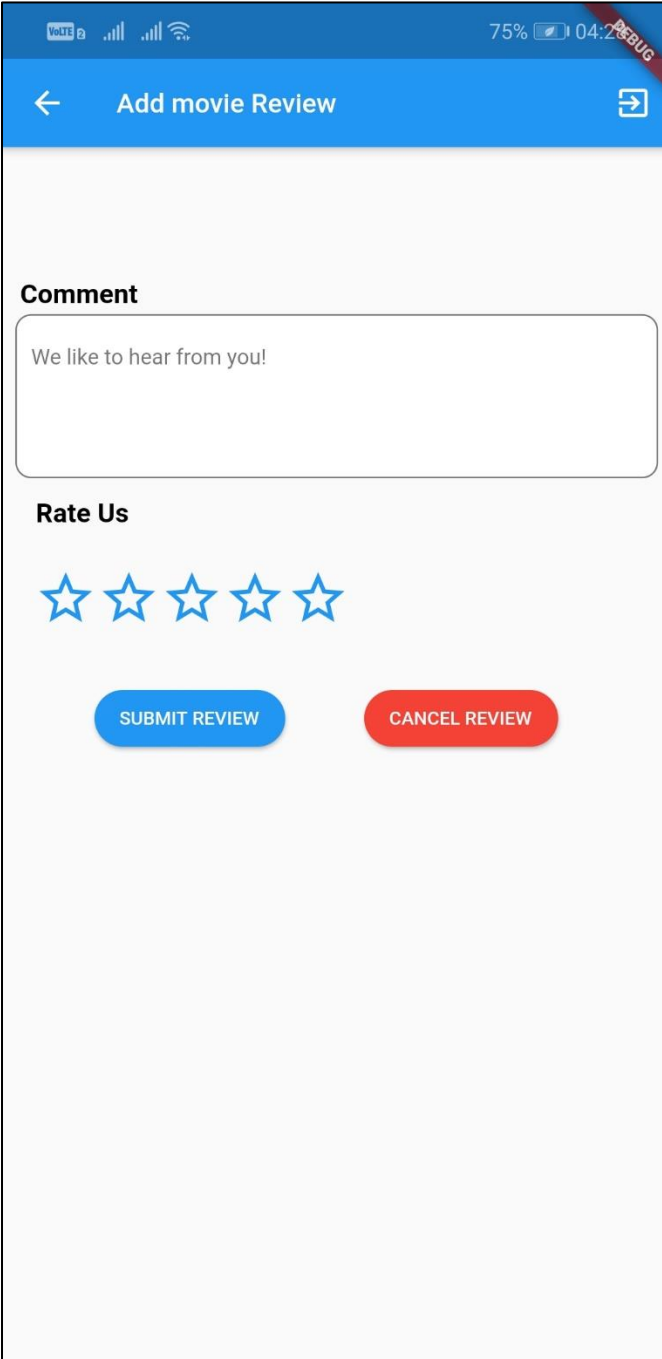
@override
String toString() => "Record <$review>";
}

```

2.0 Screenshots of UI

2.1 Add Movie Review Page

2.1.1 Add Movie Review Form User Interface



The screenshot displays a mobile application interface for adding a movie review. At the top, a blue header bar contains a back arrow, the title 'Add movie Review', and a share icon. The status bar above shows 'VoLTE', signal strength, Wi-Fi, 75% battery, and the time 04:23. A red 'DEBUG' banner is visible in the top right corner. The main content area is light gray and contains two sections: 'Comment' and 'Rate Us'. The 'Comment' section has a text input field with the placeholder text 'We like to hear from you!'. The 'Rate Us' section features five empty blue star icons. At the bottom, there are two rounded buttons: a blue 'SUBMIT REVIEW' button and a red 'CANCEL REVIEW' button.

Figure 2. 1: Add Movie Reviews Form

2.1.2 Steps to Add a Review

1. Click on “ADD REVIEW” button at the bottom of the Movie Details page for any selected movie

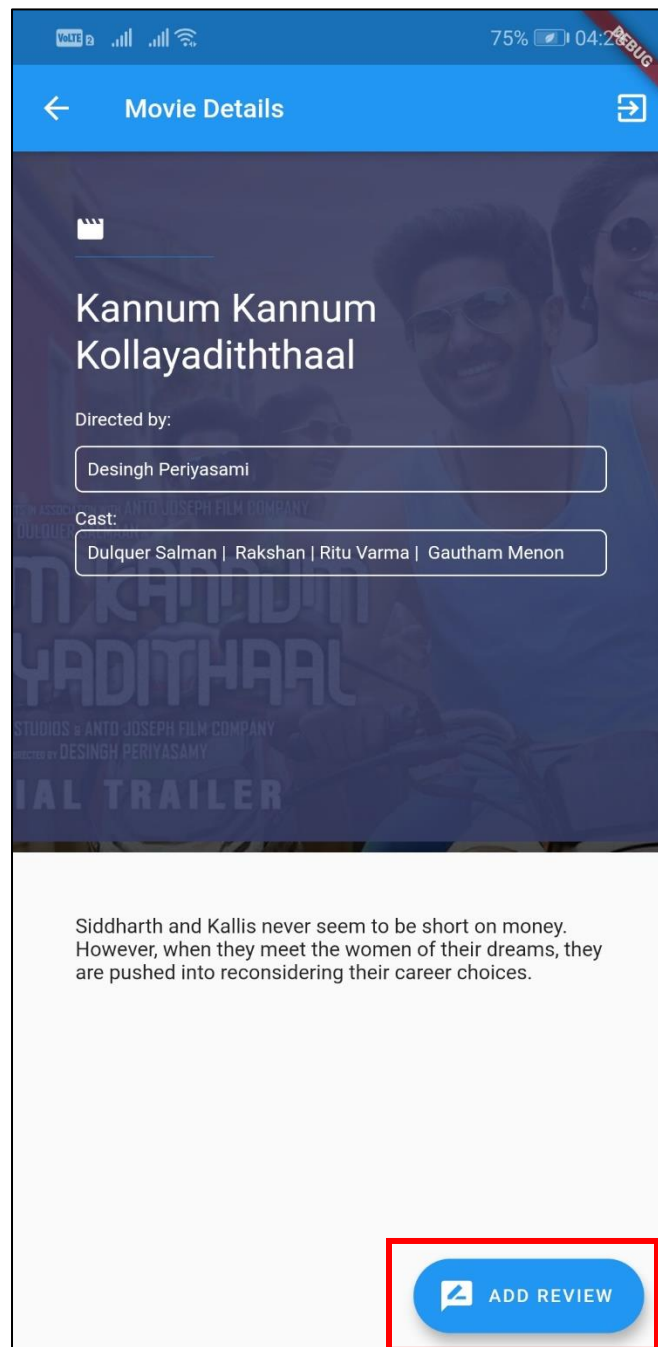


Figure 2. 2: Click on “ADD REVIEW” Button

2. Enter an appropriate comment about the movie & rate the movie by tap on stars

The screenshot displays a mobile application interface for adding a movie review. At the top, a blue header bar contains a back arrow, the text 'Add movie Review', and a share icon. Below the header, a red-bordered box highlights the 'Comment' section, which includes a text input field containing 'it's very adventurous story 🧠'. Below the comment field is the 'Rate Us' section, featuring five blue stars. At the bottom of the screen, there are two buttons: a blue 'SUBMIT REVIEW' button and a red 'CANCEL REVIEW' button. The status bar at the top shows 'VoLTE', signal strength, 75% battery, and the time 04:27. A red 'DEBUG' banner is visible in the top right corner.

Figure 2. 3: Enter Comment & Rating

3. Click on "SUBMIT" button

The screenshot shows a mobile application interface for adding a movie review. At the top, there is a blue header bar with a back arrow, the text "Add movie Review", and a share icon. Below the header, the screen is divided into two main sections. The first section, titled "Comment", contains a text input field with the text "it's very adventurous story 🌟". The second section, titled "Rate Us", displays five blue stars. Below the stars, there are two buttons: a blue "SUBMIT REVIEW" button, which is highlighted with a red rectangular box, and a red "CANCEL REVIEW" button. The status bar at the top of the phone shows "VoLTE", signal strength, Wi-Fi, 75% battery, and the time 04:20. A red "debug" banner is visible in the top right corner.

Figure 2. 4: Click on "SUBMIT REVIEW" Button

Then the app will display a toast message telling that the movie review added successfully & redirect to Movies page



Figure 2. 5: Display Toast Message

2.2 My Reviews Page

My Review page is fulfilling 3 tasks of the app. Through My Reviews page, the user will be able to view all the reviews he/she entered before, update previously entered reviews & delete reviews. User can view this page by clicking on the “My Reviews” tab at the bottom navigation bar

2.2.1 My Reviews Page User Interface

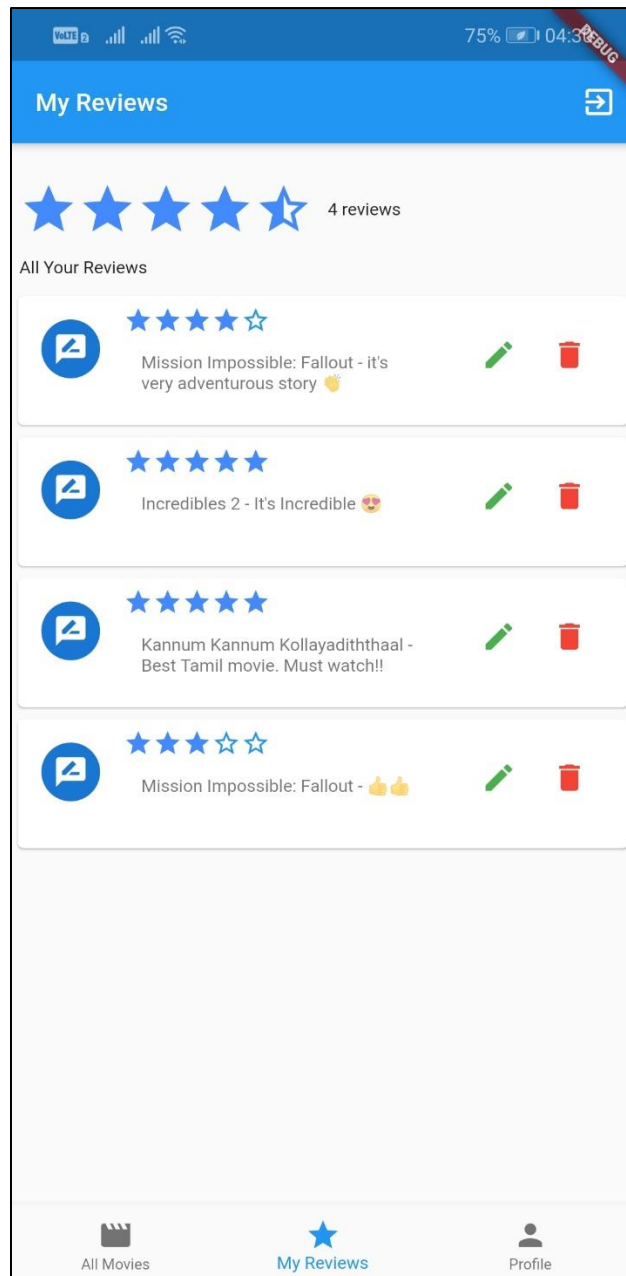


Figure 2. 6: My Reviews User Interface

2.2.2 View Reviews

As shows in above Figure, as soon as the user opens the My Reviews page, he/she can view all the reviews there. There are 2 sections in this page. The upper section displays the total number of review the user had given until the current time. The bottom section displays all the reviews which user had provided in separate review cards including the name of the reviewed movie, ratings & the review given. Also, in each card there are 2 icons: “Edit” icon & “Delete” icon which used to update & delete the reviews respectively.

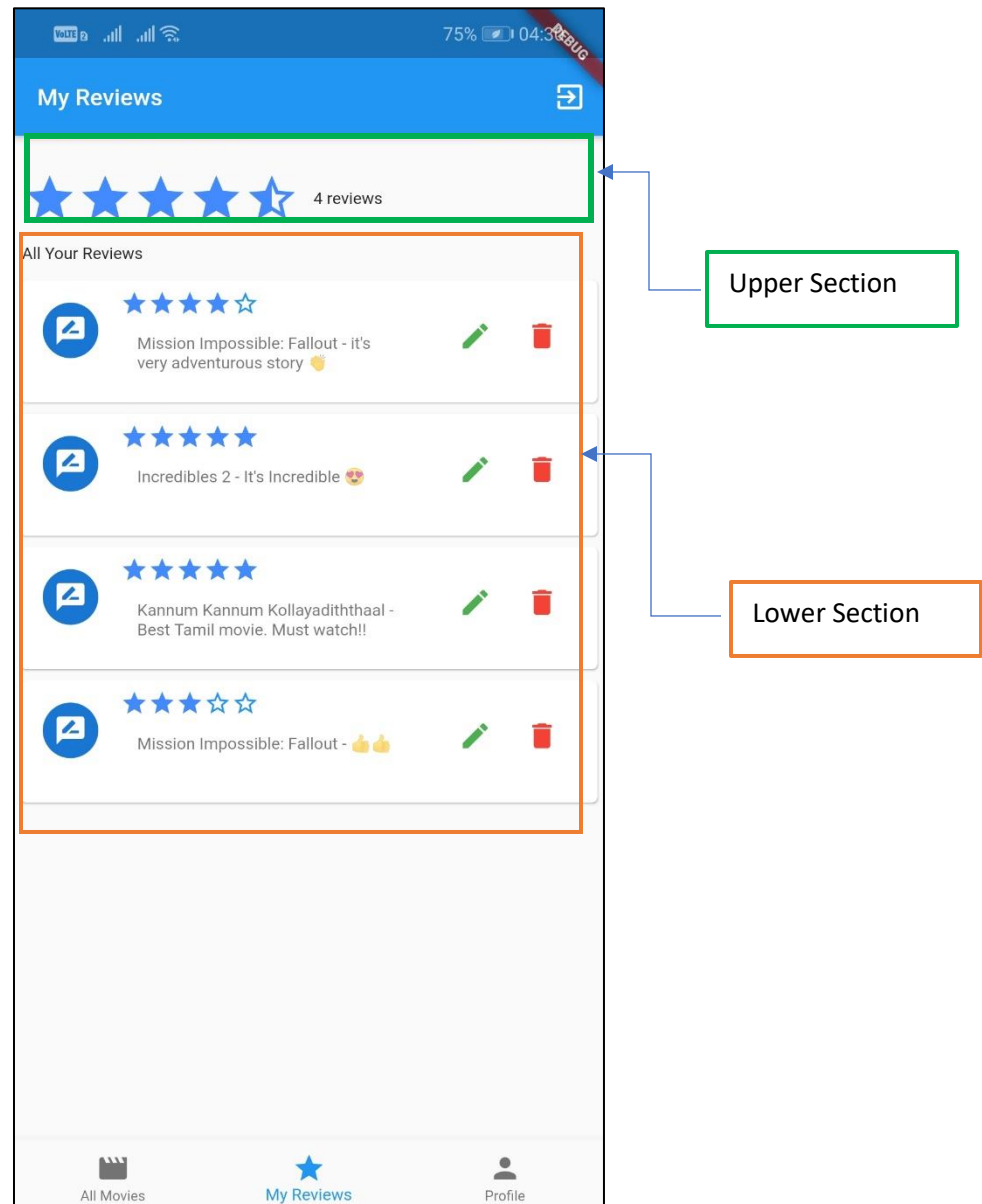


Figure 2. 7: Sections in My Reviews Page

2.2.2 Update a Selected Movie Review

Steps to update a review:

1. Click on “Edit” icon in a selected review card in My Reviews page

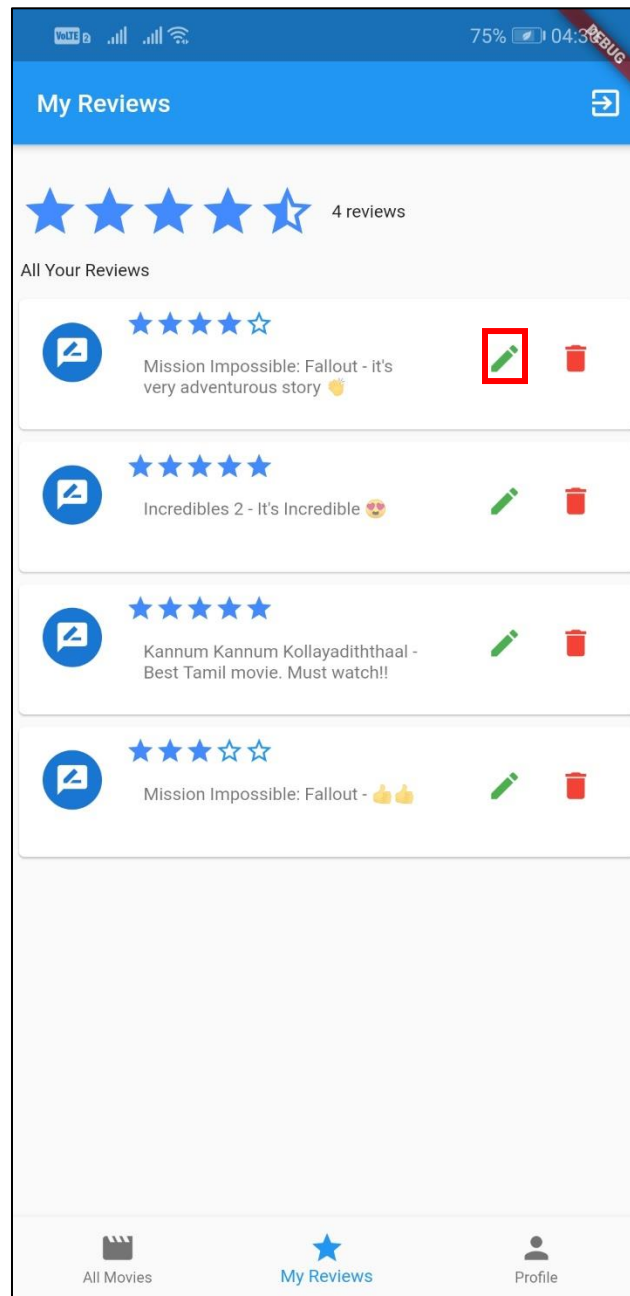


Figure 2. 8: Click on “Edit” Icon

2. Enter a new review inside the prompted dialog box

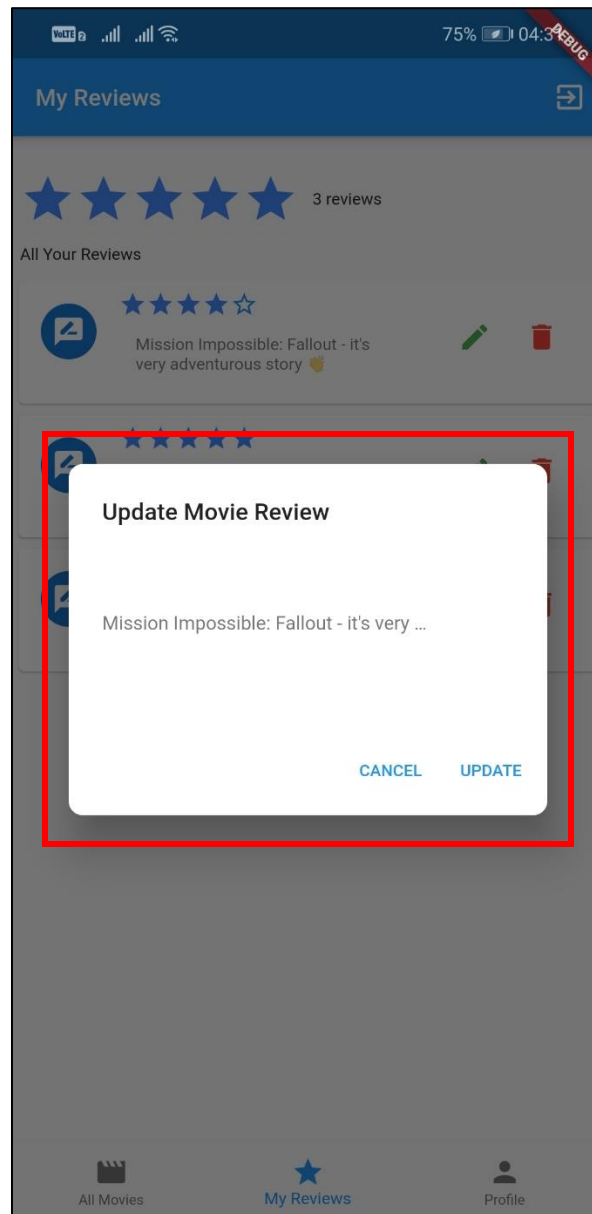


Figure 2. 9: View Dialog Box with Previously Entered Review

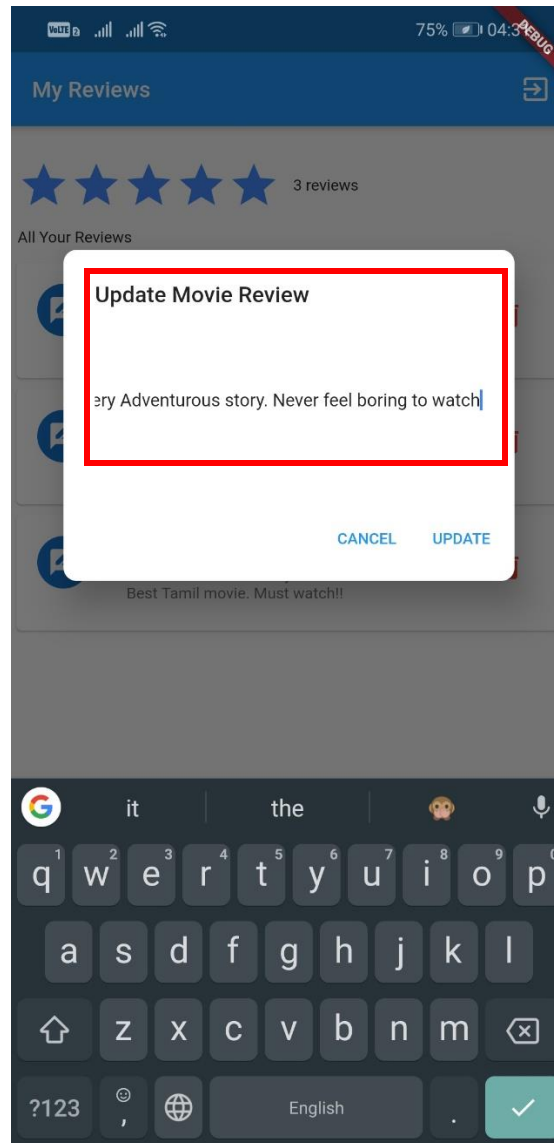


Figure 2. 10: Enter new Review in the Dialog Box

3. Click on “Update” button

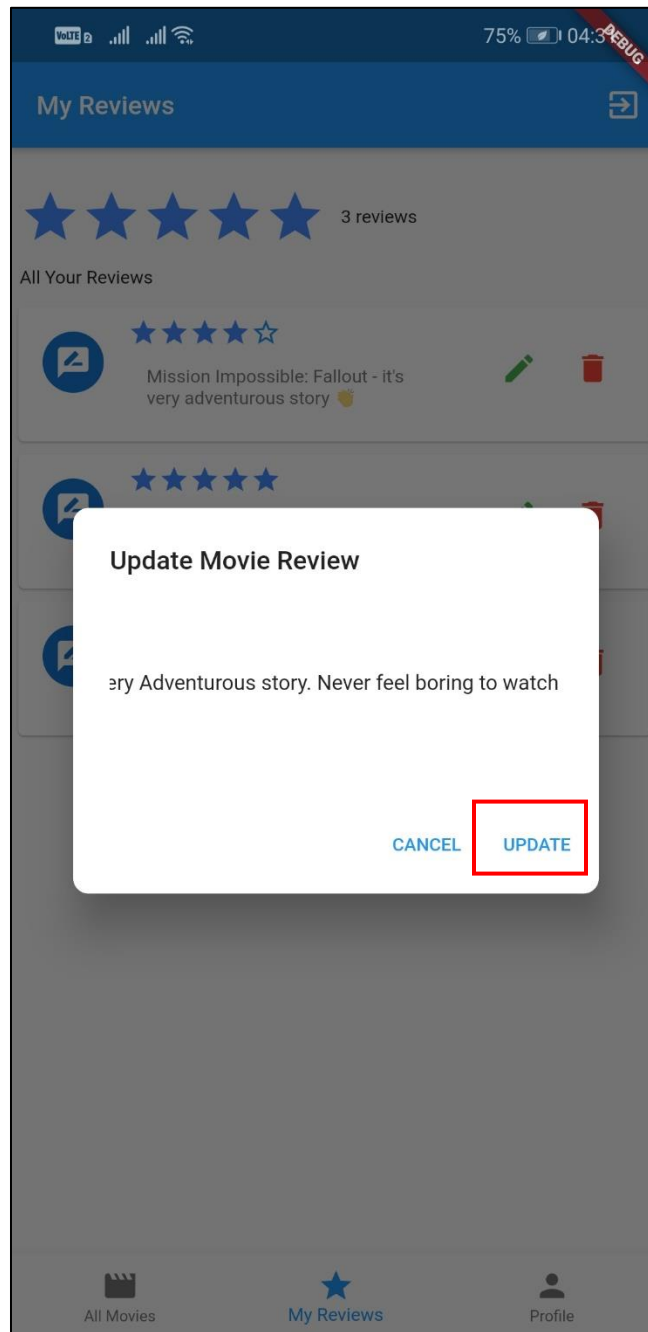


Figure 2. 11: Click on “UPDATE” Text

Then user will be redirected to the same page with a toast message telling that the review updated successfully. Also, the user can view the updated review on the same review card which he/she had selected before.

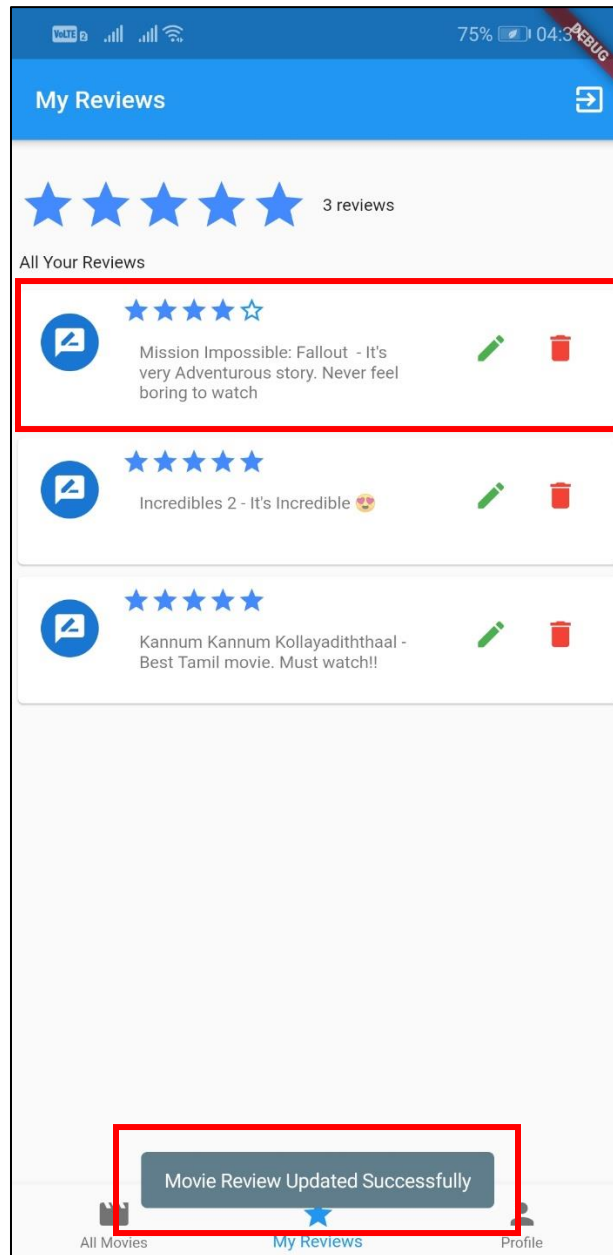


Figure 2. 12: Display the Appropriate Toast Message

2.2.3 Delete a Selected Movie Review

Steps to delete a movie review:

1. Click on “Delete” icon in a selected review card in My Reviews page

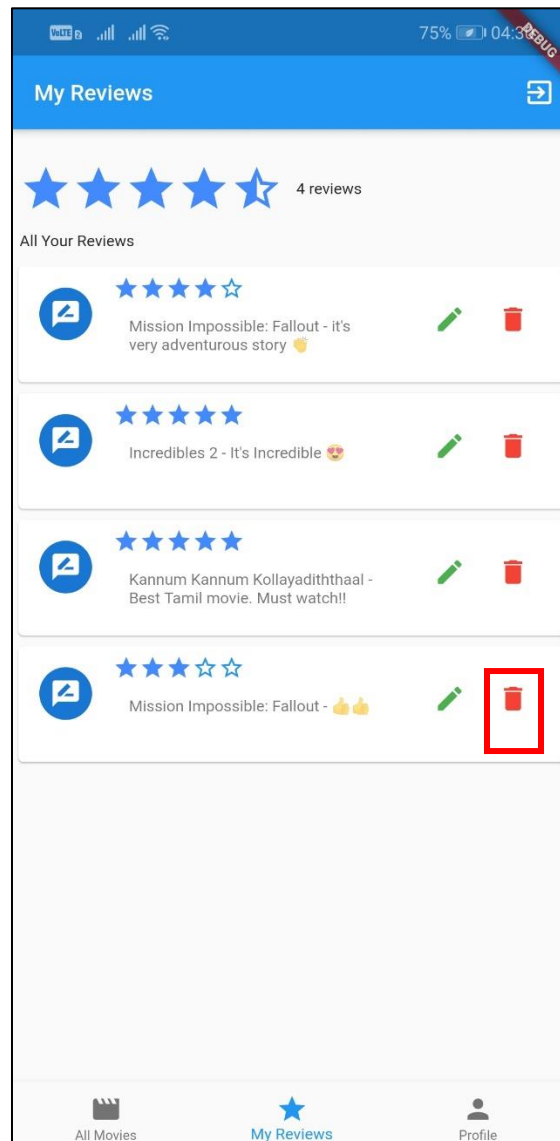


Figure 2. 13: Click on “Delete” Icon

2. The system will prompt an alert message asking to confirm the delete task. Click on “Delete” text in that message box

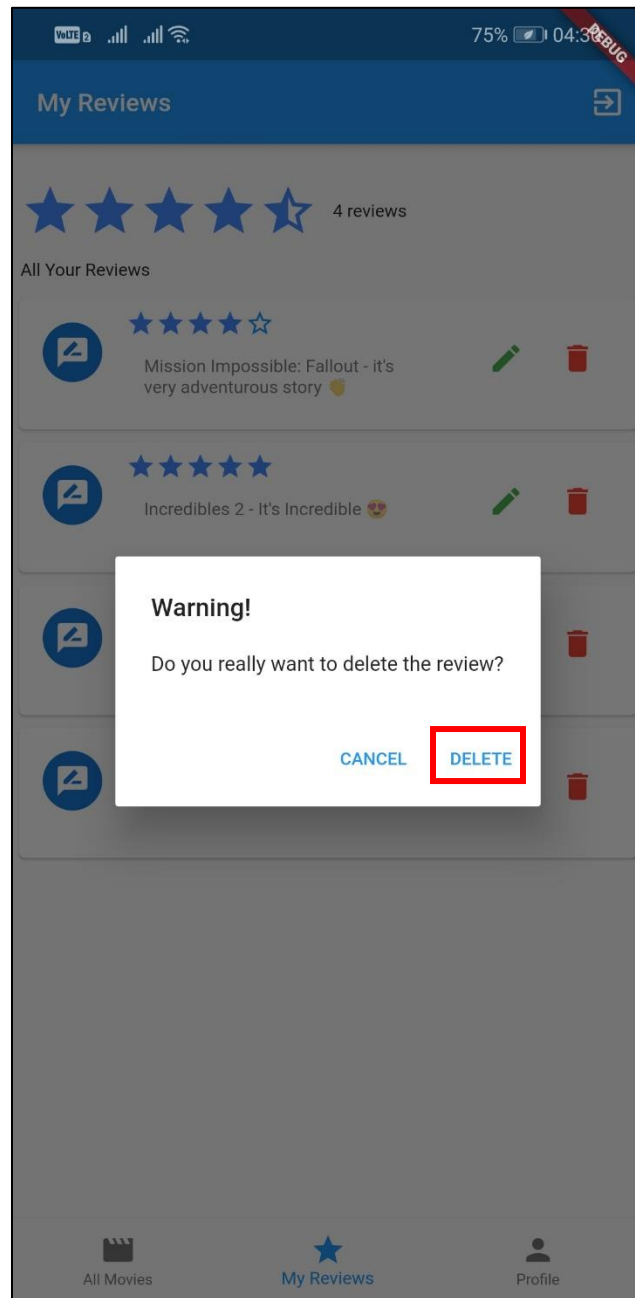


Figure 2. 14: Click on “DELETE” Text in Alert Box

Then the user will be redirected to the same My Reviews page while displaying a toast message telling that the review deleted successfully. Also, the user can see that the all the reviews except deleted review in My Reviews page

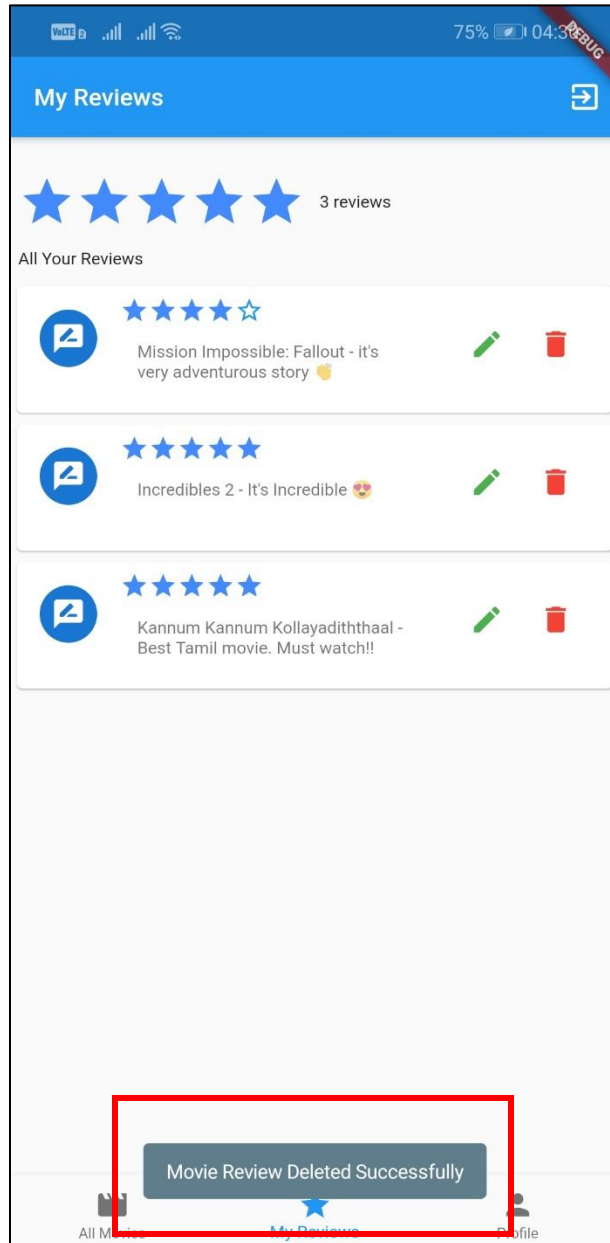


Figure 2. 15: View Appropriate Toast Message After Successful Delete Task

3.0 References

- [1] "Flutter Toast Message-How to Show Toast in Flutter App with Example," [Online]. Available: <https://protocoderspoint.com/flutter-toast-message-how-to-show-toast-in-flutter-app-with-example/>. [Accessed 03 April 2020].
- [2] "Flutter Documentation," [Online]. Available: <https://flutter.dev/docs>. [Accessed 20 March 2020].
- [3] BrandonOdiwuor, "Flutter-Restaurant-Review-App," [Online]. Available: <https://github.com/BrandonOdiwuor/Flutter-Restaurant-Review-App>. [Accessed 30 March 2020].
- [4] i. corner, "Flutter Tutorial - Firebase CRUD (Create Read Update Delete)".
- [5] R. Brunhage, "Flutter: Firebase CRUD Using Cloud Firestore".