INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

BAINISTEOIR BEAG MOBILE APPLICATION

TECHNICAL MANUAL

BY

FRANCIS HALL

Student Number: C00220910

Supervisor: Paul Barry

Submission Date: 20/04/2020

Contents

Set Up Instructions

Downloading APK

The APK file for installation is available for download at the link below:

https://drive.google.com/drive/folders/1xZlDthFlmjxUI9RFZzxT3smR7KneokRI?usp=sharing

Simply follow this link and download the APK on the target Android device.

Installation

Once the APK is downloaded to the device, simply navigate to the APK in the device's File Explorer and select the APK.

Then select 'Install' to install the application.

Note: Since the application is not available via Google Play Store, it is considered a third-party application, and one may need to follow the necessary steps to allow installations from unknown sources in order to install the application.

Access to Web Application

The web application for administrator use is hosted at the link below:

https://bainisteoir-beag.web.app/

A user has been created for demonstration purposes with the credentials:

Email: admin@bainisteoir-beag.app

Password: SoftEng1920

Mobile Application Source Code

ActivityMenu.dart

```
import 'dart:math';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:bainisteoir_beag/CardItem.dart';
import 'AuthFunctions.dart';
import 'CardScreen.dart';
import 'dart:async';
import 'dart:convert';
import 'package:flutter/services.dart' show rootBundle;
import 'RotaDB.dart';
import 'RotaItem.dart';

class ActivityMenu extends StatefulWidget {
  @override
  _ActivityMenuState createState() => _ActivityMenuState();
}

class _ActivityMenuState extends State<ActivityMenu> {
  RotaDB db = RotaDB();
  AuthFunctions _auth = AuthFunctions();
  String leaderString = "";
  List namesList = [];

  //Update a specific document in rotaNames database
  _setChosen(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.setChosen(result.uid, name, chosen);
  }

  //update all documents in rotaNames database, marking
  //each document's chosen value to false
  _resetList() async {
    dynamic result = await _auth.signInAnon();
    db.setChosenAll(result.uid, false);
  }

  //Get Names from ROTA which have not yet been marked as chosen
  Future _getRotaItems() async {
    namesList = [];
    dynamic result = await _auth.signInAnon();

    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    for (var u in qn.documents) {
      RotaItem currentItem = RotaItem(u["uid"], u["name"], u["chosen"]);

      //if currentItem matches the current user id and has not been marked as chosen
      //add it to the list
      if (currentItem.uid == result.uid && currentItem.chosen == false) {
        namesList.add(currentItem);
      }
    }

    //If the list is empty, reset
    if (namesList.length == 0) {
      _resetList();
```

```
    _getRotaItems();
  }
}

//Get Cards for List View
Future<List<CardItem>> _getCards() async {
  var data = await rootBundle.loadString("assets/cardsList.json");
  Map jsonData = json.decode(data);

  List<CardItem> cards = [];

  for (var u in jsonData['Cards']) {
    CardItem currentCard = CardItem(u["front"], u["back"], u["title"]);
    //Add Current card to list
    cards.add(currentCard);
  }

  print(cards.length);
  return cards;
}

//Display the next Leader and update the database
String printLeader() {
  Random rand = new Random();

  //Store Current random Int
  int currentRand = rand.nextInt(namesList.length);
  //Store string to output
  String chosenName = namesList[currentRand].name;
  //Update FireStore document of current item
  _setChosen(chosenName, true);

  return chosenName;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
      backgroundColor: Colors.blue[300],
      appBar: AppBar(
        leading: new IconButton(
          icon: new Icon(Icons.arrow_back, color: Colors.purple),
          onPressed: () => Navigator.of(context).pop(),
        ),
        title: Text(
          'Activities',
          style: TextStyle(fontSize: 50),
        ),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          Container(
              constraints: const BoxConstraints(minWidth: double.infinity),
              height: 500,
              child: FutureBuilder(
                  future: _getCards(),
                  builder: (BuildContext context, AsyncSnapshot snapshot) {
                    if (snapshot.data == null) {
                      return Container(
```

```
                        child: Center(child: Text("Loading")));
                  } else {
                    return ListView.builder(
                      itemCount: snapshot.data.length,
                      itemBuilder: (BuildContext context, int index) {
                        return Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: MaterialButton(
                            color: Colors.yellow[300],
                            elevation: 14.0,
                            shape: RoundedRectangleBorder(
                                borderRadius: BorderRadius.all(
                                    Radius.circular(20.0))),
                            height: 200,
                            minWidth: 500,
                            onPressed: () {
                              CardItem currentCard = new CardItem(
                                  ("assets/" + snapshot.data[index].front),
                                  ("assets/" + snapshot.data[index].back),
                                  (snapshot.data[index].title));
                              Navigator.of(context).push(MaterialPageRoute(
                                  builder: (context) =>
                                      CardScreen(currentCard)));
                            },
                            child: Text(snapshot.data[index].title,
                                style: TextStyle(
                                    fontSize: 48,
                                    color: Colors.deepPurple)),
                          ),
                        );
                      },
                    );
                  }
                })),
          Container(
              constraints: const BoxConstraints(minWidth: double.infinity),
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.only(right: 100),
                      child: Text("$leaderString",
                          style: TextStyle(
                              fontSize: 60, color: Colors.deepPurple)),
                    ),
                    Padding(
                      padding: const EdgeInsets.only(left: 20),
                      child: MaterialButton(
                        color: Colors.yellow[300],
                        elevation: 14.0,
                        shape: RoundedRectangleBorder(
                            borderRadius:
                                BorderRadius.all(Radius.circular(20.0))),
                        height: 100,
                        minWidth: 100,
                        onPressed: () async {
                          await _getRotaItems();
                          //Delay before reloading state
                          await new Future.delayed(
                              const Duration(milliseconds: 500));
                          setState(() {
```

```
                                leaderString = printLeader();
                              });
                            },
                            child: Text("Choose Leader",
                                style: TextStyle(
                                    fontSize: 48, color: Colors.deepPurple)),
                          ),
                        ),
                      ],
                    ),
                  ),
                  height: 80),
            ],
          ));
    }
}
```

AuthFunctions.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'User.dart';

class AuthFunctions {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  User _user(FirebaseUser user) {
    return user != null ? User(uid: user.uid) : null;
  }

  //Allow sign in without providing email or password.
  //This will be used automatically for use with the ROTA database
  //Where names are linked to the current device via the uid
  Future signInAnon() async {
    try {
      AuthResult result = await _auth.signInAnonymously();
      FirebaseUser user = result.user;
      return _user(user);
    } catch (e) {
      print(e.toString());
      return null;
    }
  }
}
```

CardItem.dart

```
// This is an object which will handle the necessary attributes of a card,
// ie. the name of the card (title) and the file names for the front and back
// images (front & back)
class CardItem {
  String front;
  String back;
  String title;

  CardItem(front, back, title) {
    this.front = front;
    this.back = back;
    this.title = title;
  }

  void setFront(String front) {
    this.front = front;
  }

  void setBack(String back) {
    this.back = back;
  }

  void setTitle(String title) {
    this.title = title;
  }

  String getFront() {
    return front;
  }

  String getBack() {
    return back;
  }

  String getTitle() {
    return title;
  }
}
```

CardScreen.dart

```
//This is the screen on which the chosen card is displayed.
//It utilizes the flip_card library to display the front and back image
//of a specific card when the image is tapped

import 'package:flutter/material.dart';
import 'package:flip_card/flip_card.dart';
import 'package:bainisteoir_beag/CardItem.dart';

class CardScreen extends StatefulWidget {
  final CardItem currentCard;

  CardScreen(this.currentCard);

  @override
  State<StatefulWidget> createState() {
    return CardScreenState(this.currentCard);
  }
}

class CardScreenState extends State<CardScreen> {
  CardItem currentCard;

  CardScreenState(this.currentCard);

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.only(
                left: 20.0, top: 0.0, right: 0.0, bottom: 0.0),
            child: Center(
              child: Container(
                width: MediaQuery.of(context).size.width,
                height: MediaQuery.of(context).size.height,
                child: FlipCard(
                  direction: FlipDirection.VERTICAL, // default
                  front: MaterialApp(
                    home: Scaffold(
                      appBar: AppBar(
                        leading: new IconButton(
                          icon:
                              new Icon(Icons.arrow_back, color: Colors.purple),
                          onPressed: () => Navigator.of(context).pop(),
                        ),
                        title: Text(
                          currentCard.getTitle(),
                          style: TextStyle(fontSize: 50),
                        ),
                      ),
                      body: Image.asset(
                        currentCard.getFront(),
                        fit: BoxFit.fill,
                        height: double.infinity,
                        width: double.infinity,
                        alignment: Alignment.center,
                      ), //   <-- image
                    ),
                  ),
```

```
                  back: MaterialApp(
                    home: Scaffold(
                      appBar: AppBar(
                        leading: new IconButton(
                          icon:
                              new Icon(Icons.arrow_back, color: Colors.purple),
                          onPressed: () => Navigator.of(context).pop(),
                        ),
                        title: Text(currentCard.getTitle(),
                            style: TextStyle(fontSize: 50)),
                      ),
                      body: Image.asset(
                        currentCard.getBack(),
                        fit: BoxFit.fill,
                        height: double.infinity,
                        width: double.infinity,
                        alignment: Alignment.center,
                      ), //   <-- image
                    ),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

cardsList.json

This is a json file stored in the local assets folder. It stores the title, front and back values for each card in the base set to ensure that it may always be available, even when the device is offline.


```
{
  "Cards": [
    {
      "title": "Ball Chest Push",
      "front": "ball_chest_push_front.png",
      "back": "ball_chest_push_back.png"
    },
    {
      "title": "Brain Fitness",
      "front": "brain_fitness_front.png",
      "back": "brain_fitness_back.png"
    },
    {
      "title": "Chalk Games",
      "front": "chalk_games_front.png",
      "back": "chalk_games_back.png"
    },
    {
      "title": "Continuous Cricket",
      "front": "continuous_cricket_front.png",
      "back": "continuous_cricket_back.png"
    },
    {
      "title": "Copy Cat",
      "front": "copy_cat_front.png",
      "back": "copy_cat_back.png"
    },
    {
      "title": "Hit The Target",
      "front": "hit_the_target_front.png",
      "back": "hit_the_target_back.png"
    },
    {
      "title": "Lumpy Ball",
      "front": "lumpy_ball_front.png",
      "back": "lumpy_ball_back.png"
    },
    {
      "title": "Playground Cleaner",
      "front": "playground_cleaner_front.png",
      "back": "playground_cleaner_back.png"
    },
    {
      "title": "Playground Orienteering",
      "front": "playground_orienteering_front.png",
      "back": "playground_orienteering_front.png"
    },
    {
      "title": "Rainy Day Indoor Card",
      "front": "rainy_day_indoor_card_front.png",
      "back": "rainy_day_indoor_card_back.png"
    },
    {
      "title": "Standing Hop, Step & Jump",
      "front": "standing_hop_step_jump_front.png",
      "back": "standing_hop_step_jump_back.png"
    },
    {
```

```
      "title": "Street Football",
      "front": "street_football_front.png",
      "back": "street_football_back.png"
    },
    {
      "title": "Tag Rugby",
      "front": "tag_rugby_front.png",
      "back": "tag_rugby_back.png"
    },
    {
      "title": "The Flying Javelin",
      "front": "the_flying_javelin_front.png",
      "back": "the_flying_javelin_back.png"
    },
    {
      "title": "The Olympic Rings",
      "front": "the_olympic_rings_front.png",
      "back": "the_olympic_rings_back.png"
    },
    {
      "title": "The OXO Game",
      "front": "the_oxo_game_front.png",
      "back": "the_oxo_game_back.png"
    },
    {
      "title": "The Reactive Racer",
      "front": "the_reactive_racer_front.png",
      "back": "the_reactive_racer_back.png"
    },
    {
      "title": "Three Sticks",
      "front": "three_sticks_front.png",
      "back": "three_sticks_back.png"
    },
    {
      "title": "Treasure Chest",
      "front": "treasure_chest_front.png",
      "back": "treasure_chest_back.png"
    },
    {
      "title": "Zone Ball",
      "front": "zone_ball_front.png",
      "back": "zone_ball_back.png"
    }
  ]
}
```

exerciseCards.json

Similar to cardsList.json, attributes of the Exercise cards are stored here for access offline as they are part of the base set of cards also.


```
{
  "Cards": [
    {
      "title": "Full Body: Back Planks",
      "image": "full-body-back-planks.png"
    },
    {
      "title": "Full Body: Bear Walk",
      "image": "full-body-bear-walk.png"
    },
    {
      "title": "Full Body: Box Steps",
      "image": "full-body-box-steps.png"
    },
    {
      "title": "Full Body: Bridge",
      "image": "full-body-bridge.png"
    },
    {
      "title": "Full Body: Bunny Hops",
      "image": "full-body-bunny-hops.png"
    },
    {
      "title": "Full Body: Burpees",
      "image": "full-body-burpees.png"
    },
    {
      "title": "Full Body: Crab Crawl",
      "image": "full-body-crab-crawl.png"
    },
    {
      "title": "Full Body: Donkey Kicks",
      "image": "full-body-donkey-kicks.png"
    },
    {
      "title": "Full Body: Floss Dance",
      "image": "full-body-floss-dance.png"
    },
    {
      "title": "Full Body: Forward Planks",
      "image": "full-body-forward-planks.png"
    },
    {
      "title": "Full Body: Frog Jump Squats",
      "image": "full-body-frog-jump-squats.png"
    },
    {
      "title": "Full Body: Grapevine",
      "image": "full-body-grapevine.png"
    },
    {
      "title": "Full Body: Hoop Jumps",
      "image": "full-body-hoop-jumps.png"
    },
    {
```

```
      "title": "Full Body: Jumping Jacks",
      "image": "full-body-jumping-jacks.png"
    },
    {
      "title": "Full Body: Side Shuffle",
      "image": "full-body-side-shuffle.png"
    },
    {
      "title": "Full Body: Skipping",
      "image": "full-body-skipping.png"
    },
    {
      "title": "Full Body: Snake Crawl",
      "image": "full-body-snake-crawl.png"
    },
    {
      "title": "Full Body: Splits",
      "image": "full-body-splits.png"
    },
    {
      "title": "Lower Body: Frog Squats",
      "image": "lower-body-frog-squats.png"
    },
    {
      "title": "Lower Body: Heel and Toe & Lunge",
      "image": "lower-body-heel-toe-lunge.png"
    },
    {
      "title": "Lower Body: Side Lunge",
      "image": "lower-body-side-lunge.png"
    },
    {
      "title": "Lower Body: Single Leg Lift & Balance",
      "image": "lower-body-single-leg-lift-and-balance.png"
    },
    {
      "title": "Upper Body: Arm Circles",
      "image": "upper-body-arm-circles"
    },
    {
      "title": "Upper Body: Back Stroke",
      "image": "upper-body-back-stroke.png"
    },
    {
      "title": "Upper Body: Front Crawl",
      "image": "upper-body-front-crawl.png"
    },
    {
      "title": "Upper Body: Good Morning Stretch",
      "image": "upper-body-good-morning-stretch.png"
    },
    {
      "title": "Upper Body: Press Ups",
      "image": "upper-body-press-ups.png"
    },
    {
      "title": "Upper Body: Tricep Dips",
      "image": "upper-body-tricep-dips.png"
    }
  ]
}
```

MainMenu.dart

```
import 'package:bainisteoir_beag/ActivityMenu.dart';
import 'package:bainisteoir_beag/ExercisesMenu.dart';
import 'package:bainisteoir_beag/NewCardsMenu.dart';
import 'package:bainisteoir_beag/ManageRotaScreen.dart';
import 'package:bainisteoir_beag/TutorialScreen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_staggered_grid_view/flutter_staggered_grid_view.dart';

//This is the main menu of the application, where users may navigate to other main
//screens and features
class MainMenu extends StatefulWidget {
  @override
  _MainMenuState createState() => _MainMenuState();
}

class _MainMenuState extends State<MainMenu> {
  Material menuItem(IconData icon, String title, Color color, var route) {
    return Material(
        color: Colors.yellow[300],
        elevation: 10.0,
        shadowColor: Colors.amber[500],
        borderRadius: BorderRadius.circular(24.0),
        child: InkWell(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => route),
              );
            },
            child: Center(
                child: Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: <Widget>[
                        Column(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: <Widget>[
                            Row(
                              children: <Widget>[
                                Material(
                                    color: color,
                                    borderRadius: BorderRadius.circular(24.0),
                                    child: Padding(
                                      padding: const EdgeInsets.all(16.0),
                                      child: Icon(icon,
                                          color: Colors.white, size: 40.0),
                                    )),
                                Padding(
                                  padding: const EdgeInsets.all(8.0),
                                  child: Text(
                                    title,
                                    style: TextStyle(
                                      color: color,
                                      fontSize: 30.0,
                                    ),
                                  ),
                                ),
                              ],
```

```
                            )
                          ],
                        )
                      ],
                    )))));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: Colors.blue[300],
        body: Padding(
          padding: const EdgeInsets.all(30.0),
          child: StaggeredGridView.count(
            crossAxisCount: 2,
            crossAxisSpacing: 12.0,
            mainAxisSpacing: 12.0,
            shrinkWrap: true,
            padding: EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
            children: <Widget>[
              menuItem(Icons.directions_run, "Activities", Colors.deepPurple,
                  ActivityMenu()),
              menuItem(Icons.accessibility_new, "Exercises", Colors.deepPurple,
                  ExercisesMenu()),
              menuItem(Icons.announcement, "New Cards", Colors.deepPurple,
                  NewMenu()),
              menuItem(
                  Icons.help, "Tutorial", Colors.deepPurple, TutorialScreen()),
              menuItem(Icons.assignment_ind, "Manage ROTA", Colors.deepPurple,
                  ManageRotaScreen()),
            ],
            staggeredTiles: [
              StaggeredTile.extent(2, 130.0),
              StaggeredTile.extent(2, 130.0),
              StaggeredTile.extent(2, 130.0),
              StaggeredTile.extent(1, 150.0),
              StaggeredTile.extent(1, 150.0),
            ],
          ),
        ));
  }
}
```

ManageRotaScreen.dart

```
import 'package:bainisteoir_beag/RotaDB.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'dart:async';
import 'AuthFunctions.dart';
import 'RotaItem.dart';

//This is the screen where the ROTA can be managed, that is,
//New names may be added and existing names may be removed.
class ManageRotaScreen extends StatefulWidget {
  @override
  _ManageRotaScreenState createState() => _ManageRotaScreenState();
}

class _ManageRotaScreenState extends State<ManageRotaScreen> {
  String addNameText = '';

  RotaDB db = RotaDB();
  AuthFunctions _auth = AuthFunctions();

//----------DB Functions-------------------------
  _addName(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.addName(result.uid, name, chosen);
  }

  _removeName(name) async {
    dynamic result = await _auth.signInAnon();
    db.removeName(result.uid, name);
  }

  Future _getItems() async {
    dynamic result = await _auth.signInAnon();

    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    List<RotaItem> items = [];

    for (var u in qn.documents) {
      RotaItem currentItem = RotaItem(u["uid"], u["name"], u["chosen"]);
      //Add Current card to list
      if (currentItem.uid == result.uid) {
        items.add(currentItem);
      }
    }

    //Sort Names Alphabetically
    items.sort((RotaItem a, RotaItem b) {
      return a.name.toLowerCase().compareTo(b.name.toLowerCase());
    });

    return items;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
        backgroundColor: Colors.blue[300],
        appBar: AppBar(
          leading: new IconButton(
            icon: new Icon(Icons.arrow_back, color: Colors.purple),
            onPressed: () => Navigator.of(context).pop(),
          ),
          title: Text('Manage ROTA', style: TextStyle(fontSize: 50)),
          centerTitle: true,
          backgroundColor: Colors.blue,
        ),
        body: SingleChildScrollView(
            child: Column(
          mainAxisAlignment: MainAxisAlignment.start,
          children: <Widget>[
            Container(
                padding: EdgeInsets.all(40.0),
                constraints: BoxConstraints(maxHeight: 500.0),
                child: FutureBuilder(
                    future: _getItems(),
                    builder: (BuildContext context, AsyncSnapshot snapshot) {
                      if (snapshot.data == null) {
                        return Container(
                            child: Center(child: Text("Loading...")));
                      } else {
                        return ListView.builder(
                          itemCount: snapshot.data.length,
                          itemBuilder: (BuildContext context, int index) {
                            return Padding(
                              padding: const EdgeInsets.all(8.0),
                              child: MaterialButton(
                                height: 100,
                                minWidth: 500,
                                color: Colors.yellow[300],
                                elevation: 14.0,
                                shape: RoundedRectangleBorder(
                                    borderRadius:
                                    BorderRadius.all(Radius.circular(20.0))),
                                onPressed: (){},
                                child: Row(
                                  mainAxisAlignment:
                                      MainAxisAlignment.spaceBetween,
                                  children: <Widget>[
                                    Text(snapshot.data[index].name,
                                        style: TextStyle(fontSize: 48)),
                                    Padding(
                                      padding:
                                          const EdgeInsets.fromLTRB(0, 0, 0, 0),
                                      child: MaterialButton(
                                          height: 100,
                                          minWidth: 100,
                                          color: Colors.grey,
                                          onPressed: () async {
                                            await _removeName(
                                                snapshot.data[index].name);
                                            await new Future.delayed(
                                                const Duration(
                                                    milliseconds: 500));
                                            setState(() {});
                                            print("State set");
                                          },
                                          child: Column(
                                            children: <Widget>[
                                              Text("Remove Name",
```

```
                                        style:
                                            TextStyle(fontSize: 30)),
                                    Icon(Icons.delete_forever,
                                        size: 60.0,
                                        color: Colors.red[800]),
                                ],
                            )),
                        ),
                    ],
                ),
            ),
        );
    },
  );
  }
 })),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    SizedBox(
        height: 70.0,
        width: 300.0,
        child: TextField(
          style: TextStyle(fontSize: 20),
          onChanged: (val) {
            setState(() => addNameText = val);
          },
          decoration: new InputDecoration(
            fillColor: Colors.white,
            alignLabelWithHint: true,
            hintText: 'Enter Name',
            labelText: 'Enter Name',
            border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(10.0)),
          ),
        )),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: MaterialButton(
        height: 80,
        minWidth: 100,
        color: Colors.yellow[300],
        elevation: 14.0,
        shape: RoundedRectangleBorder(
            borderRadius:
            BorderRadius.all(Radius.circular(20.0))),
        onPressed: () async {

          if(addNameText != "") {
            _addName(addNameText, false);
            await new Future.delayed(
                const Duration(milliseconds: 500));
            setState(() {});
            print("State set");
          }
        },
        child: Text(
          "Add Name",
          style: TextStyle(
            fontSize: 30,
          ),
        ),
      ),
```

```
                    )
                ],
            ),
        ],
    ));
  }
}
```

ManageRotaTest.dart

```dart
import 'package:bainisteoir_beag/ManageRotaScreen.dart';
import 'package:flutter/material.dart';
import 'package:bainisteoir_beag/RotaDB.dart';
import 'AuthFunctions.dart';

//This screen is unused in the application but was created to test that the
//functions created in RotaDB were functioning correctly

class ManageRotaTest extends StatefulWidget {
  @override
  _ManageRotaTestState createState() => _ManageRotaTestState();
}

class _ManageRotaTestState extends State<ManageRotaTest> {
  //Text Field State
  String textFieldName = '';

  RotaDB db = RotaDB();
  AuthFunctions _auth = AuthFunctions();

  _addName(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.addName(result.uid, name, chosen);
  }

  _printAll() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNames(result.uid);
  }

  _printAllNotChosen() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNamesNotChosen(result.uid);
  }

  _printAllChosen() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNamesChosen(result.uid);
  }

  _setChosen(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.setChosen(result.uid, name, chosen);
  }

  _setChosenAll(chosen) async {
    dynamic result = await _auth.signInAnon();
    db.setChosenAll(result.uid, chosen);
  }

  _removeName(name) async {
    dynamic result = await _auth.signInAnon();
    db.removeName(result.uid, name);
  }

  _removeAll() async {
    dynamic result = await _auth.signInAnon();
    db.removeAllNames(result.uid);
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Manage ROTA'),
      centerTitle: true,
      backgroundColor: Colors.blue,
    ),
    body: SingleChildScrollView(
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
          Column(mainAxisAlignment: MainAxisAlignment.start, children: <
              Widget>[
            Text(
              "Manage Rota Menu",
              style: TextStyle(fontSize: 24.0),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: MaterialButton(
                onPressed: () {
                  _printAll();
                },
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.all(Radius.circular(20.0))),
                height: 80,
                minWidth: 200,
                color: Colors.deepPurple,
                child: Text(
                  "Print All",
                  style: TextStyle(fontSize: 20.0),
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: MaterialButton(
                onPressed: () {
                  _printAllNotChosen();
                },
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.all(Radius.circular(20.0))),
                height: 80,
                minWidth: 200,
                color: Colors.deepPurple,
                child: Text(
                  "Print All Not Chosen",
                  style: TextStyle(fontSize: 20.0),
                ),
              ),
            ),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _printAllChosen();
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
```

```
          height: 80,
          minWidth: 200,
          color: Colors.deepPurple,
          child: Text(
            "Print All Chosen",
            style: TextStyle(fontSize: 20.0),
          ),
        )),
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: Container(
          child: SizedBox(
              height: 50.0,
              width: 300.0,
              child: TextFormField(
                onChanged: (val) {
                  setState(() => textFieldName = val);
                },
                decoration: new InputDecoration(
                  hintText: 'Add Name',
                  labelText: 'Add Name',
                  border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10.0)),
                ),
              )),
        ),
      ),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
              _addName(textFieldName, false);
            },
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(20.0))),
            height: 80,
            minWidth: 200,
            color: Colors.deepPurple,
            child: Text(
              "Add Name",
              style: TextStyle(fontSize: 20.0),
            ),
          )),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
              _setChosen(textFieldName, true);
            },
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(20.0))),
            height: 80,
            minWidth: 200,
            color: Colors.deepPurple,
            child: Text(
              "Set Chosen",
              style: TextStyle(fontSize: 20.0),
            ),
          )),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
```

```
                    _setChosenAll(true);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Set All True",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _setChosenAll(false);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Set All False",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _removeName(textFieldName);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Remove Selected",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _removeAll();
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Remove All Names",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
```

```
                    onPressed: () {
                      Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => ManageRotaScreen()));
                    },
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.all(Radius.circular(20.0))),
                    height: 80,
                    minWidth: 200,
                    color: Colors.deepPurple,
                    child: Text(
                      "Rota Screen",
                      style: TextStyle(fontSize: 20.0),
                    ),
                  )),
            ])
          ])),
    );
  }
}
```

NewCardScreen.dart

```
import 'package:flutter/material.dart';
import 'package:flip_card/flip_card.dart';
import 'package:bainisteoir_beag/CardItem.dart';

// This screen functions similarly to CardScreen.dart, except that the images
// are loaded in as network images (from the Firebase Storage bucket) instead of
// from local assets.
class NewCardScreen extends StatefulWidget {
  final CardItem currentCard;

  NewCardScreen(this.currentCard);

  @override
  State<StatefulWidget> createState() {
    return NewCardScreenState(this.currentCard);
  }
}

class NewCardScreenState extends State<NewCardScreen> {

  // This function creates a url based on how firebase storage viewable urls are
  // named. The static parts of the url are concatenated with the file name passed
into the
  // function, effectively "guessing" the url for the images uploaded to the storage
bucket.
  static String constructUrl(var image) {
    String first =
        "https://firebasestorage.googleapis.com/v0/b/bainisteoir-
beag.appspot.com/o/cards%2F";
    String second = image;
    String third = "?alt=media";

    String finalUrl = first + second + third;

    return finalUrl;
  }

  CardItem currentCard;

  NewCardScreenState(this.currentCard);

  @override
  Widget build(BuildContext context) {
    String frontUrl = constructUrl(currentCard.getFront());
    String backUrl = constructUrl(currentCard.getBack());
    return Container(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.only(
                left: 20.0, top: 0.0, right: 0.0, bottom: 0.0),
            child: Center(
              child: Container(
                width: MediaQuery.of(context).size.width,
                height: MediaQuery.of(context).size.height,
                child: FlipCard(
                  direction: FlipDirection.VERTICAL, // default
                  front: MaterialApp(
                    home: Scaffold(
                      appBar: AppBar(
```

```
                        leading: new IconButton(
                          icon:
                              new Icon(Icons.arrow_back, color: Colors.purple),
                          onPressed: () => Navigator.of(context).pop(),
                        ),
                        title: Text(currentCard.getTitle()),
                      ),
                      body: Image.network(
                        frontUrl,
                        fit: BoxFit.fill,
                        height: double.infinity,
                        width: double.infinity,
                        alignment: Alignment.center,
                      ), //   <-- image
                    ),
                  ),
                  back: MaterialApp(
                    home: Scaffold(
                      appBar: AppBar(
                        leading: new IconButton(
                          icon:
                              new Icon(Icons.arrow_back, color: Colors.purple),
                          onPressed: () => Navigator.of(context).pop(),
                        ),
                        title: Text(currentCard.getTitle()),
                      ),
                      body: Image.network(
                        backUrl,
                        fit: BoxFit.fill,
                        height: double.infinity,
                        width: double.infinity,
                        alignment: Alignment.center,
                      ), //   <-- image
                    ),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

NewCardsMenu.dart

```
import 'package:bainisteoir_beag/ManageRotaScreen.dart';
import 'package:flutter/material.dart';
import 'package:bainisteoir_beag/RotaDB.dart';
import 'AuthFunctions.dart';

//This screen is unused in the application but was created to test that the
//functions created in RotaDB were functioning correctly

class ManageRotaTest extends StatefulWidget {
  @override
  _ManageRotaTestState createState() => _ManageRotaTestState();
}

class _ManageRotaTestState extends State<ManageRotaTest> {
  //Text Field State
  String textFieldName = '';

  RotaDB db = RotaDB();
  AuthFunctions _auth = AuthFunctions();

  _addName(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.addName(result.uid, name, chosen);
  }

  _printAll() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNames(result.uid);
  }

  _printAllNotChosen() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNamesNotChosen(result.uid);
  }

  _printAllChosen() async {
    dynamic result = await _auth.signInAnon();
    print(result.uid);
    db.getNamesChosen(result.uid);
  }

  _setChosen(name, chosen) async {
    dynamic result = await _auth.signInAnon();
    db.setChosen(result.uid, name, chosen);
  }

  _setChosenAll(chosen) async {
    dynamic result = await _auth.signInAnon();
    db.setChosenAll(result.uid, chosen);
  }

  _removeName(name) async {
    dynamic result = await _auth.signInAnon();
    db.removeName(result.uid, name);
  }

  _removeAll() async {
    dynamic result = await _auth.signInAnon();
    db.removeAllNames(result.uid);
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Manage ROTA'),
      centerTitle: true,
      backgroundColor: Colors.blue,
    ),
    body: SingleChildScrollView(
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
          Column(mainAxisAlignment: MainAxisAlignment.start, children: <
              Widget>[
            Text(
              "Manage Rota Menu",
              style: TextStyle(fontSize: 24.0),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: MaterialButton(
                onPressed: () {
                  _printAll();
                },
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.all(Radius.circular(20.0))),
                height: 80,
                minWidth: 200,
                color: Colors.deepPurple,
                child: Text(
                  "Print All",
                  style: TextStyle(fontSize: 20.0),
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: MaterialButton(
                onPressed: () {
                  _printAllNotChosen();
                },
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.all(Radius.circular(20.0))),
                height: 80,
                minWidth: 200,
                color: Colors.deepPurple,
                child: Text(
                  "Print All Not Chosen",
                  style: TextStyle(fontSize: 20.0),
                ),
              ),
            ),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _printAllChosen();
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
```

```
          height: 80,
          minWidth: 200,
          color: Colors.deepPurple,
          child: Text(
            "Print All Chosen",
            style: TextStyle(fontSize: 20.0),
          ),
        )),
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: Container(
          child: SizedBox(
              height: 50.0,
              width: 300.0,
              child: TextFormField(
                onChanged: (val) {
                  setState(() => textFieldName = val);
                },
                decoration: new InputDecoration(
                  hintText: 'Add Name',
                  labelText: 'Add Name',
                  border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(10.0)),
                ),
              )),
        ),
      ),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
              _addName(textFieldName, false);
            },
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(20.0))),
            height: 80,
            minWidth: 200,
            color: Colors.deepPurple,
            child: Text(
              "Add Name",
              style: TextStyle(fontSize: 20.0),
            ),
          )),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
              _setChosen(textFieldName, true);
            },
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.all(Radius.circular(20.0))),
            height: 80,
            minWidth: 200,
            color: Colors.deepPurple,
            child: Text(
              "Set Chosen",
              style: TextStyle(fontSize: 20.0),
            ),
          )),
      Padding(
          padding: const EdgeInsets.all(8.0),
          child: MaterialButton(
            onPressed: () {
```

```
                    _setChosenAll(true);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Set All True",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _setChosenAll(false);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Set All False",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _removeName(textFieldName);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Remove Selected",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
                  onPressed: () {
                    _removeAll();
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(20.0))),
                  height: 80,
                  minWidth: 200,
                  color: Colors.deepPurple,
                  child: Text(
                    "Remove All Names",
                    style: TextStyle(fontSize: 20.0),
                  ),
                )),
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: MaterialButton(
```

```
                    onPressed: () {
                      Navigator.of(context).push(MaterialPageRoute(
                          builder: (context) => ManageRotaScreen()));
                    },
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.all(Radius.circular(20.0))),
                    height: 80,
                    minWidth: 200,
                    color: Colors.deepPurple,
                    child: Text(
                      "Rota Screen",
                      style: TextStyle(fontSize: 20.0),
                    ),
                  )),
            ])
          ])),
    );
  }
}
```

RotaDB.dart

This class contains a collection of functions specific to the management of the ROTA system, and interacts with the relevant FireStore database.

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

class RotaDB {
  Firestore firestore = Firestore.instance;

  //Add name to database
  addName(uid, name, chosen) {
    Firestore.instance.runTransaction((transaction) async {
      await transaction
          .set(Firestore.instance.collection("rotaNames").document(), {
        'uid': uid,
        'name': name,
        'chosen': chosen,
      });
    });
    print("Document added");
  }

  //Return all names with specified User Id
  Future getNames(uid) async {
    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    List<RotaItem> names = [];

    for (var u in qn.documents) {
      RotaItem currentItem = RotaItem(u["uid"], u["name"], u["chosen"]);
      //Add Current card to list

      //Add Only names from current user to list
      if (uid == u["uid"]) {
        names.add(currentItem);
      }
    }

    for (var u in names) {
      print(u);
    }

    print("names in set: " + (names.length).toString());

    return names;
  }

  //Return all names that have not been chosen already
  Future getNamesNotChosen(uid) async {
    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    List<RotaItem> names = [];

    for (var u in qn.documents) {
      RotaItem currentItem = RotaItem(u["uid"], u["name"], u["chosen"]);
```

```
    //Add Current card to list

    //Add Only names from current user to list
    if (uid == u["uid"] && u["chosen"] == false) {
      names.add(currentItem);
    }
  }

  for (var u in names) {
    print(u);
  }

  print("names in set: " + (names.length).toString());

  return names;
}

//Return all names that have been chosen already
Future getNamesChosen(uid) async {
  Firestore firestore;
  firestore = Firestore.instance;

  var qn = await firestore.collection("rotaNames").getDocuments();

  List<RotaItem> names = [];

  for (var u in qn.documents) {
    RotaItem currentItem = RotaItem(u["uid"], u["name"], u["chosen"]);
    //Add Only names from current user to list
    if (uid == u["uid"] && u["chosen"] == true) {
      names.add(currentItem);
    }
  }

  for (var u in names) {
    print(u);
  }

  print("names in set: " + (names.length).toString());

  return names;
}

//Set value of Chosen for specific document
Future setChosen(uid, name, chosen) async {
  Firestore firestore;
  firestore = Firestore.instance;

  var qn = await firestore.collection("rotaNames").getDocuments();

  List<RotaItem> names = [];

  //For each document in rotaNames
  for (var u in qn.documents) {
    //If Uid and Name match, change value of 'chosen'
    if (uid == u["uid"] && name == u["name"]) {
      Firestore.instance
          .collection("rotaNames")
          .document(u.documentID)
          .updateData(
              {"uid": uid, "name": name, "chosen": chosen}).catchError((e) {
        print(e);
      });
```

```
      }
    }
    for (var u in names) {
      print(u);
    }

    print("Document updated");

    return names;
  }

//Set value of Chosen for specific document
  Future setChosenAll(uid, chosen) async {
    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    List<RotaItem> names = [];

    //For each document in rotaNames

    for (var u in qn.documents) {
      if (uid == u["uid"]) {
        Firestore.instance
            .collection("rotaNames")
            .document(u.documentID)
            .updateData({
          "uid": u["uid"],
          "name": u["name"],
          "chosen": chosen
        }).catchError((e) {
          print(e);
        });
      }
    }
    for (var u in names) {
      print(u);
    }

    print("All documents updated");

    return names;
  }

  //Remove specific document by name
  Future removeName(uid, name) async {
    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    //For each document in rotaNames
    for (var u in qn.documents) {
      if (uid == u["uid"] && name == u["name"]) {
        Firestore.instance
            .collection("rotaNames")
            .document(u.documentID)
            .delete()
            .catchError((e) {
          print(e);
        });
      }
```

```
    }
    print("Document Deleted: $name");
  }

  //Remove All documents with matching Uid
  Future removeAllNames(uid) async {
    Firestore firestore;
    firestore = Firestore.instance;

    var qn = await firestore.collection("rotaNames").getDocuments();

    //For each document in rotaNames
    for (var u in qn.documents) {
      if (uid == u["uid"]) {
        Firestore.instance
            .collection("rotaNames")
            .document(u.documentID)
            .delete()
            .catchError((e) {
          print(e);
        });
      }
    }
    print("All Documents Removed");
  }
}
```

RotaItem.dart

```
// This class represents an object which will be stored in the rotaNames
// FireStore Database and managed by the application.
// Each RotaItem has three attributes as follows:
// uid:    the user id, linked to the current device. This is used when polling the
//         database to ensure that only names entered on the current device will be
//         viewable and manageable.
// name:   A string representing the name of a user. This will be displayed when
//         managing the ROTA and also when a leader is being chosen via the button
//         in the card menus.
// chosen: A boolean value to denote whether a name has been chosen already in the
//         card menus. This helps to enforce a system where every potential leader
//         will get a turn and is not at the mercy of a completely random selection
//         each time.
//         ie. The pool of potential leaders will only include RotaItems which have
//         a chosen value of false. Once a RotaItem is selected by the random
//         generator, its chosen value is updated to true, and it will not be chosen
//         again until the pool is exhausted and the chosen values are reset to false.

class RotaItem {
  String uid;
  String name;
  bool chosen;

  RotaItem(uid, name, chosen) {
    this.uid = uid;
    this.name = name;
    this.chosen = chosen;
  }

  void setUid(String uid) {
    this.uid = uid;
  }

  void setName(String name) {
    this.name = name;
  }

  void setChosen(bool chosen) {
    this.chosen = chosen;
  }

  String getUid() {
    return uid;
  }

  String getName() {
    return name;
  }

  bool getChosen() {
    return chosen;
  }
}
```

SplashScreen.dart

```
//This Screen is the first to be displayed when the application is opened.
//Simply, it displays an image that, when tapped, will navigate to the main menu

import 'package:bainisteoir_beag/MainMenu.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  Widget build(BuildContext context) {
    return Material(
        child: InkWell(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => MainMenu()),
              );
            },
            child: Image.asset('assets/SplashScreenBlue.png',
                fit: BoxFit.fill,
                height: double.infinity,
                width: double.infinity,
                alignment: Alignment.center)));
  }
}
```

TutorialScreen.dart

```
//This Screen functions as a video player for the application's tutorial video
//on how the game is played and how the application is used.

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

class TutorialScreen extends StatefulWidget {
  @override
  _TutorialScreenState createState() => _TutorialScreenState();
}

class _TutorialScreenState extends State<TutorialScreen> {
  VideoPlayerController _controller;
  Future<void> _initialiseVideoPlayer;

  @override
  void initState() {
    _controller = VideoPlayerController.asset("assets/tutorialVideo.mp4");
    _initialiseVideoPlayer = _controller.initialize();
    _controller.setLooping(true);
    _controller.setVolume(1.0);
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.blue[300],
      appBar: AppBar(
        title: Text('Tutorial', style: TextStyle(fontSize: 50)),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Column(
        children: <Widget>[
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
              Container(
                height: 600,
                child: FutureBuilder(
                    future: _initialiseVideoPlayer,
                    builder: (context, snapshot) {
                      if (snapshot.connectionState == ConnectionState.done) {
                        return AspectRatio(
                          aspectRatio: _controller.value.aspectRatio,
                          child: VideoPlayer(_controller),
                        );
                      } else {
                        return Text(
                          "Loading Video",
                          style: TextStyle(color: Colors.deepPurple, fontSize: 70),
                        );
                      }
                    }),
              ),
            ],
          ),
          Row(
              mainAxisAlignment: MainAxisAlignment.center,
```

```
              crossAxisAlignment: CrossAxisAlignment.center,
              children: <Widget>[
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: MaterialButton(
                      color: Colors.yellow[300],
                      elevation: 14.0,
                      shape: RoundedRectangleBorder(
                          borderRadius:
                          BorderRadius.all(Radius.circular(20.0))),
                      height: 40,
                      minWidth: 80,
                      onPressed: () {
                        print("Play Button Pressed");
                        setState(() {
                          _controller.play();
                        });
                      },
                      child: Icon(Icons.play_arrow)),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: MaterialButton(
                      color: Colors.yellow[300],
                      elevation: 14.0,
                      shape: RoundedRectangleBorder(
                          borderRadius:
                          BorderRadius.all(Radius.circular(20.0))),
                      height: 40,
                      minWidth: 80,
                      onPressed: () {
                        print("Pause Button Pressed");
                        setState(() {
                          _controller.pause();
                        });
                      },
                      child: Icon(Icons.pause)),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: MaterialButton(
                      color: Colors.yellow[300],
                      elevation: 14.0,
                      shape: RoundedRectangleBorder(
                          borderRadius:
                          BorderRadius.all(Radius.circular(20.0))),
                      height: 40,
                      minWidth: 80,
                      onPressed: () {
                        print("Restart Button Pressed");
                        setState(() {
                          _controller.initialize();
                        });
                      },
                      child: Icon(Icons.fast_rewind)),
                ),
              ]),
        ],
      ),
    );
  }
}
```

User.dart

```
class User {
  //User object used to contain FirebaseUser information necessary for the app
  final String uid;
  User({this.uid});
}
```

Web Application Source Code

Login.html

A simple login page using the Firebase Auth service to authenticate an already registered user.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1"
        name="viewport">
  <title>Login</title>
  <style media="screen">
        body {
            display: flex;
            min-height: 100vh;
            width: 100%;
            padding: 0;
            margin: 0;
            align-items: center;
            justify-content: center;
            flex-direction: column;
            font-family: Arial, Helvetica, sans-serif;
        }

        h2 {
            text-align: center;
            color: purple;
            text-decoration: none;
            display: inline-block;
            font-size: 26px;
        }

        label,
        input {
            display: block;
            color: purple;
            width: 250px;
        }

        input {
            font-size: 18px;
            padding: 15px;
        }
```

```
        label {
            font-size: 24px;
            margin-bottom: 10px;
        }

        button {
            background-color: purple;
            border: none;
            color: white;
            padding: 25px;
            text-align: center;
            display: inline-block;
            font-size: 16px;
            width: 50%;
            margin-left: 25%;
            margin-right: 25%
        }
    </style>
</head>
<body>
    <script src="/__/firebase/7.14.0/firebase-app.js"></script>
    <script src="/__/firebase/7.14.0/firebase-auth.js"></script>
    <script src="/__/firebase/init.js"></script>
    <div class="login-form">
      <div class="log-form">
        <h2>Bainisteoir Beag Login</h2><br>
        <form>
          <label>Email Address <input id="email"
                 placeholder="Email Address"
                 title="email"
                 type="email"></label> <label>Password<input id="password"
                 placeholder="Password"
                 title="password"
                 type="password"></label> <button class="btn"
                onclick="login()"
                type="button">Login</button>
        </form>
      </div><!--end log form -->
      <script>
            var email = document.getElementById("email");
            var password = document.getElementById("password");
            const auth = firebase.auth();

            function login() {
```

```
                emailVal = email.value;
                passwordVal = password.value;
                firebase.auth().signInWithEmailAndPassword(emailVal, passwordVal)
                    .then(function (response) {
                        console.log("User Logged in: " + email);
                        redirect("upload_card.html");
                    })
                    .catch(function (error) {
                        alert(error.message);
                    });
            }

            function redirect(url) {
      window.location.href = url;
    }

    function wait(ms) {
      var start = new Date().getTime();
      var end = start;
      while (end < start + ms) {
        end = new Date().getTime();
      }
    }

    //Check Auth status and redirect if logged in
    firebase.auth().onAuthStateChanged(function (user) {
         if (user) {
            redirect("upload_card.html");
         } else {
            console.log("user not already logged in");
         }
     });


    </script>
  </div>
</body>
</html>
```

Register_user.html

A simple register page using Firebase Auth to create a user with Email and Password. This screen may only be accessed by an authenticated user. The intended purpose of this page is for an admin to use it in the event that they want to allow the privilege of adding cards to Bainisteoir Beag.

This way, the web app should only be accessible to an admin or any users manually created by an admin.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1"
        name="viewport">
  <title>Register New Admin</title>
  <style media="screen">


    /* Style the links inside the navigation bar */
  .topnav a {
  float: left;
  color: purple;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
  }

  .topnav {
      align-items: baseline;
      justify-content: baseline;
  background-color: rgb(110, 212, 255);
  overflow: hidden;
  }

  /* Change the color of links on hover */
  .topnav a:hover {
  background-color: rgb(192, 236, 255);
  color: purple;
  }

  /* Add a color to the active/current link */
  .topnav a.active {
  background-color: rgb(245, 255, 55);
```

```
color: purple;
}


  body {
      display: flex;
      min-height: 100vh;
      width: 100%;
      padding: 0;
      margin: 0;
      align-items: center;
      justify-content: center;
      flex-direction: column;
      font-family: Arial, Helvetica, sans-serif;
  }

  h2 {
      text-align: center;
      color: purple;
      text-decoration: none;
      display: inline-block;
      font-size: 26px;
  }

  label,
  input {
      display: block;
      color: purple;
      width: 250px;
  }

  input {
      font-size: 18px;
      padding: 15px;
  }

  label {
      font-size: 24px;
      margin-bottom: 10px;
  }

  button {
      background-color: purple;
      border: none;
      color: white;
```

```
        padding: 25px;
        text-align: center;
        display: inline-block;
        font-size: 16px;
        width: 50%;
        margin-left: 25%;
        margin-right: 25%
    }
  </style>
</head>
<body>
  <!-
- Insert these scripts at the bottom of the HTML, but before you use any Firebase
 services -->
  <!-
- Firebase App (the core Firebase SDK) is always required and must be listed firs
t -->
  <script src="/__/firebase/7.14.0/firebase-app.js"></script> <!-
- If you enabled Analytics in your project, add the Firebase SDK for Analytics --
>

  <script src="/__/firebase/7.14.0/firebase-analytics.js"></script> <!-
- Add Firebase products that you want to use -->

  <script src="/__/firebase/7.14.0/firebase-auth.js"></script>
  <script src="/__/firebase/7.14.0/firebase-firestore.js"></script>
  <script src="/__/firebase/init.js"></script>
  <div class="topnav">
    <a href="upload_card.html">Add Card</a> <a class="active"
        href="register_user.html">Add New User</a> <a href="sign_out.html">Sign
Out</a>
  </div>
  <div class="login-form">
    <h2>Register New Admin</h2><label>Email Address <input id="email"
          placeholder="Email Address"
          title="email"
          type="email"></label> <label>Password <input id="password"
          placeholder="Password"
          title="Password"
          type="password"></label> <label>Confirm Password <input id="confirmPas
sword"
          placeholder="Confirm Password"
          title="confirmPassword"
          type="password"></label><button class="btn"
        onclick="register()"
```

```
        type="button">Register</button>
  </div>
  <script>
    var email = document.getElementById("email");
    var password = document.getElementById("password");
    var confirmPassword = document.getElementById("confirmPassword");
    const auth  = firebase.auth();

    function register(){
      var emailValue = email.value;
      var passwordValue = password.value;
      var confirmPasswordValue = confirmPassword.value;

      if(passwordValue == confirmPasswordValue){
      auth.createUserWithEmailAndPassword(emailValue, passwordValue).catch(functi
on(error) {
      var errorCode = error.code;
      var errorMessage = error.message;
      alert(errorMessage);
      });
      wait(1000);
      alert("User created: " + emailValue);
      location.reload();
    }
    else{
      alert("Password must match Confirm Password");
    }
  }

  function redirect(url) {
      window.location.href = url;
    }

    function wait(ms) {
      var start = new Date().getTime();
      var end = start;
      while (end < start + ms) {
        end = new Date().getTime();
      }
    }

    //Check Auth status and redirect if not logged in
     firebase.auth().onAuthStateChanged(function (user) {
         if (user) {
             console.log("Logged In");
```

```
        } else {
           redirect("login.html");
        }
    });


  </script>
</body>
</html>
```

Upload_card.html

This page allows an admin to upload new cards to the set. Once uploaded, the images are stored in a Firebase Storage bucket, and the card attributes are stored in a FireStore database, 'newCards'.

Once uploaded, the new cards may be accessed in the mobile application via the New Cards menu.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1"
        name="viewport">
  <title>Upload Card</title>
  <style media="screen">

    /* Style the links inside the navigation bar */
  .topnav a {
  float: left;
  color: purple;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
  }

  .topnav {
      align-items: baseline;
      justify-content: baseline;
  background-color: rgb(110, 212, 255);
  overflow: hidden;
  }

  /* Change the color of links on hover */
  .topnav a:hover {
  background-color: rgb(192, 236, 255);
  color: purple;
  }

  /* Add a color to the active/current link */
  .topnav a.active {
  background-color: rgb(245, 255, 55);
  color: purple;
  }

    body {
```

```
    display: flex;
    min-height: 100vh;
    width: 100%;
    padding: 0;
    margin: 0;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    font-family: Arial, Helvetica, sans-serif;
  }

  h2 {
    text-align: center;
    color: purple;
    text-decoration: none;
    display: inline-block;
    font-size: 26px;
  }

  label,
  input {
    display: block;
    color: purple;
    width: 250px;
  }

  input {
    font-size: 18px;
    padding: 15px;
  }

  label {
    font-size: 24px;
    margin-bottom: 10px;
  }

  button {
    background-color: purple;
    border: none;
    color: white;
    padding: 25px;
    text-align: center;
    display: inline-block;
    font-size: 16px;
    width: 10%;
```

```
      margin-left: 45%;
      margin-right: 45%
    }


  </style>
</head>
<body>
  <div class="topnav">
    <a class="active"
        href="upload_card.html">Add Card</a> <a href="register_user.html">Add Ne
w User</a> <a href="sign_out.html">Sign Out</a>
  </div>
  <h2>Upload New Card</h2><label>Enter Card Title <input id="cardTitle"
        placeholder="Card Title"
        type="text"></label> <label>Upload Front Image <input id="fileButtonFron
t"
        type="file"
        value="upload"></label> <label>Upload Back Image: <input id="fileButtonB
ack"
        type="file"
        value="upload"></label> <button onclick="addCard()">Save</button>
  <script src="/__/firebase/7.14.0/firebase-app.js"></script>
  <script src="/__/firebase/7.14.0/firebase-auth.js"></script>
  <script src="/__/firebase/7.14.0/firebase-firestore.js"></script>
  <script src="/__/firebase/7.14.0/firebase-storage.js"></script>
  <script src="/__/firebase/init.js"></script>
  <script>


    // Declare elements
    var fileButtonFront = document.getElementById("fileButtonFront");
    var fileButtonBack = document.getElementById("fileButtonBack");
    var saveButton = document.getElementById("saveButton");
    var URL = "gs://bainisteoir-beag.appspot.com/cards/";
    var front = "none";
    var back = "none";
    var cardTitle = "none";

    //Listen for file selection
    fileButtonFront.addEventListener("change", function (e) {
      //Get file
      var file = e.target.files[0];

      //Create a storage ref
```

```
    const storageRef = firebase.storage().ref("cards/" + file.name);
    front = file.name;

    //upload file
    var task = storageRef.put(file);
    alert("Front Image Uploaded");
  });

  fileButtonBack.addEventListener("change", function (e) {
    //Get file
    var file = e.target.files[0];
    //Create a storage ref
    const storageRef = firebase.storage().ref("cards/" + file.name);
    back = file.name;

    //upload file
    var task = storageRef.put(file);

    alert("Back Image Uploaded");
  });

  function addCard() {
    var db = firebase.firestore();
    cardTitle = document.getElementById("cardTitle").value;

    console.log("Save Button Clicked");

    if (front != "none" && back != "none" && cardTitle != "none") {
      db.collection("newCards")
        .doc(cardTitle)
        .set({
          front: front,
          back: back,
          title: cardTitle,
        })
        .then(function () {
          console.log("Document successfully written to Cards.");
          alert("Card Successfully Added.")
          location.reload();
        })
        .catch(function (error) {
          console.error("Error writing document to Cards: ", error);
          alert("Error Adding to Firestore");
        });
    }
```

```
      else {
        alert("You must fill in all forms");
      }
    }

    function redirect(url) {
      window.location.href = url;
    }

    function wait(ms) {
      var start = new Date().getTime();
      var end = start;
      while (end < start + ms) {
        end = new Date().getTime();
      }
    }

     firebase.auth().onAuthStateChanged(function (user) {
         if (user) {
             console.log("Logged In");
         } else {
            redirect("login.html");
         }
     });

  </script>
</body>
</html>
```

Declaration

- I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name: Francis Hall

Student Number: C00220910

Signature: 

Date: 20/04/2020