iGamer

Project ID: 4

Report

P.P.G.S.H.A. Guruge - IT17042352

N.J. Pathiranage - IT17129404

B.Sc. Special (Hons.) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

May 2020

Table of Contents

List of Figures

1. Introduction

Project Title: Know a game

Project Description: As a Game reviewer app user, I need to be able to see a list of games that are available, and selecting one will show more details of the game.

Name of the app: iGamer

Functionalities of the Application are as follows:

1. View a list of game reviews in card view.
2. View the detailed description of a particular game when tapped on a card.
3. Add a new game review.
4. Update an existing game review.
5. Delete an existing game review.
6. About page, for users to learn about the app.
7. Help page, for users, to learn about the app basics and manage game reviews.

2. App Icon

The name “iGamer” is chosen for the app name, and orange is the theme color chosen for the iGamer app. The launcher icon is a picture of a game console.

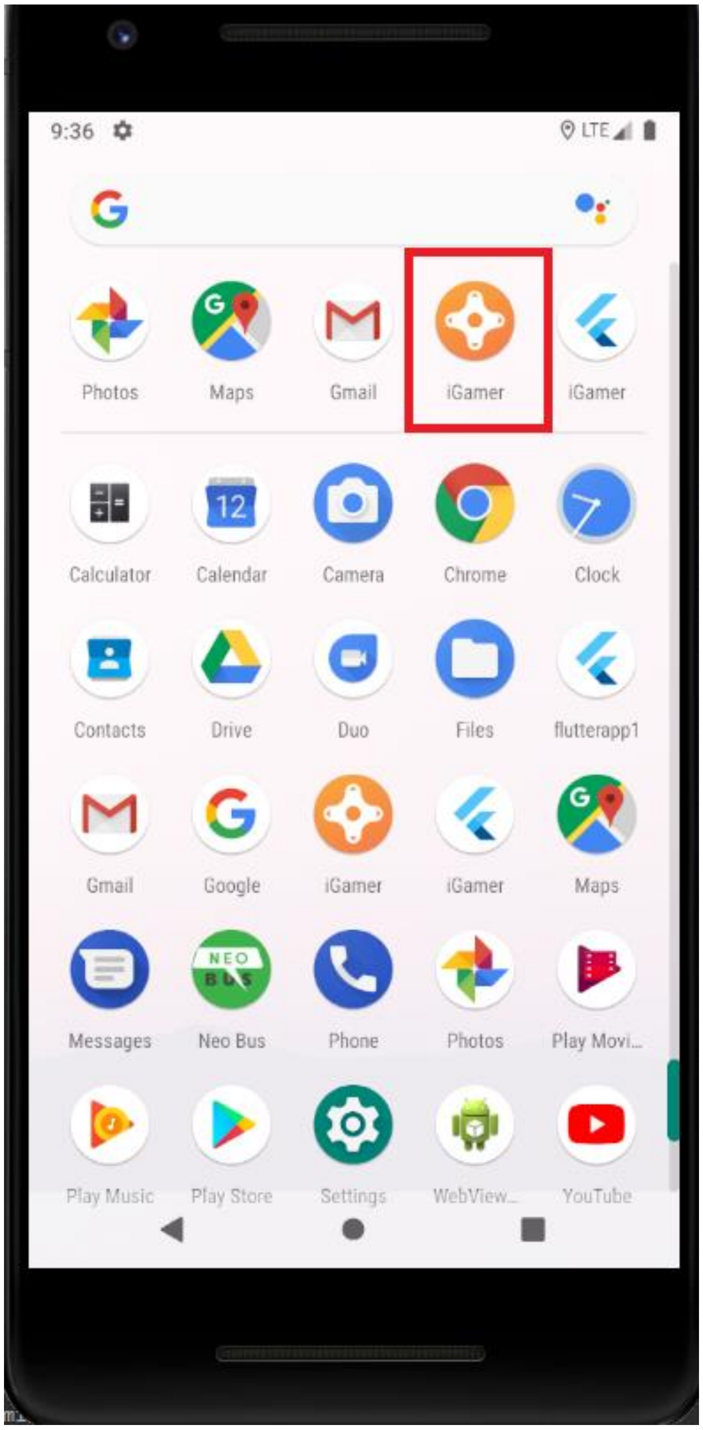


Figure 2.1: Game icon

AndroidManifest.xml

In the following AndroidManifest.xml file, the app name is set in **android:name** and app launcher icon is set in **android:icon** values.

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.igamer">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        android:name="io.flutter.app.FlutterApplication"
        android:label="iGamer"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale|la
youtDirection|fontScale|screenLayout|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

3. Splash Screen

The app consists of a Splash Screen showing the app icon and app name before starting the app.

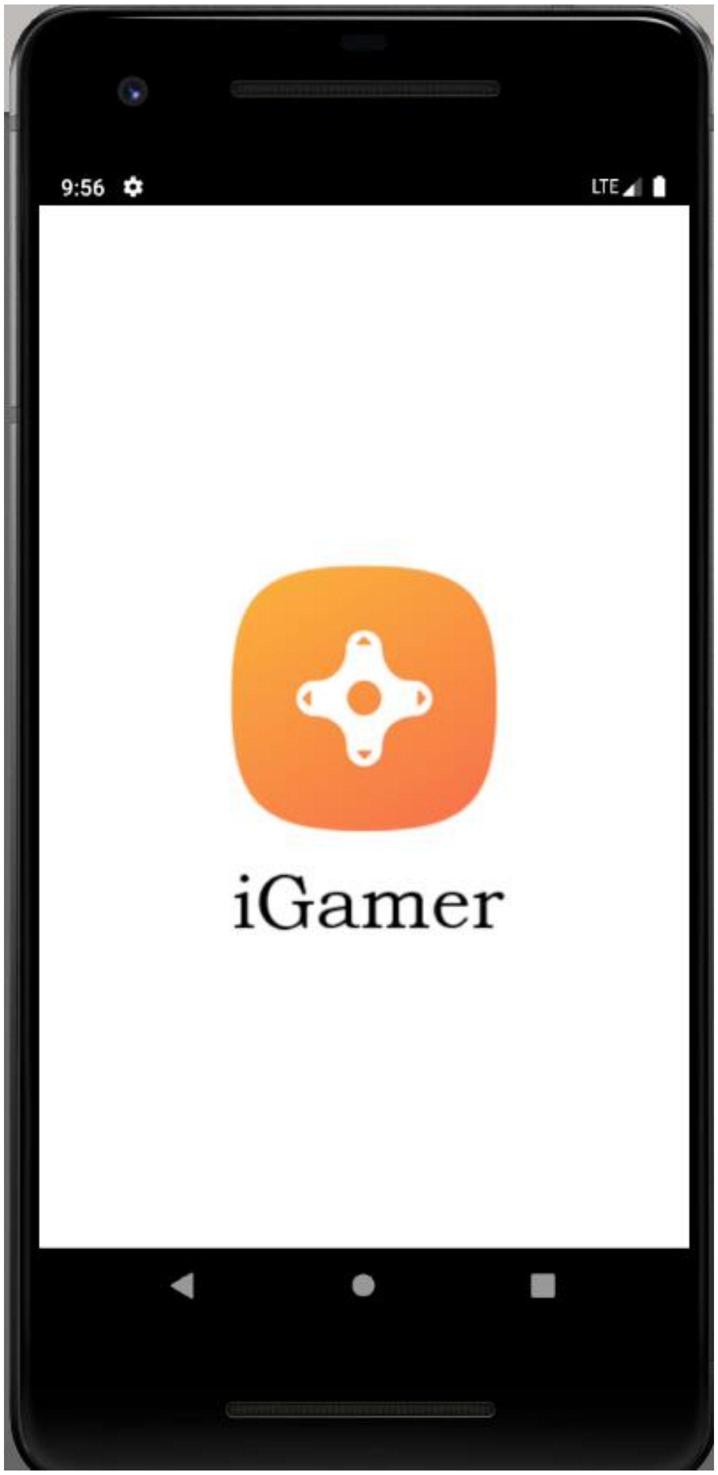


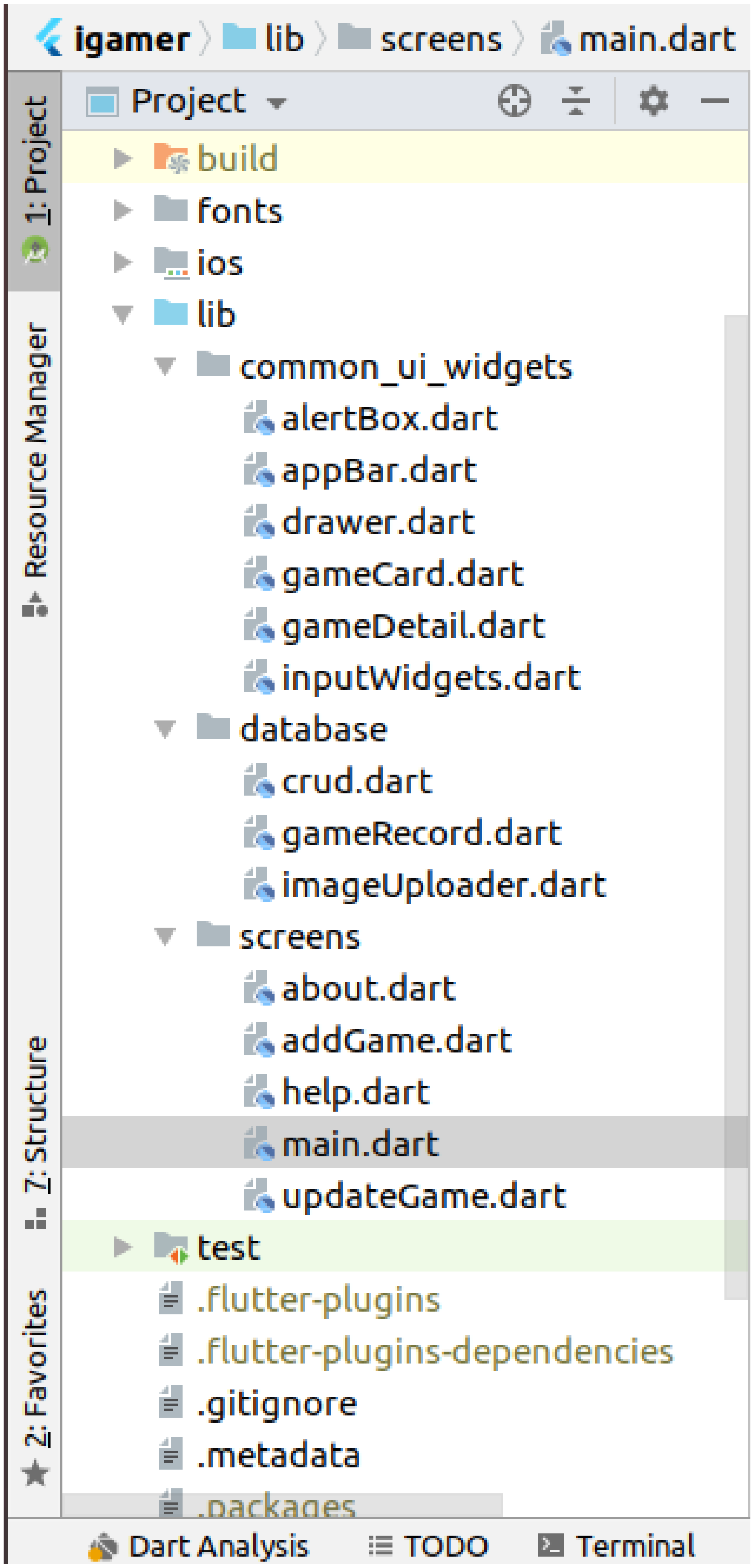
Figure 3.1: Splash screen

launch_background.xml

The splash screen is implemented as follows.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Modify this file to customize your launch splash screen -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@android:color/white" />
    <item android:bottom="12dp">
        <bitmap
            android:gravity="center"
            android:src="@drawable/icon"/>
    </item>
    <item android:top="250dp">
        <bitmap
            android:gravity="center"
            android:src="@drawable/imagetext"/>
    </item>
</layer-list>
```

4. Project Folder Structure



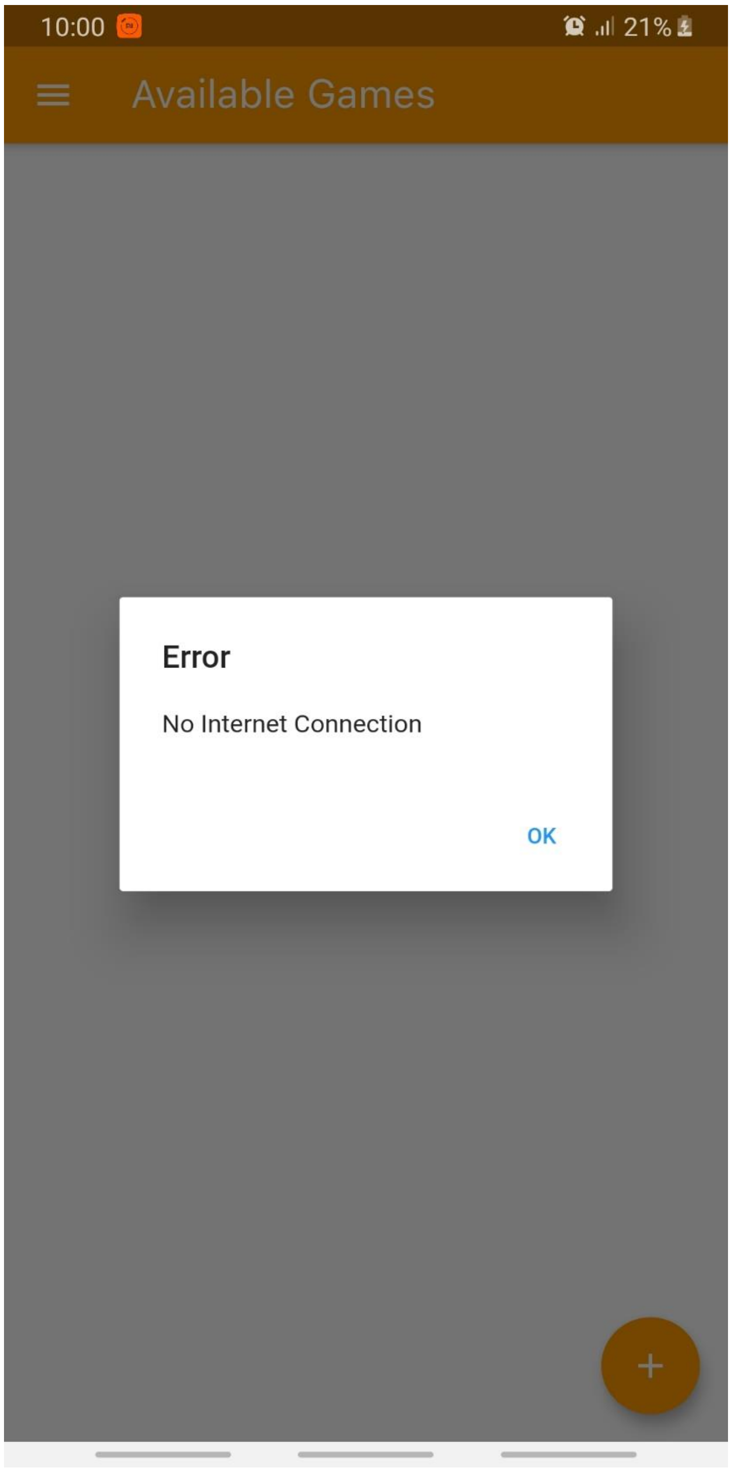
```
build
fonts
ios
lib
    common_ui_widgets
        alertBox.dart
        appBar.dart
        drawer.dart
        gameCard.dart
        gameDetail.dart
        inputWidgets.dart
    database
        crud.dart
        gameRecord.dart
        imageUploader.dart
    screens
        about.dart
        addGame.dart
        help.dart
        main.dart
        updateGame.dart
test
.flutter-plugins
.flutter-plugins-dependencies
.gitignore
.metadata
.packages
```

Figure 4.1: Folder Structure

5. Main page

5.1 Checking Internet Connection

iGamer flutter app runs if only internet connection is available. If the network connection is not available, it is notified to the user, as shown in the following image.



Error

No Internet Connection

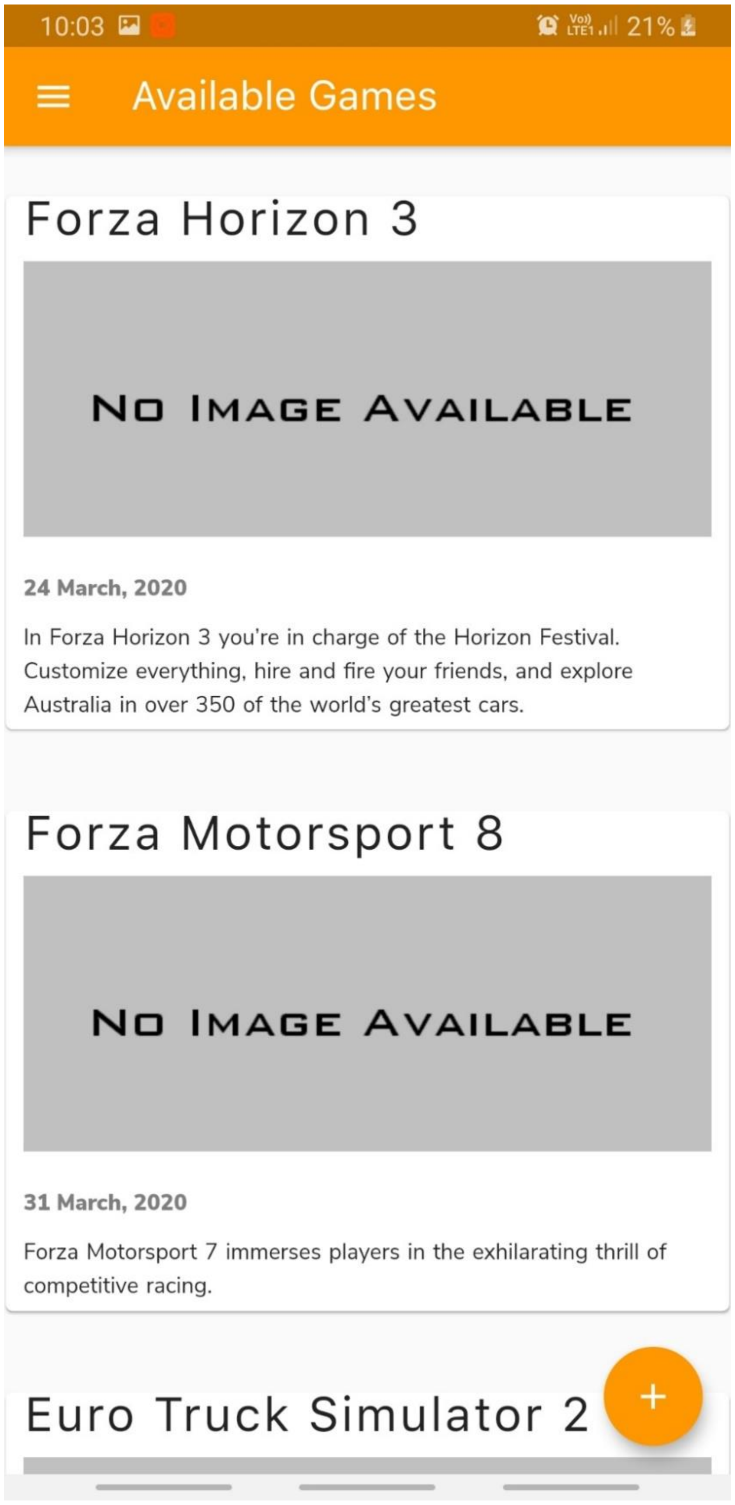
Figure 5.1: Checking internet connection

The functionality of checking the internet connection is implemented in the main.dart as follows.

```
// this function checks active internet connection
// if not, it will popup an Alert Box
_checkInternetConnection(BuildContext context) async {
  try {
    await InternetAddress.lookup('google.com');
  } on SocketException catch (_) {
    new AppAlertBox(context, "Error", "No Internet Connection", "OK")
        .showAlertDialog();
  }
}

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

5.2 Handling Issues with Loading Images

The image of each game which are stored in the Firebase store are network images. If the network connection is unavailable, those images are not loaded. To handle this issue, a no image available image is shown to the user.



Forza Horizon 3

24 March, 2020

In Forza Horizon 3 you're in charge of the Horizon Festival. Customize everything, hire and fire your friends, and explore Australia in over 350 of the world's greatest cars.

Forza Motorsport 8

31 March, 2020

Forza Motorsport 7 immerses players in the exhilarating thrill of competitive racing.

Euro Truck Simulator 2

Figure 5.2: Show no image available

5.3 Handling Issues with Loading Data

With the availability of the network connection, the user can successfully view the home page of iGamer app. Initially, a circular progress indicator is shown when establishing a connection with the database.

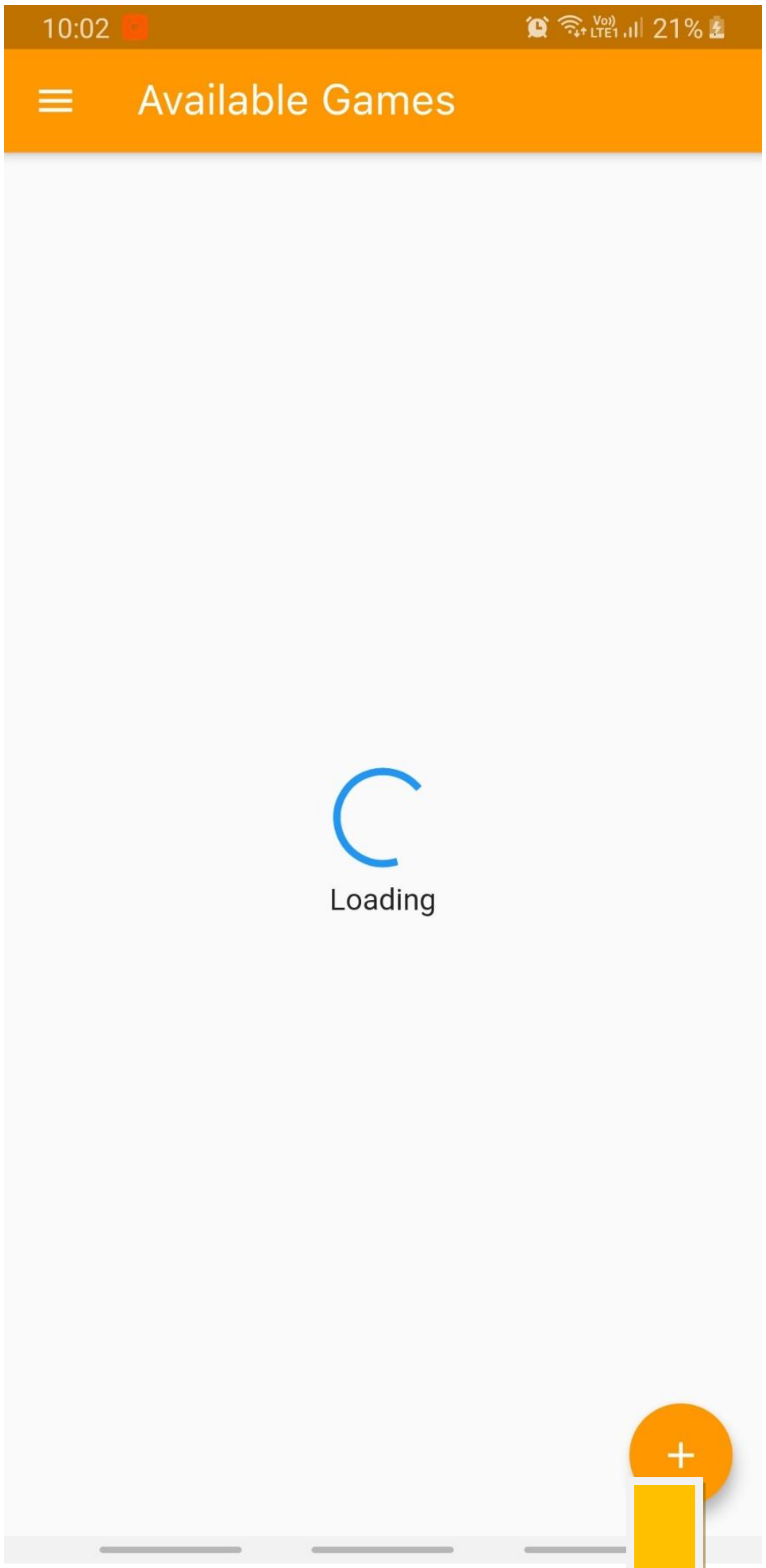
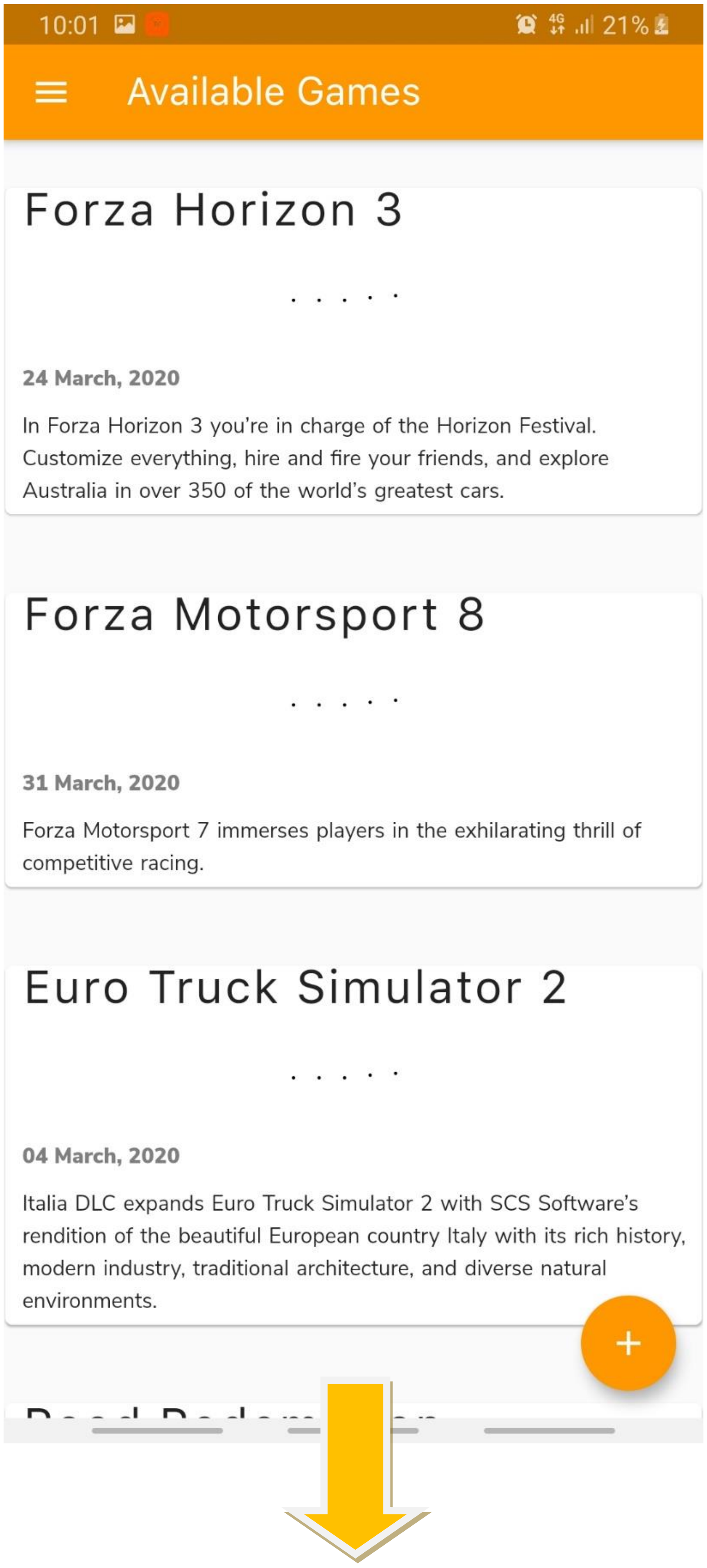


Figure 5.3: Loading screen

During the process of loading images “JumpingDotsProgressIndicator” is used.



Forza Horizon 3

24 March, 2020

In Forza Horizon 3 you’re in charge of the Horizon Festival. Customize everything, hire and fire your friends, and explore Australia in over 350 of the world’s greatest cars.

Forza Motorsport 8

31 March, 2020

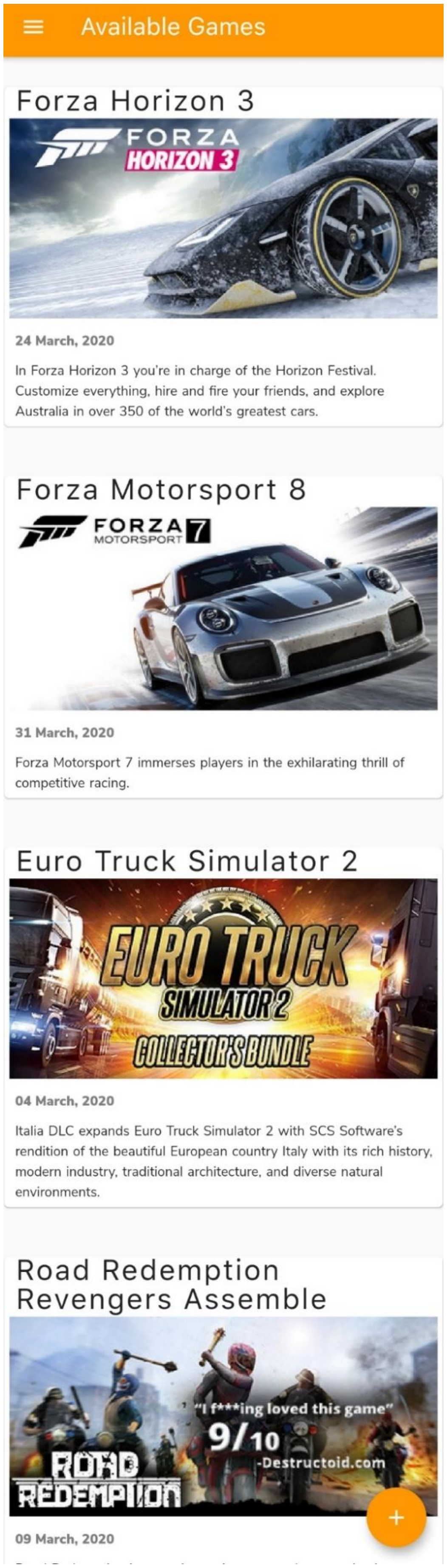
Forza Motorsport 7 immerses players in the exhilarating thrill of competitive racing.

Euro Truck Simulator 2

04 March, 2020

Italia DLC expands Euro Truck Simulator 2 with SCS Software’s rendition of the beautiful European country Italy with its rich history, modern industry, traditional architecture, and diverse natural environments.

Figure 5.4: Show JumpingDotsProgressIndicator


≡ Available Games
FORZA HORIZON 3
FORZA 7 MOTORSPORT
EURO TRUCK SIMULATOR 2 COLLECTOR'S BUNDLE
ROAD REDEMPTION
"I f***ing loved this game" 9/10 -Destructoid.com
+

Forza Horizon 3

24 March, 2020

In Forza Horizon 3 you're in charge of the Horizon Festival. Customize everything, hire and fire your friends, and explore Australia in over 350 of the world's greatest cars.

Forza Motorsport 8

31 March, 2020

Forza Motorsport 7 immerses players in the exhilarating thrill of competitive racing.

Euro Truck Simulator 2

04 March, 2020

Italia DLC expands Euro Truck Simulator 2 with SCS Software's rendition of the beautiful European country Italy with its rich history, modern industry, traditional architecture, and diverse natural environments.

Road Redemption Revengers Assemble

09 March, 2020

Figure 5.5: List of games

The following code explains how list of games are rendered including how loading circular progress indicator is handled. The implementation of no image error and jumping dots indicator is explained in section 7.1.

main.dart

```
// In the file main screen is implemented
import 'dart:async';
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:igamer/common_ui_widgets/drawer.dart';
import 'package:igamer/common_ui_widgets/inputWidgets.dart';
import 'package:igamer/database/crud.dart';
import 'package:igamer/database/gameRecord.dart';
import 'package:igamer/database/imageUploader.dart';
import '../common_ui_widgets/appBar.dart';
import 'dart:io';
import 'package:image_picker/image_picker.dart';
import 'main.dart';

// Name of the page
final title = 'Add Game Review';

// Main method
void main() => runApp(AddGame());

class AddGame extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: title,
      home: Scaffold(
        appBar: new CustomizedAppBar(title).getAppBar(),
        //getting custom built app bar
        body: AddGameForm(title: title),
        drawer: new CustomizedDrawer(context)
            .getDrawer(), //getting custom built app drawer
      ),
    );
  }
}

// Create a Form widget.
class AddGameForm extends StatefulWidget {
  final String title;

  //Constructor
  AddGameForm({Key key, this.title}) : super(key: key);

  @override
  AddGameFormState createState() {
    return AddGameFormState();
  }
}

class AddGameFormState extends State<AddGameForm> {
```

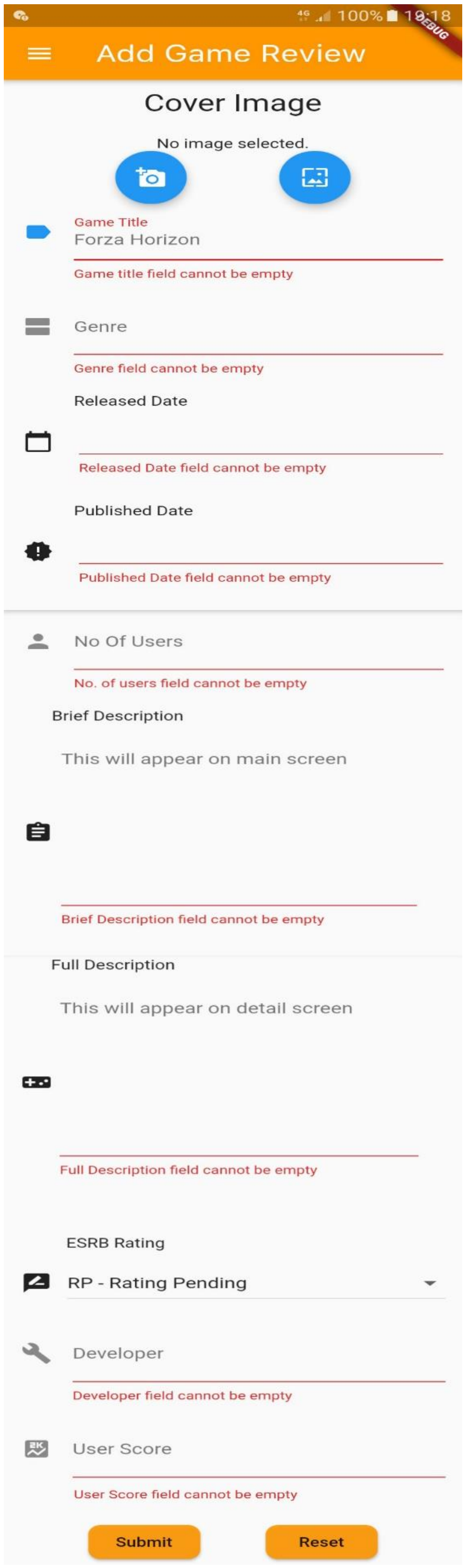
```
final _formKey = GlobalKey<FormState>();
FocusNode focusNode;
List<DropdownMenuItem<String>> _dropDownMenuItems;
String _selectedESRBRating;
File _image;
bool _greyOutBackground = false;

//Initializing text editing controllers
TextEditingController _titleController = new TextEditingController();
TextEditingController _genreController = new TextEditingController();
TextEditingController _relDateController = new TextEditingController();
TextEditingController _pubDateController = new TextEditingController();
TextEditingController _noOfUsersController = new TextEditingController();
TextEditingController _briefDescController = new TextEditingController();
TextEditingController _fullDescController = new TextEditingController();
TextEditingController _developerController = new TextEditingController();
TextEditingController _userScoreController = new TextEditingController();

//Initializing an object from CommonInputWidgets class
CommonInputWidgets _commonInputWidgets = new CommonInputWidgets();

//Initializing drop down values for ESRB Ratings
List _ratings = [
  "RP - Rating Pending",
  "EC - Early Childhood",
  "E - Everyone",
  "E10+ - Everyone 10+",
  "T - Teen",
  "M - Mature",
  "AO - Adults Only"
];

// initializing local variables at the beginning of the screen
@override
void initState() {
  super.initState();
  focusNode = FocusNode();
  _dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
  _selectedESRBRating = _dropDownMenuItems[0].value;
  _image = null;
  _greyOutBackground = false;
}

@override
Widget build(BuildContext context) {
  return new Scaffold(
    backgroundColor:
        _greyOutBackground == true ? Colors.grey : Colors.transparent,
    body: new GestureDetector(
      onTap: (){
        FocusScope.of(context).requestFocus(new FocusNode());
      },
      child: Stack(
        children: <Widget>[
          if (_greyOutBackground) _getCircularProgressIndicator(),
          //display an empty form
          Form(
            key: _formKey,
            child: SingleChildScrollView(
              padding: const EdgeInsets.only(left: 15, right: 15, top: 10),
              child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      _getImagePicker(),
                      //Title field
                      _commonInputWidgets.getTextField(
                          "Game Title",
                          "Forza Horizon",
                          Icons.label,
                          _titleController,
                          "Game title field cannot be empty",
                          focusNode),
                      //Genre field
                      _commonInputWidgets.getTextField(
                          "Genre",
                          "Racing, Simulation, Automobile",
                          Icons.view_agenda,
                          _genreController,
                          "Genre field cannot be empty",
                          focusNode),
                      //Released Date field
                      _commonInputWidgets.getDatePicker(
                          "Released Date",
                          Icons.calendar_today,
                          _relDateController,
                          "Released Date field cannot be empty",
                          focusNode),
                      //Published Date field
                      _commonInputWidgets.getDatePicker(
                          "Published Date",
                          Icons.new_releases,
                          _pubDateController,
                          "Published Date field cannot be empty",
                          focusNode),
                      //No. of users field
                      _commonInputWidgets.getNumberTextField(
                          "No Of Users",
                          "2",
                          Icons.person,
                          true,
                          _noOfUsersController,
                          "No. of users field cannot be empty",
                          focusNode),
                      //Brief Description field
                      _commonInputWidgets.getTextArea(
                          "Brief Description",
                          "This will appear on main screen",
                          Icons.assignment,
                          _briefDescController,
                          "Brief Description field cannot be empty",
                          focusNode),
                      //Full Description field
                      _commonInputWidgets.getTextArea(
                          "Full Description",
                          "This will appear on detail screen",
                          Icons.videogame_asset,
                          _fullDescController,
                          "Full Description field cannot be empty",
                          focusNode),
                      //ESRB Rating dropdown field
                      _getDropDown("ESRB Rating", Icons.rate_review),
                      //Developer field
```

```
_commonInputWidgets.getTextField(
    "Developer",
    "Playground Games",
    Icons.build,
    _developerController,
    "Developer field cannot be empty",
    focusNode),
//User score field
_commonInputWidgets.getNumberTextField(
    "User Score",
    "7.8",
    Icons.score,
    false,
    _userScoreController,
    "User Score field cannot be empty",
    focusNode),
new Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    new RaisedButton(
        padding: const EdgeInsets.all(10.0),
        onPressed: () async {
          setState(() {
            _greyOutBackground = true;
          });
          //if the form fields are validated
          if (_formKey.currentState.validate()) {
            Scaffold.of(context).showSnackBar(SnackBar(
              content: Text('Adding New Game Review'),
            ));
            //uploads the image
            ImageUploader uploader = new ImageUploader(
                _titleController.text +
                    "-" +
                    _generateID().toString(),
                _image);
            var imageURL = await uploader.uploadFile();
            GameRecord game = new GameRecord(
                _generateID(),
                _titleController.text,
                _pubDateController.text,
                _briefDescController.text,
                imageURL,
                _genreController.text,
                _developerController.text,
                _relDateController.text,
                _fullDescController.text,
                _selectedESRBRating,
                _userScoreController.text,
                _noOfUsersController.text,
                null);
            await CRUD().addGame(game);
            setState(() {
              _greyOutBackground = false;
            });
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => MyHomePage()));
          }
        },
                                //Submit button
                                child: new Text('Submit'),
                                color: Colors.orange.withOpacity(0.9),
                                shape: RoundedRectangleBorder(
                                    borderRadius: new BorderRadius.circular(10.0))),
                            //reset button
                            new RaisedButton(
                                padding: const EdgeInsets.all(10.0),
                                onPressed: () {
                                  _formKey.currentState.reset();
                                  _clearImage();
                                  _titleController.clear();
                                  _genreController.clear();
                                  _relDateController.clear();
                                  _pubDateController.clear();
                                  _noOfUsersController.clear();
                                  _briefDescController.clear();
                                  _fullDescController.clear();
                                  _getlist();
                                  _developerController.clear();
                                  _userScoreController.clear();
                                },
                                child: new Text('Reset'),
                                color: Colors.orange.withOpacity(0.9),
                                shape: RoundedRectangleBorder(
                                    borderRadius: new BorderRadius.circular(10.0)))
                          ],
                        ),
                      ],
                    ),
                  ),
                )
              ],
            ),
          )
        );
  }

  // this function returns a random integer between 0 and 10000
  int _generateID() {
    var random = Random();
    return random.nextInt(10000);
  }

  // this function brings the drop down list to its initial state
  Future _getlist() async {
    return setState(() {
      _dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
      _selectedESRBRating = _dropDownMenuItems[0].value;
    });
  }

  // this function gets an image from the camera and set to _image
  Future _getImageFromCamera() async {
    return ImagePicker.pickImage(source: ImageSource.camera).then((file) {
      setState(() {
        _image = file;
      });
    });
  }
  // this function picks an image from the gallery and set to _image
  Future _getImageFromGallery() async {
    return ImagePicker.pickImage(source: ImageSource.gallery).then((file) {
      setState(() {
        _image = file;
      });
    });
  }

  // the function removes the selected image from the gallery
  Future _clearImage() async {
    setState(() {
      _image = null;
    });
  }

  // this function returns a Column having an Image Picker
  Column _getImagePicker() {
    return (new Column(
      children: <Widget>[
        Container(
          margin: const EdgeInsets.only(bottom: 20),
          child: Text(
            'Cover Image',
            style: TextStyle(fontSize: 25),
          ),
        ),
        _image == null ? new Text('No image selected.') : Image.file(_image, height: 187, width:
400, fit: BoxFit.fitWidth,),
        // if no image is selected show Text else show the image
        _image == null // if no image is selected, show Choose Image button
            ? new Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            FloatingActionButton(
              heroTag: 'btn_camera',
              onPressed: _getImageFromCamera,
              tooltip: 'Pick Image',
              child: Icon(Icons.add_a_photo),
            ),
            FloatingActionButton(
              heroTag: 'btn_gallery',
              onPressed: _getImageFromGallery,
              tooltip: 'Pick Image',
              child: Icon(Icons.wallpaper),),],)
            : Container(),
        _image != null // if image is selected, show Remove Button
            ? new Container(
                margin: const EdgeInsets.only(top: 10, bottom: 20),
                child: new RaisedButton(
                    child: Container(
                      width: 85,
                      height: 40,
                      child: Row(
                        children: <Widget>[Icon(Icons.delete), Text('Remove')],
                        mainAxisAlignment: MainAxisAlignment.center,
                      ),
                    ),
                    onPressed: () {
                      setState(() {
                        _clearImage();
```

```
                  });
                },
                color: Colors.red.withOpacity(0.80),
                elevation: 0,
                shape: RoundedRectangleBorder(
                    borderRadius: new BorderRadius.circular(10.0))),
          )
        : Container()
    ],
  ));
}

// this function adds items in the ratings list to the Drop down menu item
List<DropdownMenuItem<String>> _buildAndGetDropDownMenuItems(List ratings) {
  List<DropdownMenuItem<String>> items = List();
  for (String rating in ratings) {
    items.add(DropdownMenuItem(value: rating, child: Text(rating)));
  }
  return items;
}

// this function returns a dropdown list
Container _getDropDown(String label, IconData icon) {
  return (Container(
    margin: const EdgeInsets.only(bottom: 20),
    child: Column(
      children: <Widget>[
        Container(
          alignment: Alignment(-1, -1),
          margin: const EdgeInsets.only(top: 30, bottom: 10, left: 35),
          child: Text(label),
        ),
        Row(
          children: <Widget>[
            Container(
              child: Icon(icon),
            ),
            Container(
              width: 330,
              margin: const EdgeInsets.only(left: 12),
              child: DropdownButton(
                value: _selectedESRBRating,
                items: _dropDownMenuItems,
                onChanged: (pickedValue) {
                  setState(() {
                    _selectedESRBRating = pickedValue;
                  });
                },
                isExpanded: true,
              ),
            )
          ],
        )
      ],
    ),
  ));
}

//function which returns Circular Progress Indicator
Center _getCircularProgressIndicator() {
  return Center(
        child: new Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Container(
          height: 50,
          width: 50,
          child: CircularProgressIndicator(),
        ),
        Container(
          margin: const EdgeInsets.only(top: 10),
          child: Text(
            "Please wait",
            style: TextStyle(fontSize: 18),
          ),
        )
      ],
    ));
  }
}
```

6. Add Game

The game reviewers can add a new game to the app.



Cover Image

No image selected.

Game Title
Forza Horizon
Game title field cannot be empty

Genre
Genre field cannot be empty

Released Date
Released Date field cannot be empty

Published Date
Published Date field cannot be empty

No Of Users
No. of users field cannot be empty

Brief Description
This will appear on main screen
Brief Description field cannot be empty

Full Description
This will appear on detail screen
Full Description field cannot be empty

ESRB Rating
RP - Rating Pending

Developer
Developer field cannot be empty

User Score
User Score field cannot be empty

Figure 6.1: Add game screen

An image of the game can be uploaded, either selecting from the gallery, or via capturing through the phone camera. The app has linked with the phone's camera. Also, all the fields are validated in order for the users, and validation is displayed, as shown in the above figure.

After a new game is added successfully, the user is notified with two confirmation messages, a loading screen with greyed out background, and a snack bar at the bottom of the screen. Once the game is successfully added, the user is navigated back to the home screen.

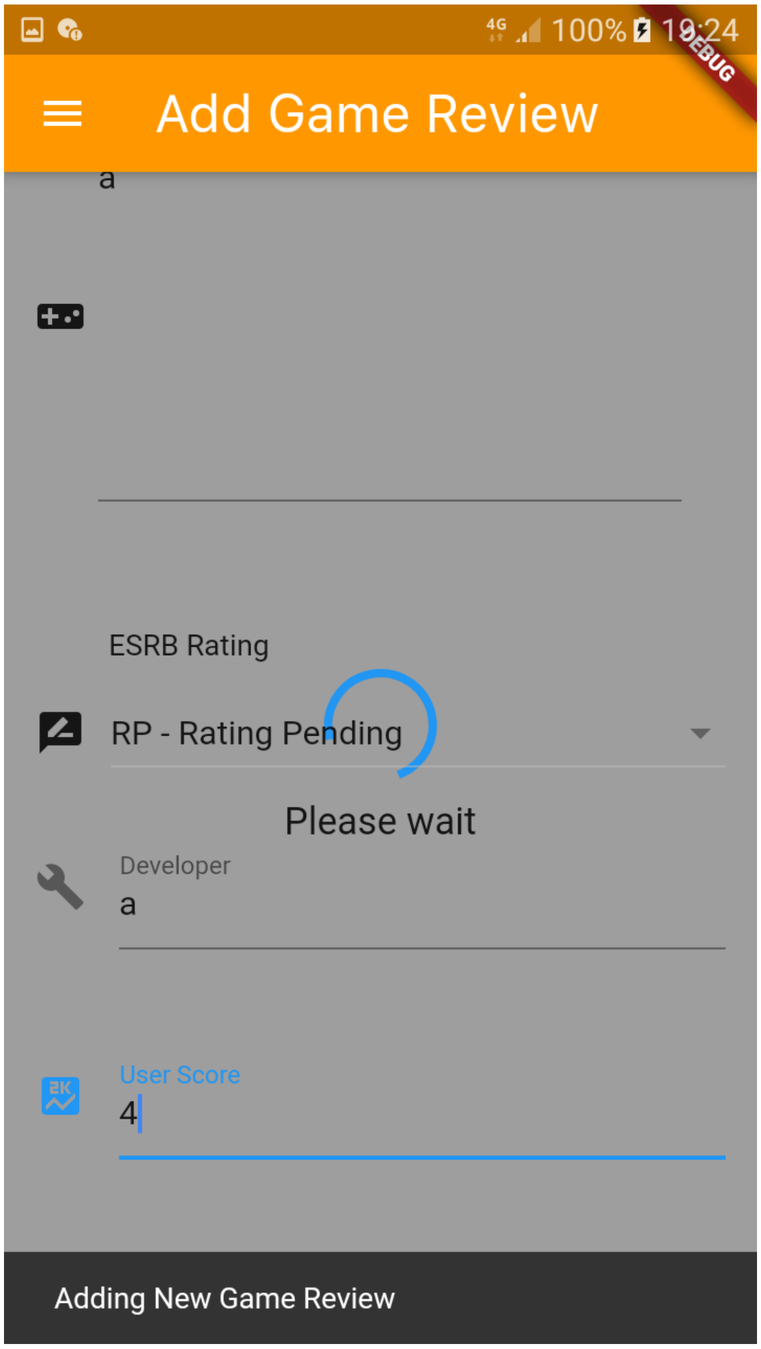


Figure 6.2: Adding a new game

The following code explains how the above functionality is implemented.

addGame.dart

```
// In the file add game screen is implemented
import 'dart:async';
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:igamer/common_ui_widgets/drawer.dart';
import 'package:igamer/common_ui_widgets/inputWidgets.dart';
import 'package:igamer/database/crud.dart';
import 'package:igamer/database/gameRecord.dart';
import 'package:igamer/database/imageUploader.dart';
import '../common_ui_widgets/appBar.dart';
import 'dart:io';
import 'package:image_picker/image_picker.dart';
import 'main.dart';

// Name of the page
final title = 'Add Game Review';

// Main method
void main() => runApp(AddGame());

class AddGame extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: title,
      home: Scaffold(
        appBar: new CustomizedAppBar(title).getAppBar(),
        //getting custom built app bar
        body: AddGameForm(title: title),
        drawer: new CustomizedDrawer(context)
            .getDrawer(), //getting custom built app drawer
      ),
    );
  }
}

// Create a Form widget.
class AddGameForm extends StatefulWidget {
  final String title;

  //Constructor
  AddGameForm({Key key, this.title}) : super(key: key);

  @override
  AddGameFormState createState() {
    return AddGameFormState();
  }
}

class AddGameFormState extends State<AddGameForm> {
  final _formKey = GlobalKey<FormState>();
  FocusNode focusNode;
  List<DropdownMenuItem<String>> _dropDownMenuItems;
```

```
String _selectedESRBRating;
File _image;
bool _greyOutBackground = false;

//Initializing text editing controllers
TextEditingController _titleController = new TextEditingController();
TextEditingController _genreController = new TextEditingController();
TextEditingController _relDateController = new TextEditingController();
TextEditingController _pubDateController = new TextEditingController();
TextEditingController _noOfUsersController = new TextEditingController();
TextEditingController _briefDescController = new TextEditingController();
TextEditingController _fullDescController = new TextEditingController();
TextEditingController _developerController = new TextEditingController();
TextEditingController _userScoreController = new TextEditingController();

//Initializing an object from CommonInputWidgets class
CommonInputWidgets _commonInputWidgets = new CommonInputWidgets();

//Initializing drop down values for ESRB Ratings
List _ratings = [
  "RP - Rating Pending",
  "EC - Early Childhood",
  "E - Everyone",
  "E10+ - Everyone 10+",
  "T - Teen",
  "M - Mature",
  "AO - Adults Only"
];

// initializing local variables at the beginning of the screen
@override
void initState() {
  super.initState();
  focusNode = FocusNode();
  _dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
  _selectedESRBRating = _dropDownMenuItems[0].value;
  _image = null;
  _greyOutBackground = false;
}

@override
Widget build(BuildContext context) {
  return new Scaffold(
    backgroundColor:
        _greyOutBackground == true ? Colors.grey : Colors.transparent,
    body: new GestureDetector(
      onTap: (){
        FocusScope.of(context).requestFocus(new FocusNode());
      },
      child: Stack(
        children: <Widget>[
          if (_greyOutBackground) _getCircularProgressIndicator(),
          //display an empty form
          Form(
            key: _formKey,
            child: SingleChildScrollView(
              padding: const EdgeInsets.only(left: 15, right: 15, top: 10),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  _getImagePicker(),
//Title field
_commonInputWidgets.getTextField(
    "Game Title",
    "Forza Horizon",
    Icons.label,
    _titleController,
    "Game title field cannot be empty",
    focusNode),
//Genre field
_commonInputWidgets.getTextField(
    "Genre",
    "Racing, Simulation, Automobile",
    Icons.view_agenda,
    _genreController,
    "Genre field cannot be empty",
    focusNode),
//Released Date field
_commonInputWidgets.getDatePicker(
    "Released Date",
    Icons.calendar_today,
    _relDateController,
    "Released Date field cannot be empty",
    focusNode),
//Published Date field
_commonInputWidgets.getDatePicker(
    "Published Date",
    Icons.new_releases,
    _pubDateController,
    "Published Date field cannot be empty",
    focusNode),
//No. of users field
_commonInputWidgets.getNumberTextField(
    "No Of Users",
    "2",
    Icons.person,
    true,
    _noOfUsersController,
    "No. of users field cannot be empty",
    focusNode),
//Brief Description field
_commonInputWidgets.getTextArea(
    "Brief Description",
    "This will appear on main screen",
    Icons.assignment,
    _briefDescController,
    "Brief Description field cannot be empty",
    focusNode),
//Full Description field
_commonInputWidgets.getTextArea(
    "Full Description",
    "This will appear on detail screen",
    Icons.videogame_asset,
    _fullDescController,
    "Full Description field cannot be empty",
    focusNode),
//ESRB Rating dropdown field
_getDropDown("ESRB Rating", Icons.rate_review),
//Developer field
_commonInputWidgets.getTextField(
    "Developer",
    "Playground Games",
    Icons.build,
    _developerController,
    "Developer field cannot be empty",
    focusNode),
//User score field
_commonInputWidgets.getNumberTextField(
    "User Score",
    "7.8",
    Icons.score,
    false,
    _userScoreController,
    "User Score field cannot be empty",
    focusNode),
new Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    new RaisedButton(
        padding: const EdgeInsets.all(10.0),
        onPressed: () async {
          setState(() {
            _greyOutBackground = true;
          });
          //if the form fields are validated
          if (_formKey.currentState.validate()) {
            Scaffold.of(context).showSnackBar(SnackBar(
              content: Text('Adding New Game Review'),
            ));
            //uploads the image
            ImageUploader uploader = new ImageUploader(
                _titleController.text +
                    "-" +
                    _generateID().toString(),
                _image);
            var imageURL = await uploader.uploadFile();
            GameRecord game = new GameRecord(
                _generateID(),
                _titleController.text,
                _pubDateController.text,
                _briefDescController.text,
                imageURL,
                _genreController.text,
                _developerController.text,
                _relDateController.text,
                _fullDescController.text,
                _selectedESRBRating,
                _userScoreController.text,
                _noOfUsersController.text,
                null);
            await CRUD().addGame(game);
            setState(() {
              _greyOutBackground = false;
            });
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => MyHomePage()));
          }
        },
        //Submit button
        child: new Text('Submit'),
        color: Colors.orange.withOpacity(0.9),
            shape: RoundedRectangleBorder(
                borderRadius: new BorderRadius.circular(10.0))),
        //reset button
        new RaisedButton(
            padding: const EdgeInsets.all(10.0),
            onPressed: () {
              _formKey.currentState.reset();
              _clearImage();
              _titleController.clear();
              _genreController.clear();
              _relDateController.clear();
              _pubDateController.clear();
              _noOfUsersController.clear();
              _briefDescController.clear();
              _fullDescController.clear();
              _getlist();
              _developerController.clear();
              _userScoreController.clear();
            },
            child: new Text('Reset'),
            color: Colors.orange.withOpacity(0.9),
            shape: RoundedRectangleBorder(
                borderRadius: new BorderRadius.circular(10.0)))
      ],
    ),
  ],
),
),
)
],
),
)
);
}

// this function returns a random integer between 0 and 10000
int _generateID() {
var random = Random();
return random.nextInt(10000);
}

// this function brings the drop down list to its initial state
Future _getlist() async {
return setState(() {
_dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
_selectedESRBRating = _dropDownMenuItems[0].value;
});
}

// this function gets an image from the camera and set to _image
Future _getImageFromCamera() async {
return ImagePicker.pickImage(source: ImageSource.camera).then((file) {
setState(() {
_image = file;
});
});
}

// this function picks an image from the gallery and set to _image
Future _getImageFromGallery() async {
return ImagePicker.pickImage(source: ImageSource.gallery).then((file) {
setState(() {
_image = file;
});
});
}

// the function removes the selected image from the gallery
Future _clearImage() async {
setState(() {
_image = null;
});
}

// this function returns a Column having an Image Picker
Column _getImagePicker() {
return (new Column(
children: <Widget>[
Container(
margin: const EdgeInsets.only(bottom: 20),
child: Text(
'Cover Image',
style: TextStyle(fontSize: 25),
),
),
_image == null ? new Text('No image selected.') : Image.file(_image, height: 187, width:
400, fit: BoxFit.fitWidth,),
// if no image is selected show Text else show the image
_image == null // if no image is selected, show Choose Image button
? new Row(
mainAxisAlignment: MainAxisAlignment.spaceEvenly,
children: <Widget>[
FloatingActionButton(
  heroTag: 'btn_camera',
  onPressed: _getImageFromCamera,
  tooltip: 'Pick Image',
  child: Icon(Icons.add_a_photo),
),
FloatingActionButton(
  heroTag: 'btn_gallery',
  onPressed: _getImageFromGallery,
  tooltip: 'Pick Image',
  child: Icon(Icons.wallpaper),),],)
: Container(),
_image != null // if image is selected, show Remove Button
? new Container(
    margin: const EdgeInsets.only(top: 10, bottom: 20),
    child: new RaisedButton(
        child: Container(
          width: 85,
          height: 40,
          child: Row(
            children: <Widget>[Icon(Icons.delete), Text('Remove')],
            mainAxisAlignment: MainAxisAlignment.center,
          ),
        ),
        onPressed: () {
          setState(() {
            _clearImage();
          });
        },
        color: Colors.red.withOpacity(0.80),
      elevation: 0,
      shape: RoundedRectangleBorder(
          borderRadius: new BorderRadius.circular(10.0))),
)
: Container()
],
));
}

// this function adds items in the ratings list to the Drop down menu item
List<DropdownMenuItem<String>> _buildAndGetDropDownMenuItems(List ratings) {
List<DropdownMenuItem<String>> items = List();
for (String rating in ratings) {
items.add(DropdownMenuItem(value: rating, child: Text(rating)));
}
return items;
}

// this function returns a dropdown list
Container _getDropDown(String label, IconData icon) {
return (Container(
margin: const EdgeInsets.only(bottom: 20),
child: Column(
children: <Widget>[
Container(
alignment: Alignment(-1, -1),
margin: const EdgeInsets.only(top: 30, bottom: 10, left: 35),
child: Text(label),
),
Row(
children: <Widget>[
Container(
  child: Icon(icon),
),
Container(
  width: 330,
  margin: const EdgeInsets.only(left: 12),
  child: DropdownButton(
    value: _selectedESRBRating,
    items: _dropDownMenuItems,
    onChanged: (pickedValue) {
      setState(() {
        _selectedESRBRating = pickedValue;
      });
    },
    isExpanded: true,
  ),
)
],
)
],
),
));
}

//function which returns Circular Progress Indicator
Center _getCircularProgressIndicator() {
return Center(
child: new Column(
mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
Container(
height: 50,
width: 50,
child: CircularProgressIndicator(),
),
Container(
margin: const EdgeInsets.only(top: 10),
child: Text(
"Please wait",
style: TextStyle(fontSize: 18),
),
)
],
));
}
}
```

7. List of Games

After a new game is added successfully, the particular game can be viewed on the home page as a card.

7.1 Card

Each game in the list is displayed as a card, as shown below.

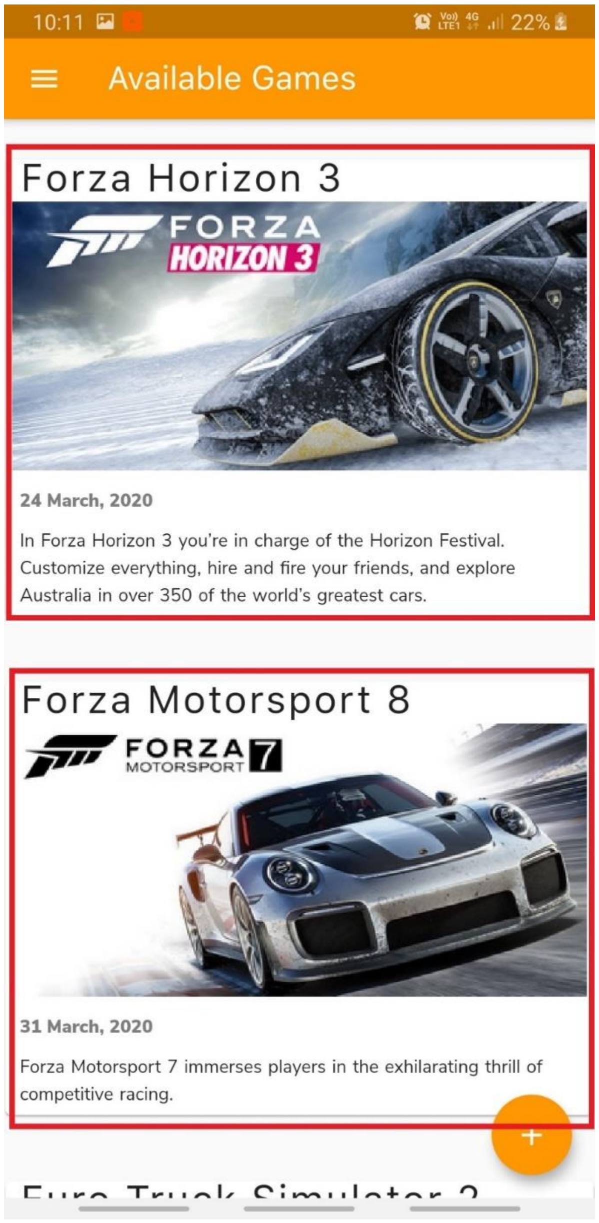

10:11
22%
Available Games
Forza Horizon 3
FORZA HORIZON 3
24 March, 2020
In Forza Horizon 3 you're in charge of the Horizon Festival. Customize everything, hire and fire your friends, and explore Australia in over 350 of the world's greatest cars.
Forza Motorsport 8
FORZA MOTORSPORT 7
31 March, 2020
Forza Motorsport 7 immerses players in the exhilarating thrill of competitive racing.
+
Euro Truck Simulator 2

Figure 7.1: Game card

The following code explains how the card is implemented.

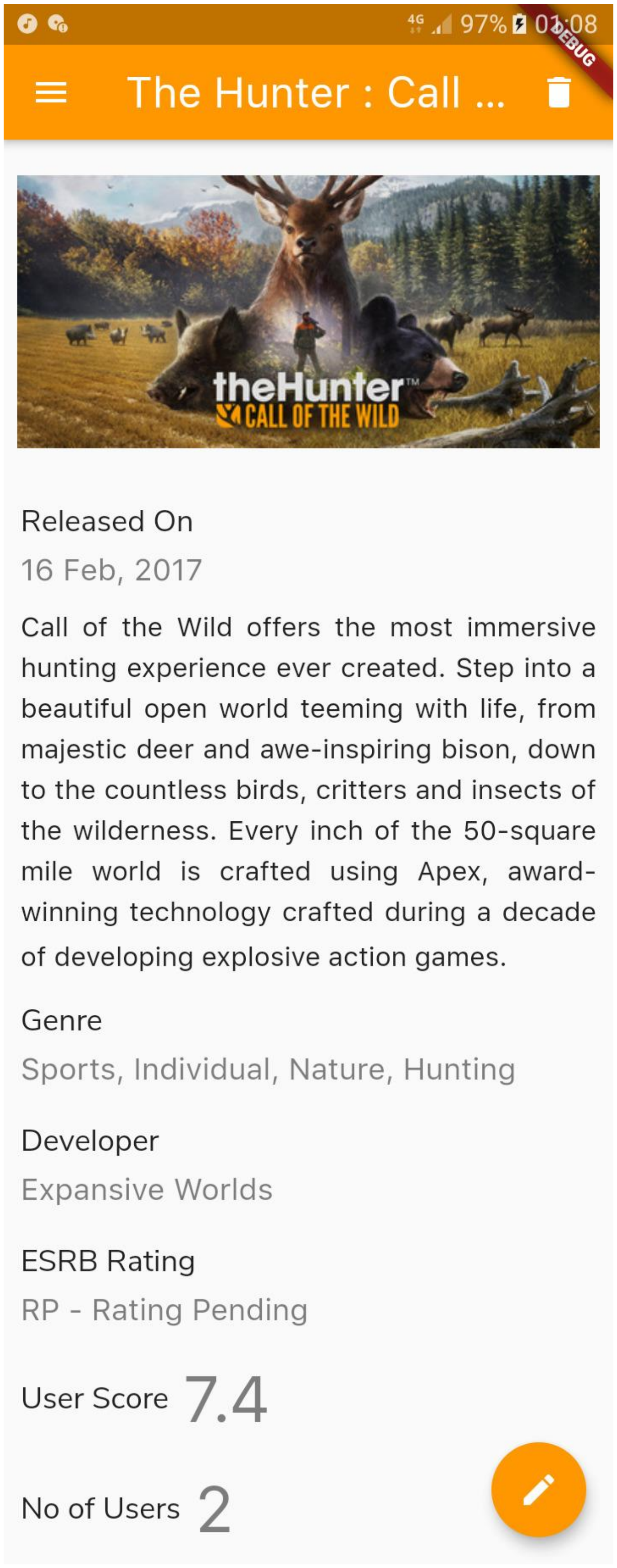
gameCard.dart

```dart
// this class contains methods and attributes used for Card in the List in the app
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:progress_indicators/progress_indicators.dart';
import '../database/gameRecord.dart';
import 'gameDetail.dart';

class GameCard extends StatelessWidget {
  //Constructor
  const GameCard({Key key, this.game, this.selected: false}) : super(key: key);

  final GameRecord game;
  final bool selected;

  @override
  Widget build(BuildContext context) {
    TextStyle textStyle = Theme.of(context).textTheme.display1;
    //if the option is selected, returning a card
    if (selected)
      textStyle = textStyle.copyWith(color: Colors.lightGreenAccent[400]);
    return new Container(
        decoration: new BoxDecoration(boxShadow: [
          new BoxShadow(color: Colors.transparent, blurRadius: 50)
        ]),
        child: new Card(
            margin: const EdgeInsets.only(bottom: 30),
            color: Colors.white,
            child: new InkWell(
              // if the card is pressed navigate to Detailed Screen
              onTap: () => {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) => GameDetailPage(game),
                  ),
                ),
              },
              child: Column(
                children: <Widget>[
                  // Title
                  new Container(
                    margin: const EdgeInsets.only(left: 10),
                    child: new Text(
                      game.title,
                      style: new TextStyle(
                          fontSize: 30,
                          letterSpacing: 1.5,
                          height: 1,
                          fontFamily: 'SanFrancisco'),
                    ),
                  ),
                  // Image of the Game
                  new Container(
//                  child: Image.network(game.imageLink),
                    child: CachedNetworkImage(
                      imageUrl: game.imageLink,
                      width: 400,
                      height: 187,
                      fit: BoxFit.fitWidth,
                      placeholder: (context, url) => Center(
                        child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Container(
                            margin: const EdgeInsets.only(top: 10, bottom: 10),
                            child: Column(
                              children: <Widget>[
                                Container(
                                  child: JumpingDotsProgressIndicator(
                                    fontSize: 20,
                                    numberOfDots: 5,
                                    dotSpacing: 10,
                                    milliseconds: 250,
                                  ),
                                ),
                              ],
                            ),
                          ),
                        ),
                      ),
                      errorWidget: (context, url, error) => Container(
                        margin: const EdgeInsets.only(
                            top: 10, bottom: 10, left: 10, right: 10),
                        child: Center(
                          child: Image.asset(
                              "assets/images/no-image-available.png"),
                        ),
                      ),
                    ),
                  ),
                  //Published Date
                  new Container(
                    padding: const EdgeInsets.all(10.0),
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: <Widget>[
                        new Column(
                          children: <Widget>[
                            Text(game.publishedDate,
                                style: TextStyle(
                                    color: Colors.black.withOpacity(0.5),
                                    fontFamily: 'NunitoSansBlack')),
                          ],
                        )
                      ],
                    ),
                  ),
                  // Game Description
                  Container(
                    margin: const EdgeInsets.only(left: 10, bottom: 5),
                    child: new Text(
                      game.gameDescription,
                      style: new TextStyle(
                          wordSpacing: 1,
                          height: 1.5,
                          fontFamily: 'NunitoSans'),
                    ),
                  )
                ],
                crossAxisAlignment: CrossAxisAlignment.stretch,
              ),
            )));
  }
}
```

After tapping on the card, the user can see the detailed description of the particular game.

7.2 Game Detail

As shown in the image, the user can see a detailed description of the game review. The details include released date, the review, genre, developer, esrb rating, and the user score.



Released On

16 Feb, 2017

Call of the Wild offers the most immersive hunting experience ever created. Step into a beautiful open world teeming with life, from majestic deer and awe-inspiring bison, down to the countless birds, critters and insects of the wilderness. Every inch of the 50-square mile world is crafted using Apex, award-winning technology crafted during a decade of developing explosive action games.

Genre

Sports, Individual, Nature, Hunting

Developer

Expansive Worlds

ESRB Rating

RP - Rating Pending

User Score 7.4

No of Users 2

Figure 7.2: Game details

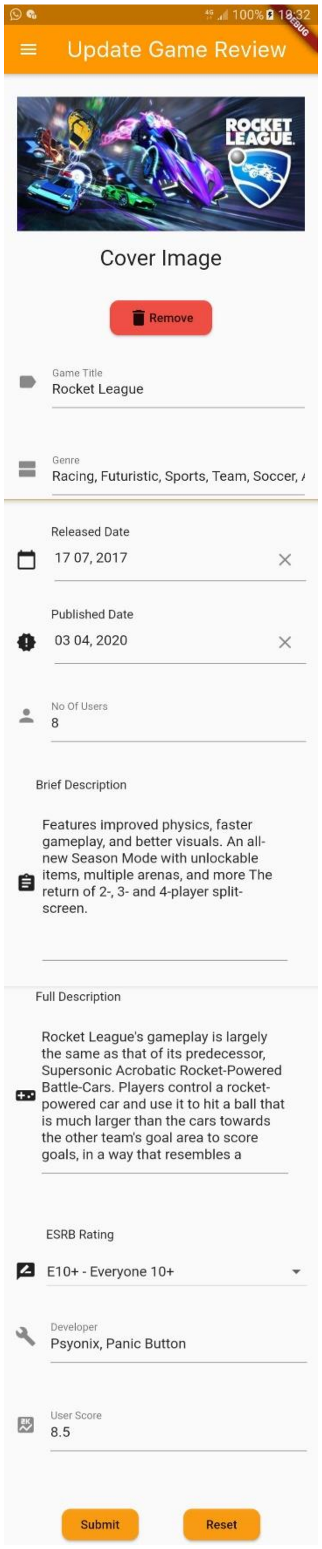
The following code explains how the details of a game are shown to the user.

gameDetail.dart

```dart
// Game Detail Page is implemented in this dart file
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:igamer/screens/updateGame.dart';
import 'package:progress_indicators/progress_indicators.dart';
import 'appBar.dart';
import '../database/gameRecord.dart';
import 'drawer.dart';

//main method of GameDetailPage
void main() {
  runApp(GameDetailPage(null));
}

// ignore: must_be_immutable
class GameDetailPage extends StatelessWidget {
  GameRecord game;

  GameDetailPage(this.game);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: game.title, //returns game title in the Appbar
      home: Scaffold(
        appBar:
            new CustomizedAppBar.fromGameDetail(game.title, context, this.game)
                .getAppBar(),
        // get customized app bar
        drawer: new CustomizedDrawer(context).getDrawer(),
        // get customized app drawer
        body: SingleChildScrollView(
            child: Column(
          children: <Widget>[
            // Image
            new Container(
              padding: const EdgeInsets.all(8.0),
              child: CachedNetworkImage(
                imageUrl: game.imageLink,
                width: 400,
                height: 187,
                fit: BoxFit.fitWidth,
                placeholder: (context, url) => Center(
                  child: Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Container(
                      margin: const EdgeInsets.only(top: 10, bottom: 10),
                      child: Column(
                        children: <Widget>[
                          Container(
                            child: JumpingDotsProgressIndicator(
                              fontSize: 20,
                              numberOfDots: 5,
                              dotSpacing: 10,
                              milliseconds: 250,
                            ),
                          ),
                        ],
                      ),
                    ),
                  ),
                ),
                errorWidget: (context, url, error) => Container(
                  margin: const EdgeInsets.only(
                      top: 10, bottom: 10, left: 10, right: 10),
                  child: Center(
                    child: Image.asset("assets/images/no-image-available.png"),
                  ),
                ),
              ),
            ),

            // Released Date
            getDetailRow("Released On", game.releaseDate),

            // Full Description
            new Container(
              margin: const EdgeInsets.only(left: 10, bottom: 5, right: 10),
              child: new Text(
                game.fullDescription,
                style: new TextStyle(
                    fontSize: 16, height: 1.5, fontFamily: 'SanFrancisco'),
                textAlign: TextAlign.justify,
              ),
            ),

            // Genre
            getDetailRow("Genre", game.genre),

            // Developer
            getDetailRow("Developer", game.developer),

            // ESRB Rating
            getDetailRow("ESRB Rating", game.esrbRating),

            // User Score
            getDetailRowHorizontal("User Score", game.userScore),

            // No of Users
            getDetailRowHorizontal("No of Users", game.noOfUsers),
          ],
          crossAxisAlignment: CrossAxisAlignment.start,
        )),
        //update option
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => UpdateGame(
                        game: game))); // Navigates to Add Game screen
          },
          tooltip: 'Increment',
          child: Icon(Icons.edit),
          backgroundColor: Colors.orange,
            ),
          ),
        );
}

// this function returns a Row in which the label is on top and the value is at bottom
Widget getDetailRow(String label, String value) {
  return new Container(
    padding: const EdgeInsets.all(10.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        new Container(
          child: new Text(
            label,
            style: new TextStyle(fontSize: 18, fontFamily: 'NunitoSans'),
          ),
          margin: const EdgeInsets.only(bottom: 5),
        ),
        new Column(
          children: <Widget>[
            Text(value,
                style: TextStyle(
                    color: Colors.black.withOpacity(0.5),
                    fontSize: 18,
                    fontFamily: 'SanFrancisco'))
          ],
        )
      ],
    ),
  );
}

// this function returns a Row in which the value is place next to the label
Widget getDetailRowHorizontal(String label, String value) {
  return new Container(
    padding: const EdgeInsets.all(10.0),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
        new Container(
          child: new Text(
            label,
            style: new TextStyle(fontSize: 18, fontFamily: 'NunitoSans'),
          ),
          margin: const EdgeInsets.only(right: 8),
        ),
        new Column(
          children: <Widget>[
            Text(value,
                style: TextStyle(
                    color: Colors.black.withOpacity(0.5),
                    fontSize: 38,
                    fontFamily: 'SanFrancisco'))
          ],
        )
      ],
    ),
  );
        }
}
```

8 Update Game

The game reviewers can hopefully update an existing game in the app. After the game is updated successfully, the user gets a confirmation message.



Update Game Review

Cover Image

Remove

Game Title
Rocket League

Genre
Racing, Futuristic, Sports, Team, Soccer, /

Released Date
17 07, 2017

Published Date
03 04, 2020

No Of Users
8

Brief Description

Features improved physics, faster gameplay, and better visuals. An all-new Season Mode with unlockable items, multiple arenas, and more The return of 2-, 3- and 4-player split-screen.

Full Description

Rocket League's gameplay is largely the same as that of its predecessor, Supersonic Acrobatic Rocket-Powered Battle-Cars. Players control a rocket-powered car and use it to hit a ball that is much larger than the cars towards the other team's goal area to score goals, in a way that resembles a

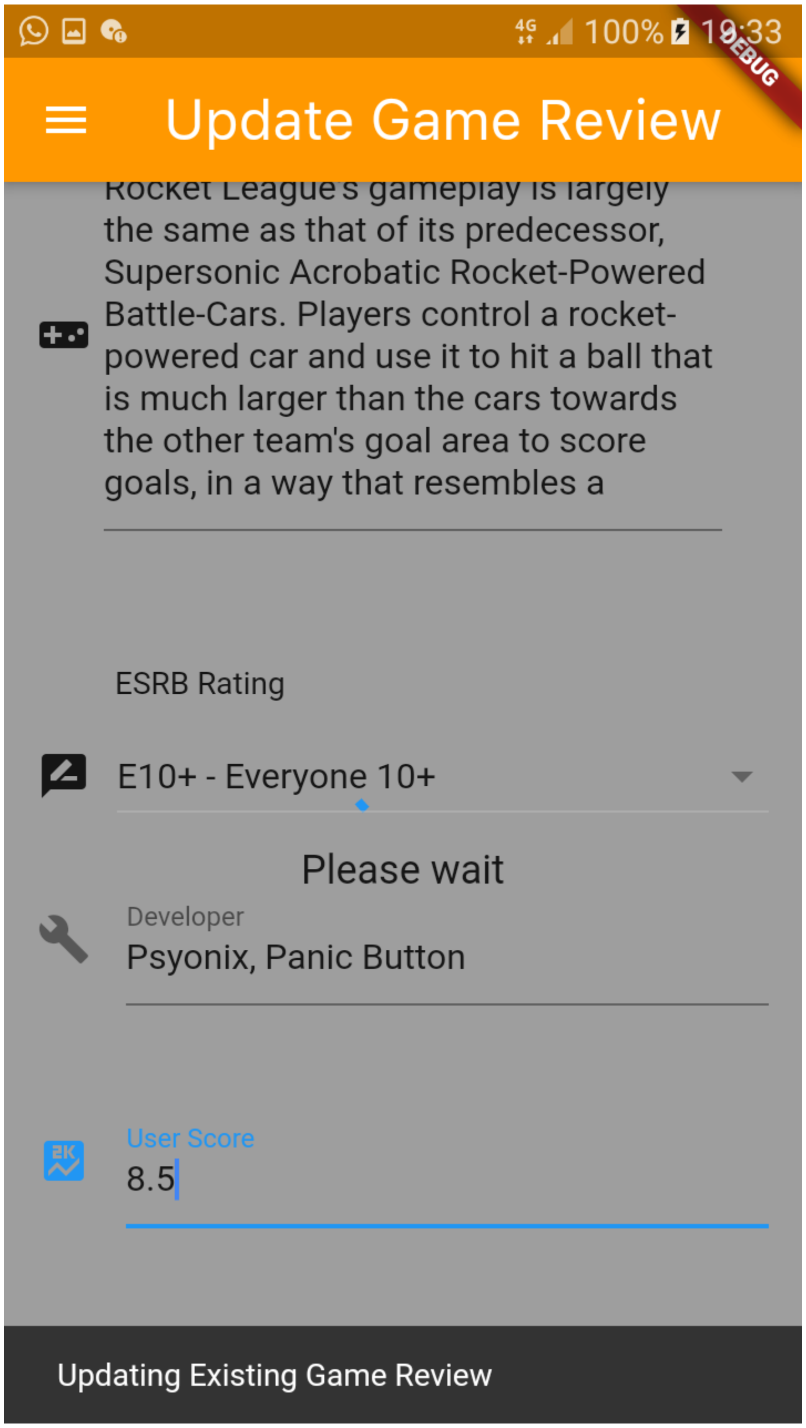
ESRB Rating
E10+ - Everyone 10+

Developer
Psyonix, Panic Button

User Score
8.5

Submit Reset

Figure 8.1: Update game screen



Update Game Review

Rocket League's gameplay is largely the same as that of its predecessor, Supersonic Acrobatic Rocket-Powered Battle-Cars. Players control a rocket-powered car and use it to hit a ball that is much larger than the cars towards the other team's goal area to score goals, in a way that resembles a

ESRB Rating

E10+ - Everyone 10+

Please wait

Developer

Psyonix, Panic Button

User Score

8.5

Updating Existing Game Review

Figure 8.2: Updating a game

updateGame.dart

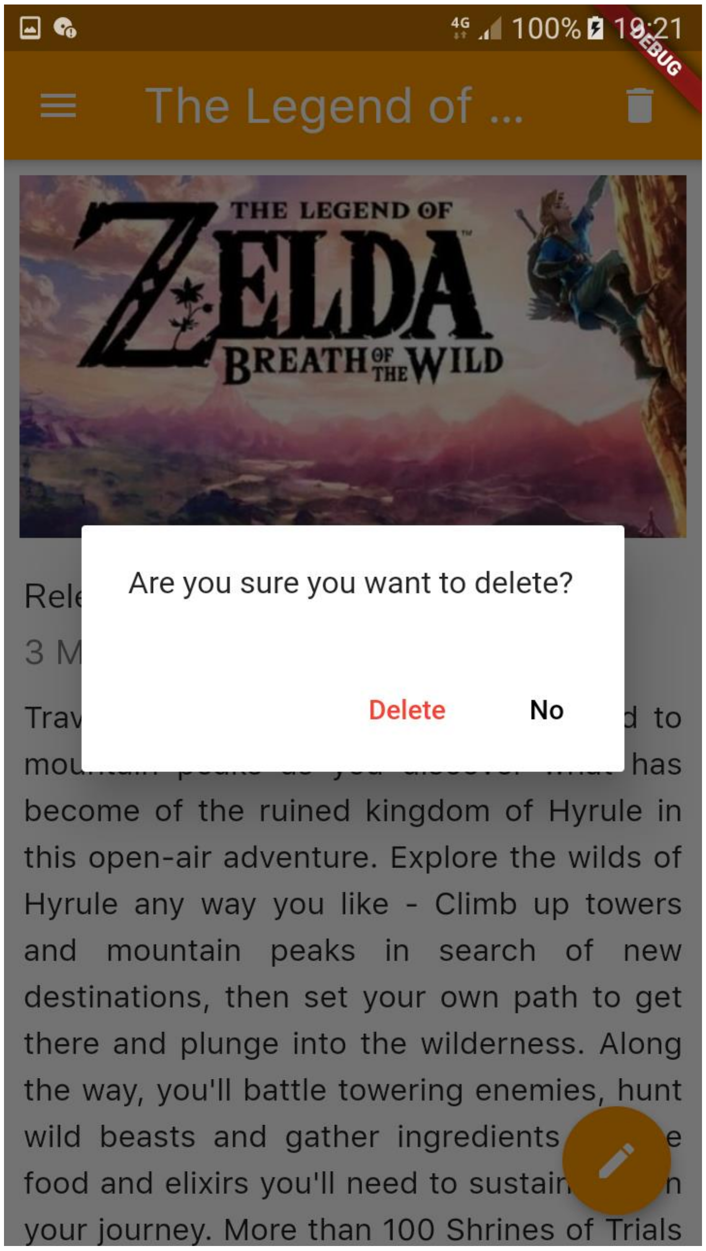
```dart
// In the file update game screen is implemented
import 'dart:async';

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:igamer/common_ui_widgets/drawer.dart';
import 'package:igamer/common_ui_widgets/inputWidgets.dart';
import 'package:igamer/database/crud.dart';
import 'package:igamer/database/gameRecord.dart';
import 'package:igamer/database/imageUploader.dart';
import 'package:progress_indicators/progress_indicators.dart';
import '../common_ui_widgets/appBar.dart';
import 'dart:io';
import 'package:image_picker/image_picker.dart';
import 'main.dart';

// Name of the page
final title = 'Update Game Review';

// Main method
void main() => runApp(UpdateGame());

class UpdateGame extends StatelessWidget {
  final GameRecord game;

  //Constructor
  const UpdateGame({Key key, this.game}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: title,
      home: Scaffold(
        appBar: new CustomizedAppBar(title).getAppBar(),
        //getting custom built app bar
        body: UpdateGameForm(title: title, gameRecord: game),
        drawer: new CustomizedDrawer(context)
            .getDrawer(), //getting custom built app drawer
      ),
    );
  }
}

// Create a Form widget.
class UpdateGameForm extends StatefulWidget {
  final String title;
  final GameRecord gameRecord;

  //Constructor
  UpdateGameForm({Key key, this.title, this.gameRecord}) : super(key: key);

  @override
  UpdateGameFormState createState() {
    return UpdateGameFormState();
  }
}
```

```
class UpdateGameFormState extends State<UpdateGameForm> {
  final _formKey = GlobalKey<FormState>();
  FocusNode focusNode;
  List<DropdownMenuItem<String>> _dropDownMenuItems;
  String _selectedESRBRating;
  File _image;
  bool _greyOutBackground = false;
  bool _showOriginalImage = true;

  //Initializing text editing controllers for update operation
  TextEditingController _titleController = new TextEditingController();
  TextEditingController _genreController = new TextEditingController();
  TextEditingController _relDateController = new TextEditingController();
  TextEditingController _pubDateController = new TextEditingController();
  TextEditingController _noOfUsersController = new TextEditingController();
  TextEditingController _briefDescController = new TextEditingController();
  TextEditingController _fullDescController = new TextEditingController();
  TextEditingController _developerController = new TextEditingController();
  TextEditingController _userScoreController = new TextEditingController();
  CommonInputWidgets _commonInputWidgets = new CommonInputWidgets();

  //Initializing drop down values for ESRB Ratings
  List _ratings = [
    "RP - Rating Pending",
    "EC - Early Childhood",
    "E - Everyone",
    "E10+ - Everyone 10+",
    "T - Teen",
    "M - Mature",
    "AO - Adults Only"
  ];

  // initializing local variables at the beginning of the screen
  @override
  void initState() {
    super.initState();

    focusNode = FocusNode();
    _dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
    _selectedESRBRating = widget.gameRecord.esrbRating;
    _greyOutBackground = false;

    //setting value of image link through File Constructor for image removal operation
    _image = new File(
        widget.gameRecord.imageLink != null ? widget.gameRecord.imageLink : '');
    Image.file(_image);

    //setting values through Text Editing Controllers
    _titleController = TextEditingController(
        text: widget.gameRecord.title != null ? widget.gameRecord.title : '');
    _genreController = TextEditingController(
        text: widget.gameRecord.genre != null ? widget.gameRecord.genre : '');
    _relDateController = TextEditingController(
        text: widget.gameRecord.releaseDate != null
            ? widget.gameRecord.releaseDate
            : '');
    _pubDateController = TextEditingController(
        text: widget.gameRecord.publishedDate != null
            ? widget.gameRecord.publishedDate
            : '');
    _briefDescController = TextEditingController(
        text: widget.gameRecord.gameDescription != null
            ? widget.gameRecord.gameDescription
            : '');
    _fullDescController = TextEditingController(
        text: widget.gameRecord.fullDescription != null
            ? widget.gameRecord.fullDescription
            : '');
    _noOfUsersController = TextEditingController(
        text: widget.gameRecord.noOfUsers != null
            ? widget.gameRecord.noOfUsers
            : '');
    _developerController = TextEditingController(
        text: widget.gameRecord.developer != null
            ? widget.gameRecord.developer
            : '');
    _userScoreController = TextEditingController(
        text: widget.gameRecord.userScore != null
            ? widget.gameRecord.userScore
            : '');
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      backgroundColor:
          _greyOutBackground == true ? Colors.grey : Colors.transparent,
      body: Stack(
        children: <Widget>[
          if (_greyOutBackground) _getCircularProgressIndicator(),
          //display a form with pre-filled fields
          Form(
            key: _formKey,
            child: SingleChildScrollView(
              padding: const EdgeInsets.only(left: 15, right: 15, top: 10),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  //Displays previously uploaded image in the AddForm
                  _showOriginalImage == true
                      ? CachedNetworkImage(
                          imageUrl: widget.gameRecord.imageLink,
                          width: 400,
                          height: 187,
                          fit: BoxFit.fitWidth,
                          placeholder: (context, url) => Center(
                            child: Padding(
                              padding: const EdgeInsets.all(8.0),
                              child: Container(
                                margin:
                                    const EdgeInsets.only(top: 10, bottom: 10),
                                child: Column(
                                  children: <Widget>[
                                    Container(
                                      child: JumpingDotsProgressIndicator(
                                        fontSize: 20,
                                        numberOfDots: 5,
                                        dotSpacing: 10,
                                        milliseconds: 250,
                                      ),
                                    ),
                  ],
                ),
              ),
            ),
          ),
          errorWidget: (context, url, error) => Container(
            margin: const EdgeInsets.only(
                top: 10, bottom: 10, left: 10, right: 10),
            child: Center(
              child: Image.asset(
                  "assets/images/no-image-available.png"),
            ),
          ),
        )
      : Container(),
  _getImagePicker(),
  //Title field
  _commonInputWidgets.getTextField(
      "Game Title",
      "Forza Horizon",
      Icons.label,
      _titleController,
      "Game title field cannot be empty",
      focusNode),
  //Genre field
  _commonInputWidgets.getTextField(
      "Genre",
      "Racing, Simulation, Automobile",
      Icons.view_agenda,
      _genreController,
      "Genre field cannot be empty",
      focusNode),
  //Released Date field
  _commonInputWidgets.getDatePicker(
      "Released Date",
      Icons.calendar_today,
      _relDateController,
      "Released Date field cannot be empty",
      focusNode),
  //Published Date field
  _commonInputWidgets.getDatePicker(
      "Published Date",
      Icons.new_releases,
      _pubDateController,
      "Published Date field cannot be empty",
      focusNode),
  //No. of users field
  _commonInputWidgets.getNumberTextField(
      "No Of Users",
      "2",
      Icons.person,
      true,
      _noOfUsersController,
      "No. of users field cannot be empty",
      focusNode),
  //Brief Description field
  _commonInputWidgets.getTextArea(
      "Brief Description",
      "This will appear on main screen",
      Icons.assignment,
      _briefDescController,
            "Brief Description field cannot be empty",
            focusNode),
        //Full Description field
        _commonInputWidgets.getTextArea(
            "Full Description",
            "This will appear on detail screen",
            Icons.videogame_asset,
            _fullDescController,
            "Full Description field cannot be empty",
            focusNode),
        //ESRB Rating dropdown field
        _getDropDown("ESRB Rating", Icons.rate_review),
        //Developer field
        _commonInputWidgets.getTextField(
            "Developer",
            "Playground Games",
            Icons.build,
            _developerController,
            "Developer field cannot be empty",
            focusNode),
        //User score field
        _commonInputWidgets.getNumberTextField(
            "User Score",
            "7.8",
            Icons.score,
            false,
            _userScoreController,
            "User Score field cannot be empty",
            focusNode),
        new Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            new RaisedButton(
                padding: const EdgeInsets.all(10.0),
                onPressed: () async {
                  setState(() {
                    _greyOutBackground = true;
                  });
                  //if the form fields are validated
                  if (_formKey.currentState.validate()) {
                    Scaffold.of(context).showSnackBar(SnackBar(
                      content: Text('Updating Existing Game Review'),
                    ));

                    var imageURL;
                    //uploads the image if the image is updated only
                    if (_showOriginalImage == false) {
                      ImageUploader uploader = new ImageUploader(
                          _titleController.text +
                              "-" +
                              widget.gameRecord.gameID.toString(),
                          _image);
                      imageURL = await uploader.uploadFile();
                    }
                    //keeps the previously added image
                    else {
                      imageURL = widget.gameRecord.imageLink;
                    }
                    GameRecord game = new GameRecord(
                        widget.gameRecord.gameID,
                        _titleController.text,
                                    _pubDateController.text,
                                    _briefDescController.text,
                                    imageURL,
                                    _genreController.text,
                                    _developerController.text,
                                    _relDateController.text,
                                    _fullDescController.text,
                                    _selectedESRBRating,
                                    _userScoreController.text,
                                    _noOfUsersController.text,
                                    widget.gameRecord.reference);
                                await CRUD().editGame(game, game.reference);
                                setState(() {
                                  _greyOutBackground = false;
                                });
                                Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                        builder: (context) => MyHomePage()));
                              }
                              setState(() {
                                _greyOutBackground = false;
                              });
                            },
                            //Submit button
                            child: new Text('Submit'),
                            color: Colors.orange.withOpacity(0.9),
                            shape: RoundedRectangleBorder(
                                borderRadius: new BorderRadius.circular(10.0))),
                        //reset button
                        new RaisedButton(
                            padding: const EdgeInsets.all(10.0),
                            onPressed: () {
                              _formKey.currentState.reset();
                              _clearImage();
                              _titleController.clear();
                              _genreController.clear();
                              _relDateController.clear();
                              _pubDateController.clear();
                              _noOfUsersController.clear();
                              _briefDescController.clear();
                              _fullDescController.clear();
                              _getlist();
                              _developerController.clear();
                              _userScoreController.clear();
                            },
                            child: new Text('Reset'),
                            color: Colors.orange.withOpacity(0.9),
                            shape: RoundedRectangleBorder(
                                borderRadius: new BorderRadius.circular(10.0)))
                      ],
                    ),
                  ],
                ),
              ),
            )
          ],
        ),
      );
    }
```

```
// this function brings the drop down list to its initial state
Future _getlist() async {
  return setState(() {
    _dropDownMenuItems = _buildAndGetDropDownMenuItems(_ratings);
    _selectedESRBRating = _dropDownMenuItems[0].value;
  });
}

// this function gets an image from the camera and set to _image
Future _getImageFromCamera() async {
  return ImagePicker.pickImage(source: ImageSource.camera).then((file) {
    setState(() {
      _image = file;
    });
  });
}

// this function picks an image from the gallery and set to _image
Future _getImageFromGallery() async {
  return ImagePicker.pickImage(source: ImageSource.gallery).then((file) {
    setState(() {
      _image = file;
    });
  });
}

// the function removes the selected image from the gallery
Future _clearImage() async {
  setState(() {
    _image = null;
  });
}

// this function returns a Column having an Image Picker
Column _getImagePicker() {
  return (new Column(
    children: <Widget>[
      Container(
        margin: const EdgeInsets.only(bottom: 20),
        child: Text(
          'Cover Image',
          style: TextStyle(fontSize: 25),
        ),
      ),
      _image == null ? new Text('No image selected.') : Image.file(_image),
      // if no image is selected show Text else show the image
      _image == null // if no image is selected, show Choose Image button
          ? new Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                FloatingActionButton(
                  heroTag: 'btn_camera',
                  onPressed: _getImageFromCamera,
                  tooltip: 'Pick Image',
                  child: Icon(Icons.add_a_photo),
                ),
                FloatingActionButton(
                  heroTag: 'btn_gallery',
                  onPressed: _getImageFromGallery,
                  tooltip: 'Pick Image',
                  child: Icon(Icons.wallpaper),
                )
              ],
            )
          : Container(),
      _image != null // if image is selected, show Remove Button
          ? new Container(
              margin: const EdgeInsets.only(top: 10, bottom: 20),
              child: RaisedButton(
                  child: Container(
                    width: 85,
                    height: 40,
                    child: Row(
                      children: <Widget>[Icon(Icons.delete), Text('Remove')],
                      mainAxisAlignment: MainAxisAlignment.center,
                    ),
                  ),
                  onPressed: () {
                    setState(() {
                      _clearImage();
                      _showOriginalImage = false;
                    });
                  },
                  color: Colors.red.withOpacity(0.9),
                  shape: RoundedRectangleBorder(
                      borderRadius: new BorderRadius.circular(10.0))),
            )
          : Container()
    ],
  ));
}

// this function adds items in the ratings list to the Drop down menu item
List<DropdownMenuItem<String>> _buildAndGetDropDownMenuItems(List ratings) {
  List<DropdownMenuItem<String>> items = List();
  for (String rating in ratings) {
    items.add(DropdownMenuItem(value: rating, child: Text(rating)));
  }
  return items;
}

// this function returns a dropdown list
Container _getDropDown(String label, IconData icon) {
  return (Container(
    margin: const EdgeInsets.only(bottom: 20),
    child: Column(
      children: <Widget>[
        Container(
          alignment: Alignment(-1, -1),
          margin: const EdgeInsets.only(top: 30, bottom: 10, left: 35),
          child: Text(label),
        ),
        Row(
          children: <Widget>[
            Container(
              child: Icon(icon),
            ),
            Container(
              width: 330,
              margin: const EdgeInsets.only(left: 12),
              child: DropdownButton(
                value: _selectedESRBRating,
                    items: _dropDownMenuItems,
                    onChanged: (pickedValue) {
                      setState(() {
                        _selectedESRBRating = pickedValue;
                      });
                    },
                    isExpanded: true,
                  ),
                )
              ],
            )
          ],
        ),
      ));
    }

    //function which returns Circular Progress Indicator
    Center _getCircularProgressIndicator() {
      return Center(
          child: new Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Container(
            height: 50,
            width: 50,
            child: CircularProgressIndicator(),
          ),
          Container(
            margin: const EdgeInsets.only(top: 10),
            child: Text(
              "Please wait",
              style: TextStyle(fontSize: 18),
            ),
          )
        ],
      ));
    }
  }
```

9. Delete Game

A game reviewer can delete an existing game by tapping on the delete icon in the app bar of a particular game. Then, he/she will get a confirmation box either to delete the game or not.



Are you sure you want to delete?

Rele

3 M

Trav d to mou has become of the ruined kingdom of Hyrule in this open-air adventure. Explore the wilds of Hyrule any way you like - Climb up towers and mountain peaks in search of new destinations, then set your own path to get there and plunge into the wilderness. Along the way, you'll battle towering enemies, hunt wild beasts and gather ingredients e food and elixirs you'll need to sustain n your journey. More than 100 Shrines of Trials

Figure 9.1: Confirmation box when deleting

10. Database

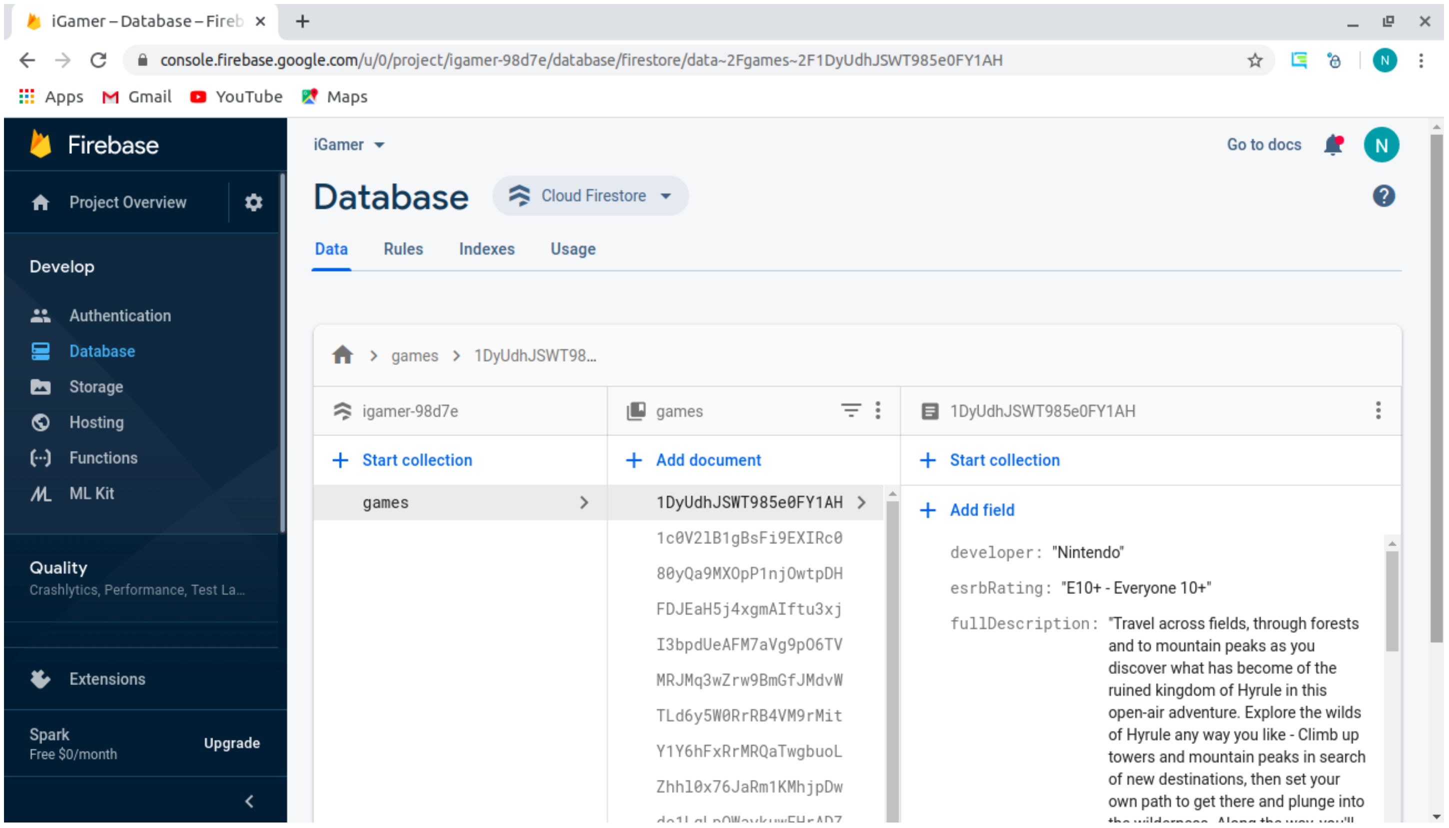
Class crud is used to add, update, delete, get the game records stored in the database.

crud.dart

```
// This class contains the necessary CRUD actions and attributes for the games used in the app
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:igamer/database/gameRecord.dart';

//
gameID|this.title|publishedDate|gameDescription|imageLink|genre|developer|releaseDate|fullDescript
ion|esrbRating|userScore|noOfUsers

class CRUD {
  //makes singleton
  static final CRUD _crud = CRUD._internal();
  Firestore _db = Firestore.instance;

  //internal constructor
  CRUD._internal();

  factory CRUD() {
    return _crud;
  }

  // Collection name
  final String _collection = "games";

  // Add a new game
  Future<void> addGame(GameRecord gameRecord) async {
    await _db
        .collection("games")
        .add(gameRecord.toMap())
        .then((documentReference) {
      print(documentReference.documentID);
    }).catchError((e) {
      print("Execution terminated with the Exception : " + e);
    });
  }

  // Get all games
  Stream<QuerySnapshot> getGames() {
    return _db.collection(_collection).snapshots();
  }

  // Update an existing game
  Future<void> editGame(
      GameRecord gameRecord, DocumentReference reference) async {
    await _db
        .collection("games")
        .document(reference.documentID)
        .updateData(gameRecord.toMap())
        .then((documentReference) {
      print(reference.documentID);
    }).catchError((e) {
      print("Execution terminated with the Exception : " + e);
    });
  }

  //Delete an existing game
  Future<void> deleteGame(
      BuildContext context, DocumentReference reference) async {
    if (await showConfirmationDialog(context)) {
      try {
        await _db.collection("games").document(reference.documentID).delete();
      } catch (e) {
        print(e);
      }
    }
  }

  //show confirmation box when deleting
  Future<bool> showConfirmationDialog(BuildContext context) async {
    return showDialog(
        context: context,
        barrierDismissible: true,
        builder: (context) => AlertDialog(
              content: Text('Are you sure you want to delete?'),
              actions: <Widget>[
                FlatButton(
                  textColor: Colors.red,
                  child: Text('Delete'),
                  onPressed: () => Navigator.pop(context, true),
                ),
                FlatButton(
                  textColor: Colors.black,
                  child: Text('No'),
                  onPressed: () => Navigator.pop(context),
                )
              ],
            ));
  }
}
```

Firebase Store

The following image shows the firebase console. Here we can view the database, and storage.



games

1DyUdhJSWT985e0FY1AH
1c0V21B1gBsFi9EXIRc0
80yQa9MXOpP1njOwtpDH
FDJEaH5j4xgmAIftu3xj
I3bpdUeAFM7aVg9p06TV
MRJMq3wZrw9BmGfJMdvW
TLd6y5W0RrRB4VM9rMit
Y1Y6hFxRrMRQaTwgbuoL
Zhh10x76JaRm1KMhjpDw

developer: "Nintendo"
esrbRating: "E10+ - Everyone 10+"
fullDescription: "Travel across fields, through forests and to mountain peaks as you discover what has become of the ruined kingdom of Hyrule in this open-air adventure. Explore the wilds of Hyrule any way you like - Climb up towers and mountain peaks in search of new destinations, then set your own path to get there and plunge into

Figure 10.1: Firebase database

11. Game Record

The class gameRecord is used to represent a game object, a gameRecord object is used when passing values to the dataset during crud operations.

gameRecord.dart

```
// This class contains attributes and relevant methods for the Game entity
import 'package:cloud_firestore/cloud_firestore.dart';

class GameRecord {
  final int gameID;
  final String title;
  final String publishedDate;
  final String gameDescription;
  final String imageLink;
  final String genre;
  final String developer;
  final String releaseDate;
  final String fullDescription;
  final String esrbRating;
  final String userScore;
  final String noOfUsers;
  final DocumentReference reference;
  // Constructor
  GameRecord(this.gameID, this.title, this.publishedDate, this.gameDescription,
      this.imageLink, this.genre, this.developer, this.releaseDate,
      this.fullDescription, this.esrbRating, this.userScore, this.noOfUsers, this.reference);
  // this function maps the attributes received from map to GameRecord class
  // meanwhile this function also asserts if the all the mapping attributes are null
  GameRecord.fromMap(Map<String, dynamic> map, {this.reference}):
        assert(map['gameID'] != null),
        assert(map['title'] != null),
        assert(map['publishedDate'] != null),
        assert(map['gameDescription'] != null),
        assert(map['imageLink'] != null),
        assert(map['genre'] != null),
        assert(map['developer'] != null),
        assert(map['releaseDate'] != null),
        assert(map['fullDescription'] != null),
        assert(map['esrbRating'] != null),
        assert(map['userScore'] != null),
        assert(map['noOfUsers'] != null),
        gameID = map['gameID'],
        title = map['title'],
        publishedDate = map['publishedDate'],
        gameDescription = map['gameDescription'],
        imageLink = map['imageLink'],
        genre = map['genre'],
        developer = map['developer'],
        releaseDate = map['releaseDate'],
        fullDescription = map['fullDescription'],
        esrbRating = map['esrbRating'],
        userScore = map['userScore'],
            noOfUsers = map['noOfUsers'];
    // this function maps the attributes received to map to GameRecord class
    Map<String, dynamic> toMap(){
      return {
        'gameID': gameID,
        'title': title,
        'publishedDate': publishedDate,
        'gameDescription': gameDescription,
        'imageLink': imageLink,
        'genre': genre,
        'developer': developer,
        'releaseDate': releaseDate,
        'fullDescription': fullDescription,
        'esrbRating': esrbRating,
        'userScore': userScore,
        'noOfUsers': noOfUsers,
      };
    }
    //used for listing games
    GameRecord.fromSnapshot(DocumentSnapshot snapshot):
            this.fromMap(snapshot.data, reference: snapshot.reference);
    @override
    String toString() => "Record<$title:$title>";
}
```

12. Image Uploader

The imageUploader is a separate class implemented to perform the uploading function of an image in the add/update game. The following code explains how an image is uploaded to the firebase storage.

imageUploader.dart

```dart
// This class is used for uploading an Image to firebase
import 'package:firebase_storage/firebase_storage.dart'; // For File Upload To Firestoreker
import 'package:path/path.dart' as Path;
import 'dart:io';

class ImageUploader {
  String _imagePath;
  File _image;
  String __uploadedFileURL = "null";
  final String _remotePath = "images/";
  // Constructor
  ImageUploader(this._imagePath, this._image);
  // this function upload the image to the firebase a returns the downloadable image link in return
  Future<String> uploadFile() async {
    StorageReference storageReference = FirebaseStorage.instance
        .ref()
        .child('$_remotePath/${Path.basename(_imagePath)}');
    StorageUploadTask uploadTask = storageReference.putFile(_image);
    await uploadTask.onComplete;
    await storageReference.getDownloadURL().then((fileURL) {
      this.__uploadedFileURL = fileURL;
    });
    return this.__uploadedFileURL;
  }
}
```

13. About Screen

The about screen provides detail of the app, such as terms and conditions, app version, and the developer to which app belongs.

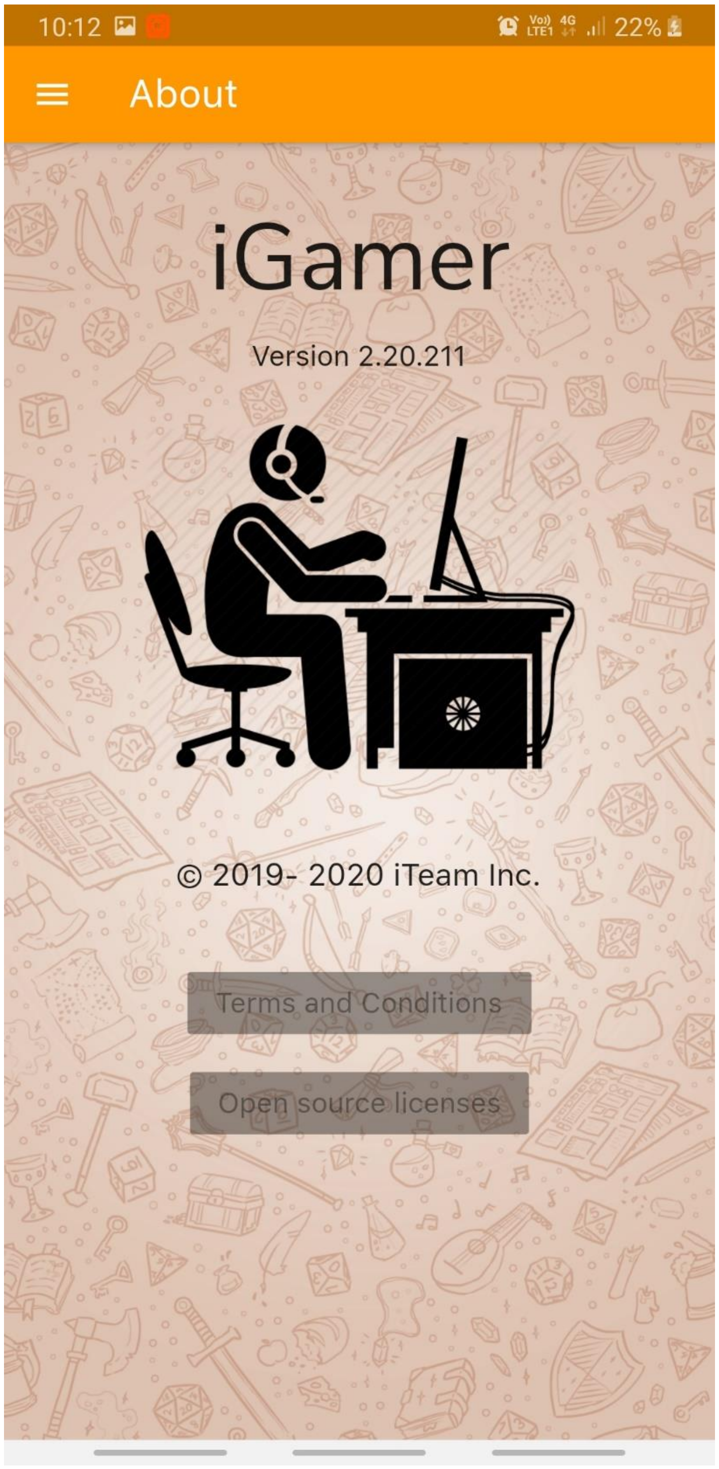


Figure 13.1: About screen

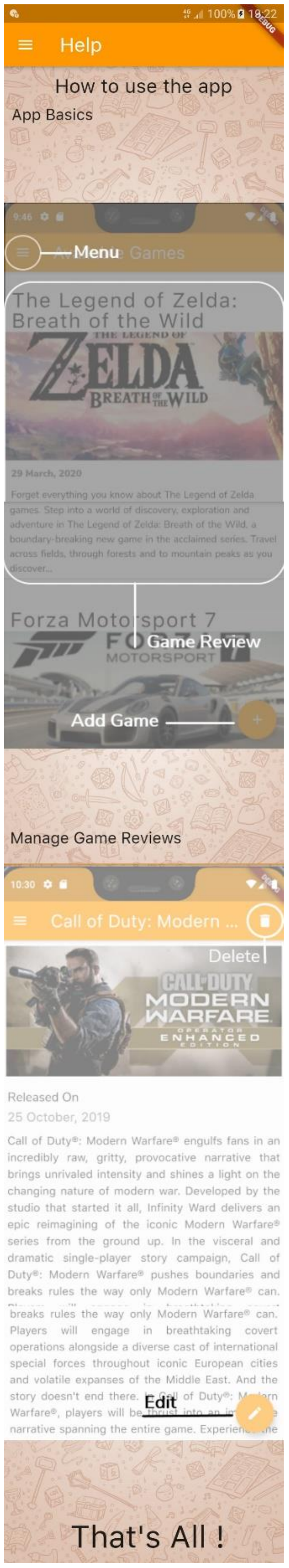
about.dart

```
// In the file About screen is implemented
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:igamer/common_ui_widgets/appBar.dart';
import 'package:igamer/common_ui_widgets/drawer.dart';
// Page Title
final String pageTitle = "About";
// Main method
void main() {
  runApp(MaterialApp(
    home: AboutScreenPage(),
    theme: ThemeData(fontFamily: 'SanFrancisco'),
  ));
}
//About us page
class AboutScreenPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      body: new Stack(
        children: <Widget>[
          // background image
          new Container(
            decoration: new BoxDecoration(
              image: new DecorationImage(
                  image: new AssetImage("assets/images/aboutus.png"),
                  fit: BoxFit.cover,
                  colorFilter: new ColorFilter.mode(
                      Colors.black.withOpacity(0.1), BlendMode.softLight)),
            ),
          ),
          //returns the title
          SingleChildScrollView(
            child: Column(
            children: <Widget>[
              Container(
                margin: const EdgeInsets.only(top: 30),
                alignment: Alignment.center,
                child: Text(
                  "iGamer",
                  style: TextStyle(fontSize: 60, fontFamily: 'NunitoSans'),
                ),
              ),
              //returns the version of the app
              Container(
                margin: const EdgeInsets.only(top: 10),
                alignment: Alignment.center,
                child: Text(
                  "Version 2.20.211",
                  style: TextStyle(fontSize: 18, fontFamily: 'SanFrancisco'),
                ),
              ),
              //returns image
              Container(
                alignment: Alignment.center,
                margin: const EdgeInsets.only(top: 30, bottom: 20),
                height: 200,
                child: Image.asset('assets/images/gamer.png'),
              ),
              Container(
                margin: const EdgeInsets.only(top: 30, bottom: 20),
                child: Text(
                  '© 2019- 2020 iTeam Inc.',
                  style: TextStyle(fontSize: 20, fontFamily: 'SanFrancisco'),
                ),
              ),
              Container(
                margin: const EdgeInsets.only(top: 30),
                child: RaisedButton(
                  child: Text(
                    'Terms and Conditions',
                    style: TextStyle(fontSize: 18, fontFamily: 'SanFrancisco'),
                  ),
                ),
              ),
              Container(
                margin: const EdgeInsets.only(top: 20),
                child: RaisedButton(
                  child: Text(
                      'Open source licenses',
                      style: TextStyle(fontSize: 18, fontFamily: 'SanFrancisco')),
                ),
              ),
            ],
          ),
          ),
        ],
      ),
      drawer: new CustomizedDrawer(context).getDrawer(),
      appBar: new CustomizedAppBar(pageTitle).getAppBar(),
    );
  }
}
```

14. Help Screen

With this page, a new user can learn how to use the iGamer app and about its basic functionalities.

1. App Basics
2. Manage Game Reviews



29 March, 2020

Forget everything you know about The Legend of Zelda games. Step into a world of discovery, exploration and adventure in The Legend of Zelda: Breath of the Wild, a boundary-breaking new game in the acclaimed series. Travel across fields, through forests and to mountain peaks as you discover...

Released On

25 October, 2019

Call of Duty®: Modern Warfare® engulfs fans in an incredibly raw, gritty, provocative narrative that brings unrivaled intensity and shines a light on the changing nature of modern war. Developed by the studio that started it all, Infinity Ward delivers an epic reimagining of the iconic Modern Warfare® series from the ground up. In the visceral and dramatic single-player story campaign, Call of Duty®: Modern Warfare® pushes boundaries and breaks rules the way only Modern Warfare® can.

breaks rules the way only Modern Warfare® can.

Players will engage in breathtaking covert operations alongside a diverse cast of international special forces throughout iconic European cities and volatile expanses of the Middle East. And the story doesn't end there. In Call of Duty®: Modern Warfare®, players will be thrust into an im narrative spanning the entire game. Experien the

Figure 14.1: Help Screen

help.dart

```dart
// In the file help screen is implemented
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:igamer/common_ui_widgets/appBar.dart';
import 'package:igamer/common_ui_widgets/drawer.dart';
// Name of the page
final String pageTitle = "Help";
// Main method
void main() {
  runApp(MaterialApp(home: HelpScreenPage()));
}
class HelpScreenPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      body: new Stack(
        children: <Widget>[
          // Background Image
          new Container(
            decoration: new BoxDecoration(
              image: new DecorationImage(
                  image: new AssetImage("assets/images/aboutus.png"),
                  fit: BoxFit.cover,
                  colorFilter: new ColorFilter.mode(
                      Colors.black.withOpacity(0.1), BlendMode.softLight)),
            ),
          ),
          SingleChildScrollView(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                Container(
                  alignment: Alignment.center,
                  margin: const EdgeInsets.all(10),
                  child: Text(
                    "How to use the app",
                    style: TextStyle(fontFamily: 'SanFrancisco', fontSize: 25),
                  ),
                ),
                Container(
                  alignment: Alignment.centerLeft,
                  margin: const EdgeInsets.only(left: 10),
                  child: Text(
                    "App Basics",
                    style: TextStyle(fontFamily: 'SanFrancisco', fontSize: 20),
                  ),
                ),
                Container(
                  alignment: Alignment.center,
                  child: Image.asset(
                    'assets/images/help_one.png',
                    width: 400,
                    height: 900,
                        fit: BoxFit.fitWidth,
                      ),
                    ),
                    Container(
                      alignment: Alignment.centerLeft,
                      margin: const EdgeInsets.only(left: 10),
                      child: Text(
                        "Manage Game Reviews",
                        style: TextStyle(fontFamily: 'SanFrancisco', fontSize: 20),
                      ),
                    ),
                    Container(
                      alignment: Alignment.center,
                      child: Image.asset(
                        'assets/images/help_two.png',
                        width: 400,
                        height: 900,
                        fit: BoxFit.fitWidth,
                      ),
                    ),
                    Container(
                      alignment: Alignment.center,
                      margin: const EdgeInsets.only(left: 10, bottom: 10),
                      child: Text(
                        "That's All !",
                        style: TextStyle(fontFamily: 'SanFrancisco', fontSize: 40),
                      ),
                    ),
                  ],
                ),
              )
            ],
          ),
      drawer: new CustomizedDrawer(context).getDrawer(),
      // getting Custom Built Drawer
      appBar: new CustomizedAppBar(pageTitle)
          .getAppBar(), // getting Custom Built App Bar
    );
  }
}
```

15. Common UI Widgets

Common UI widgets like app bar, app drawer, input fields are extracted to separate classes to improve code reusability,

15.1 App Bar

appBar.dart

```dart
// This class contains attributes and methods for a Customized App Bar
import 'package:flutter/material.dart';
import 'package:igamer/database/crud.dart';
import 'package:igamer/database/gameRecord.dart';
import 'package:igamer/screens/main.dart';


class CustomizedAppBar {
  final Color backgroundColor = Colors.orange;
  String title;
  BuildContext context;
  GameRecord game;

  // Constructor
  CustomizedAppBar(this.title);

  CustomizedAppBar.fromGameDetail(
      String title, BuildContext context, GameRecord game) {
    this.title = title;
    this.context = context;
    this.game = game;
  }

  // this function returns the Customized App Bar
  Widget getAppBar() {
    return new AppBar(
      title: Text(
        title,
        style: TextStyle(fontSize: 25, fontFamily: 'SanFrancisco'),
      ),
      backgroundColor: backgroundColor,
      actions: <Widget>[
        //delete option
        game != null
            ? Padding(
                padding: EdgeInsets.only(right: 20.0),
                child: GestureDetector(
                  onTap: () async {
                    await new CRUD().deleteGame(context, game.reference);
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => MyHomePage()),
                    ); //
                  },
                    child: Icon(Icons.delete),
                  ))
              : Container(),
        ],
      );
    }
  }
```

15.2 Drawer

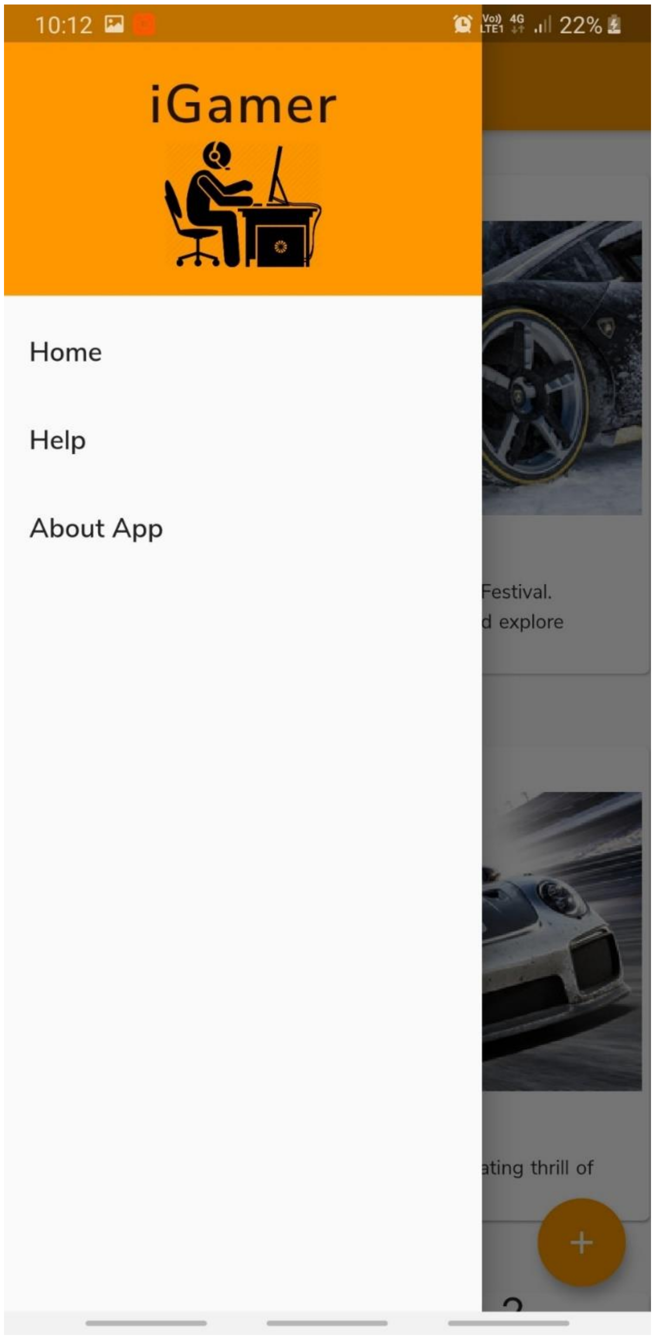


Figure 15.1: App drawer

drawer.dart

```
// this class contains attributes and methods for App Drawer
import 'package:flutter/material.dart';
import 'package:igamer/screens/about.dart';
import 'package:igamer/screens/help.dart';
import 'package:igamer/screens/main.dart';

class CustomizedDrawer {
  final Color backgroundColor = Colors.orange;
  BuildContext context;
  CustomizedDrawer(this.context);
  // this function returns a Customized App Drawer
  Widget getDrawer() {
    return new Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: <Widget>[
          DrawerHeader(
            child: new Column(
              children: <Widget>[
                new Container(
                  child: Text('iGamer',
                    style: TextStyle(fontSize: 38, letterSpacing: 1.5 , fontFamily: 'NunitoSansSemiBold'),),
                ),
                new Container(
                  child: new Image.asset('assets/images/gamer.png'),
                  height: 80,
                  width: 100,
                )
              ],
            ),
            decoration: BoxDecoration(color: Colors.orange),
          ),
          ListTile(
            title: Text('Home', style: TextStyle(fontFamily: 'NunitoSansSemiBold', fontSize: 19),),
            onTap: () => {
              Navigator.push(context, MaterialPageRoute(builder: (context) => new MyHomePage()))},
          ),
          ListTile(
            title: Text('Help', style: TextStyle(fontFamily: 'NunitoSansSemiBold', fontSize: 19),),
            onTap: () => {
              Navigator.push(context, MaterialPageRoute(builder: (context) => new HelpScreenPage()))},
          ),
          ListTile(
            title: Text('About App', style: TextStyle(fontFamily: 'NunitoSansSemiBold', fontSize: 19),),
            onTap: () => {Navigator.push(context, MaterialPageRoute(builder: (context) => new
AboutScreenPage()))},
          )
        ],
      ),
    );
  }}
```

The following class is used by addGame.dart and updateGame.dart, for displaying the input fields of a game.

inputWidgets.dart

```
// this class contains all the common input widgets used in the app
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:datetime_picker_formfield/datetime_picker_formfield.dart';
import 'package:igamer/screens/addGame.dart';
import 'package:intl/intl.dart';

class CommonInputWidgets {
  // this function returns a TextField
  Container getTextField(String labelText, String hintText, IconData icon,
      TextEditingController controller, String validator, FocusNode focusNode) {
    return (Container(
      height: 100,
      child: TextFormField(
        //focus node
        textInputAction: TextInputAction.next,
        onEditingComplete: (){
          FocusScope.of(new AddGameFormState().context).requestFocus(focusNode);
        },
        focusNode: new AddGameFormState().focusNode,
        //decoration to the field
        decoration: InputDecoration(
            labelText: labelText, hintText: hintText, icon: Icon(icon)),
        controller: controller,
        autocorrect: true,
        autofocus: true,
        //validations
        autovalidate: true,
        validator: (value){
          if(value == null || value.isEmpty){
            return validator;
          }
          if(value.trim() == ""){
            return "Only Space is Not Valid !!!";
          }
          return null;
        },
      ),
    ));
  }
  // this function return a date picker
  Container getDatePicker(
      String label, IconData icon, TextEditingController controller, String validator, FocusNode focusNode) {
```

```
    //date format
    final format = DateFormat("dd MM, yyyy");
    return Container(
      margin: const EdgeInsets.only(bottom: 30),
      child: Column(children: <Widget>[
        Container(
          alignment: Alignment(-1, -1),
          child: Text(label),
          margin: const EdgeInsets.only(left: 40),
        ),
        Row(
          children: <Widget>[
            Container(
              child: new Icon(icon),
            ),
            Container(
              width: 335,
              margin: const EdgeInsets.only(left: 20),
              child: DateTimeField(
                format: format,
                //uses datepicker option
                onShowPicker: (context, currentValue) {
                  return showDatePicker(
                      context: context,
                      firstDate: DateTime(1900),
                      initialDate: currentValue ?? DateTime.now(),
                      lastDate: DateTime(2100));
                },
                  controller: controller,
                  autocorrect: true,
                  autofocus: true,
                  //validations
                  autovalidate: true,
                  validator: (value) {
                    if (value.toString() == null) {
                      return validator;
                    }
                    else {
                      return null;
                    }
                  }
              ),
            )
          ],
        )
      ]),
    );
  }
  // this function returns a Number Text Field
  // if the parameter onlyDigits is true , only digits can be entered (not point values)
  Container getNumberTextField(String labelText, String hintText, IconData icon,
      bool onlyDigits, TextEditingController controller, String validator, FocusNode focusNode ) {
    return (Container(
      height: 100,
      child: TextFormField(
        //focus node
        textInputAction: TextInputAction.next,
          onEditingComplete: (){
            FocusScope.of(new AddGameFormState().context).requestFocus(focusNode);
          },
          focusNode: new AddGameFormState().focusNode,
          //field decorations
          decoration: InputDecoration(
              labelText: labelText, hintText: hintText, icon: Icon(icon)),
          controller: controller,
          autocorrect: true,
          autofocus: true,
          //validations
          autovalidate: true,
          validator: (value){
            if(value == null || value.isEmpty){
              return validator;
            }
            if(value.trim() == ""){
              return "Only Space is Not Valid !!!";
            }
            return null;
          },
          keyboardType: TextInputType.number,
          inputFormatters: <TextInputFormatter>[
            if (onlyDigits) WhitelistingTextInputFormatter.digitsOnly
          ],
        ),
    ));
}
// this function returns a Text Area
Container getTextArea(String labelText, String hintText, IconData icon,
        TextEditingController controller, String validator, FocusNode focusNode) {
    return (Container(
          margin: const EdgeInsets.only(bottom: 30),
          child: Column(
            children: <Widget>[
              Container(
                alignment: Alignment(-.8, -1),
                child: Text(labelText),
                margin: const EdgeInsets.only(bottom: 15),
              ),
              Row(
                children: <Widget>[
                  Container(
                    child: new Icon(icon),
                  ),
                  Container(
                    width: 335,
                    child: Card(
                        color: Colors.transparent,
                        elevation: 0,
                        margin: const EdgeInsets.only(left: 5),
                        child: Padding(
                          padding: EdgeInsets.all(1.0),
                          child: TextFormField(
                            maxLines: 8,
                            //focus node
                            textInputAction: TextInputAction.next,
                                        onEditingComplete: (){
                                          FocusScope.of(new AddGameFormState().context).requestFocus(focusNode);
                                        },
                                        focusNode: new AddGameFormState().focusNode,
                                        //field decorations
                                        decoration: InputDecoration(hintText: hintText),
                                        controller: controller,
                                        autocorrect: true,
                                        autofocus: true,
                                        //validations
                                        autovalidate: true,
                                        validator: (value){
                                          if(value == null || value.isEmpty){
                                            return validator;
                                          }
                                          if(value.trim() == ""){
                                            return "Only Space is Not Valid !!!";
                                          }
                                          return null;
                                        },
                                      ),
                                    )),
                            )
                          ],
                        )
                      ],
                    )));
        }
      }
```

15.3 Alert box

The following class is used by main.dart class, for returning the alert box, used for checking the network connectivity.

alertBox.dart

```
// this class contains attributes and method for an Alert Box
import 'package:flutter/material.dart';

class AppAlertBox {
  BuildContext context;
  String title;
  String message;
  String buttonText;
  // Constructor
  AppAlertBox(this.context, this.title, this.message, this.buttonText);
  // this function pops up Alert Box
  showAlertDialog() {
    Widget okButton = FlatButton(
      child: Text(this.buttonText),
      onPressed: () {
        Navigator.pop(this.context);
      },
    );
    // set up the AlertDialog
    AlertDialog alert = AlertDialog(
      title: Text(this.title),
      content: Text(this.message),
      actions: [
        okButton,
      ],
    );
    // show the AlertDialog
    showDialog(
      context: this.context,
      builder: (BuildContext context) {
        return alert;
      },
    );
  }
}
```