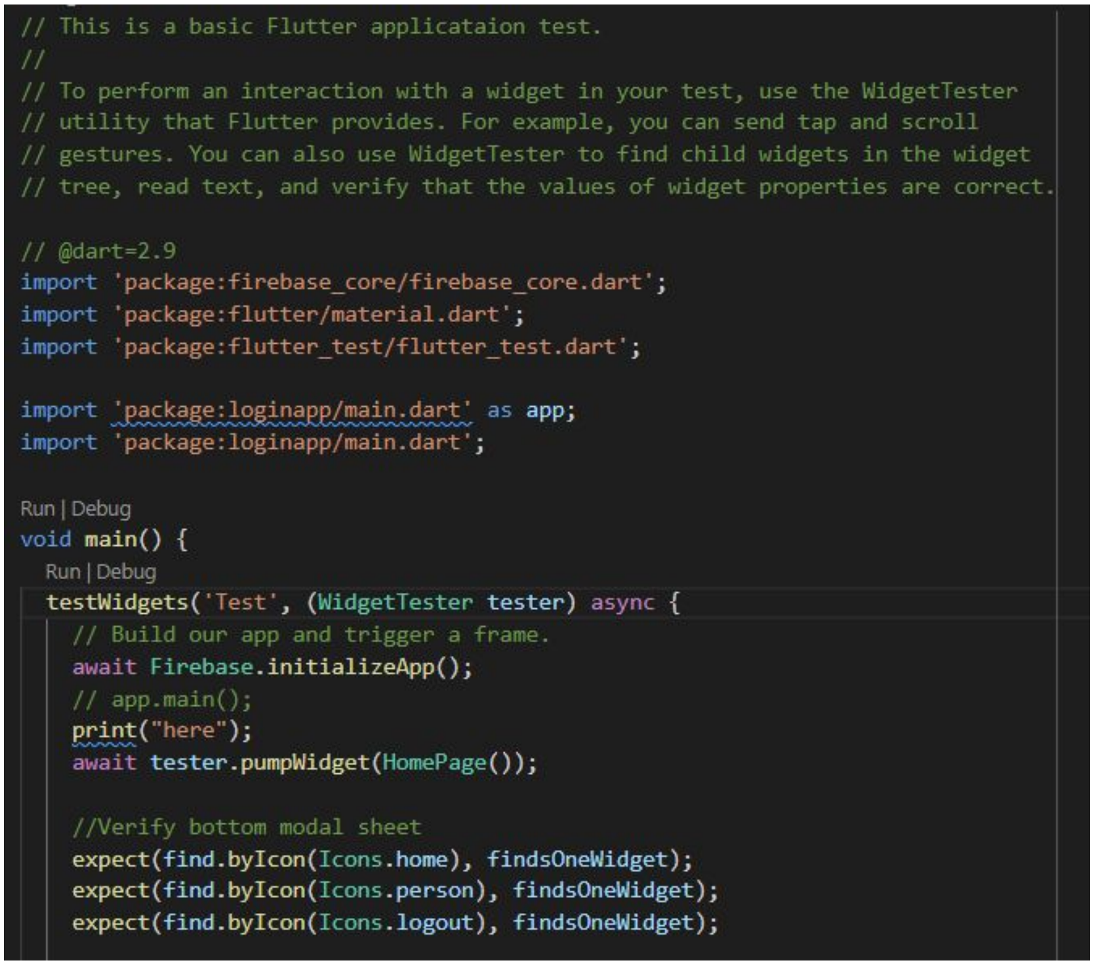
Automated Test Script

- We have created entire application using flutter / dart programming language using Google Firebase and FIrestore as backend to store models and Imgur platform as dynamic image storage.

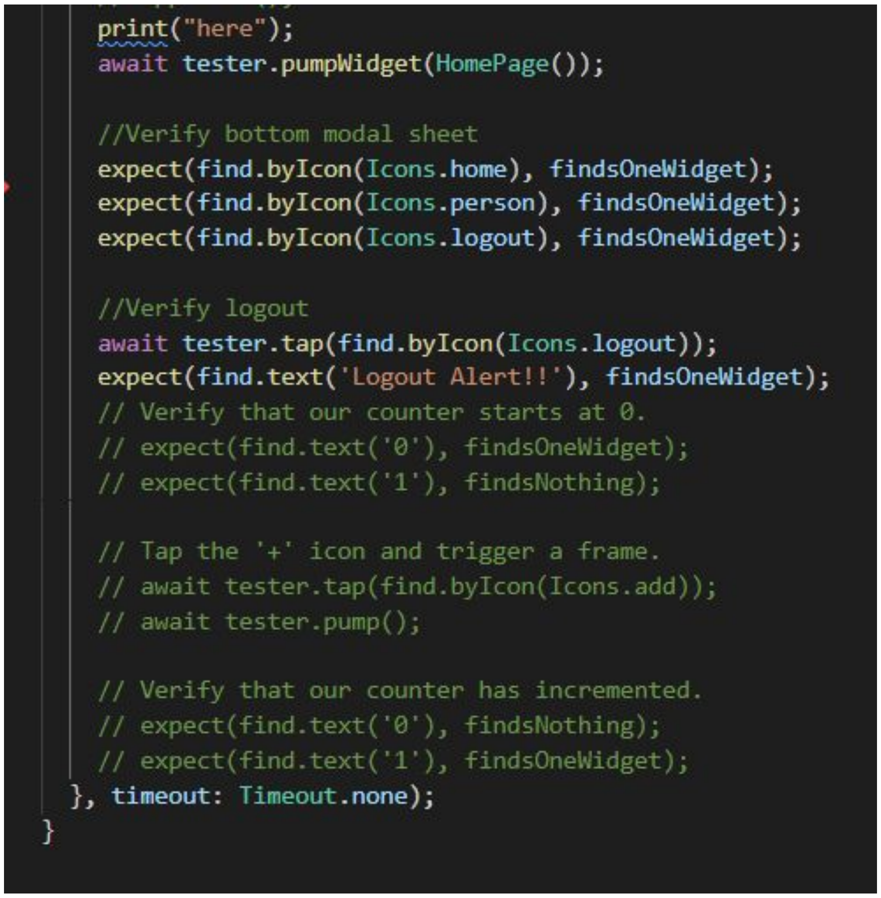
- Because of usage of Google Firebase and Firestore automated testing scripts take a huge amount of time to run, as it waits for the confirmation of the application from the google server, and thousands of applications are running on it.

- Due to this, automated testing scripts of application using google firebase is not possible in major cases, although I have provided the testing script wrote for the application but it takes huge amount of time to run.

- There is a separate platform called “Test Lab” on the firebase wherein apps can be tested.

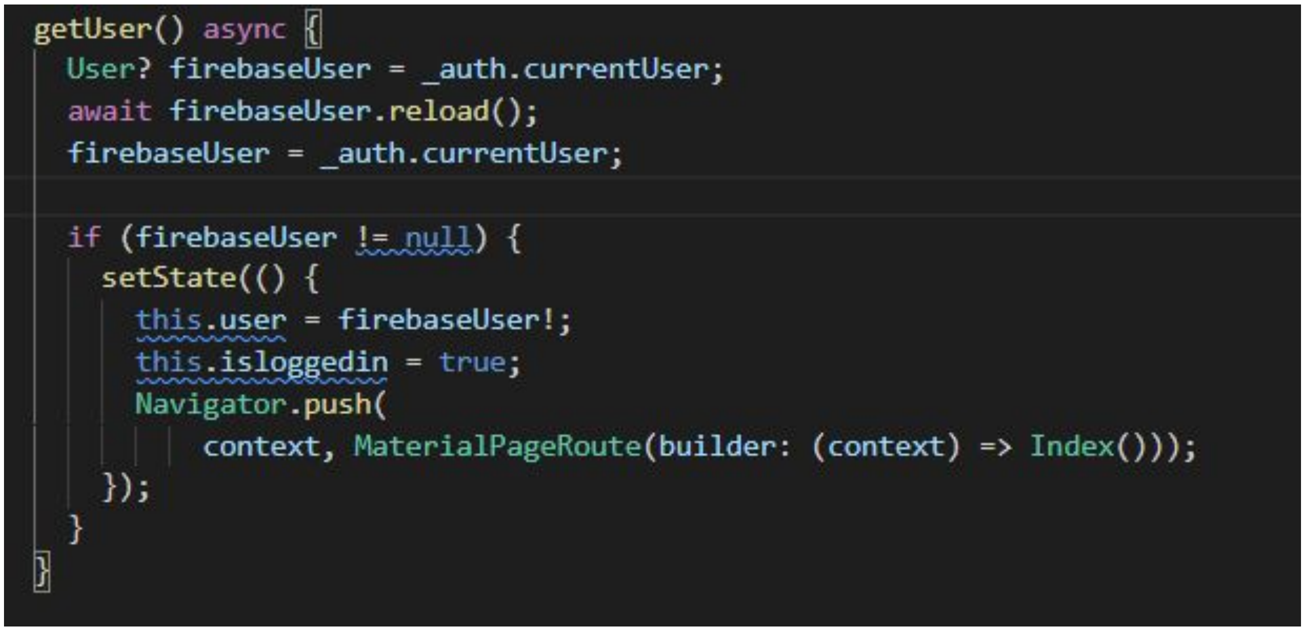


```
// This is a basic Flutter applicataion test.
//
// To perform an interaction with a widget in your test, use the WidgetTester
// utility that Flutter provides. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.

// @dart=2.9
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'package:loginapp/main.dart' as app;
import 'package:loginapp/main.dart';

Run | Debug
void main() {
  Run | Debug
  testWidgets('Test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await Firebase.initializeApp();
    // app.main();
    print("here");
    await tester.pumpWidget(HomePage());

    //Verify bottom modal sheet
    expect(find.byIcon(Icons.home), findsOneWidget);
    expect(find.byIcon(Icons.person), findsOneWidget);
    expect(find.byIcon(Icons.logout), findsOneWidget);
```

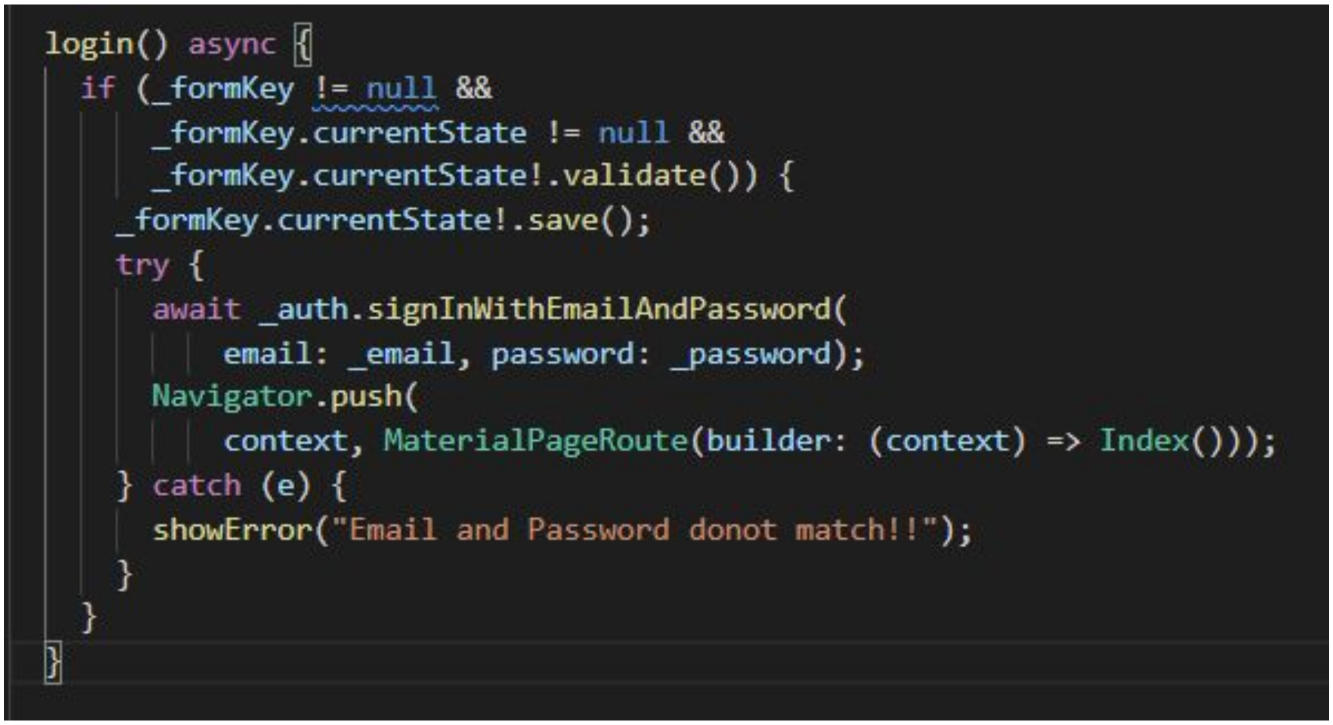


```dart
    print("here");
    await tester.pumpWidget(HomePage());

    //Verify bottom modal sheet
    expect(find.byIcon(Icons.home), findsOneWidget);
    expect(find.byIcon(Icons.person), findsOneWidget);
    expect(find.byIcon(Icons.logout), findsOneWidget);

    //Verify logout
    await tester.tap(find.byIcon(Icons.logout));
    expect(find.text('Logout Alert!!'), findsOneWidget);
    // Verify that our counter starts at 0.
    // expect(find.text('0'), findsOneWidget);
    // expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    // await tester.tap(find.byIcon(Icons.add));
    // await tester.pump();

    // Verify that our counter has incremented.
    // expect(find.text('0'), findsNothing);
    // expect(find.text('1'), findsOneWidget);
  }, timeout: Timeout.none);
}
```

- **Testing Script for the test cases mentioned:**
  - To check whether the user is already logged in or not.

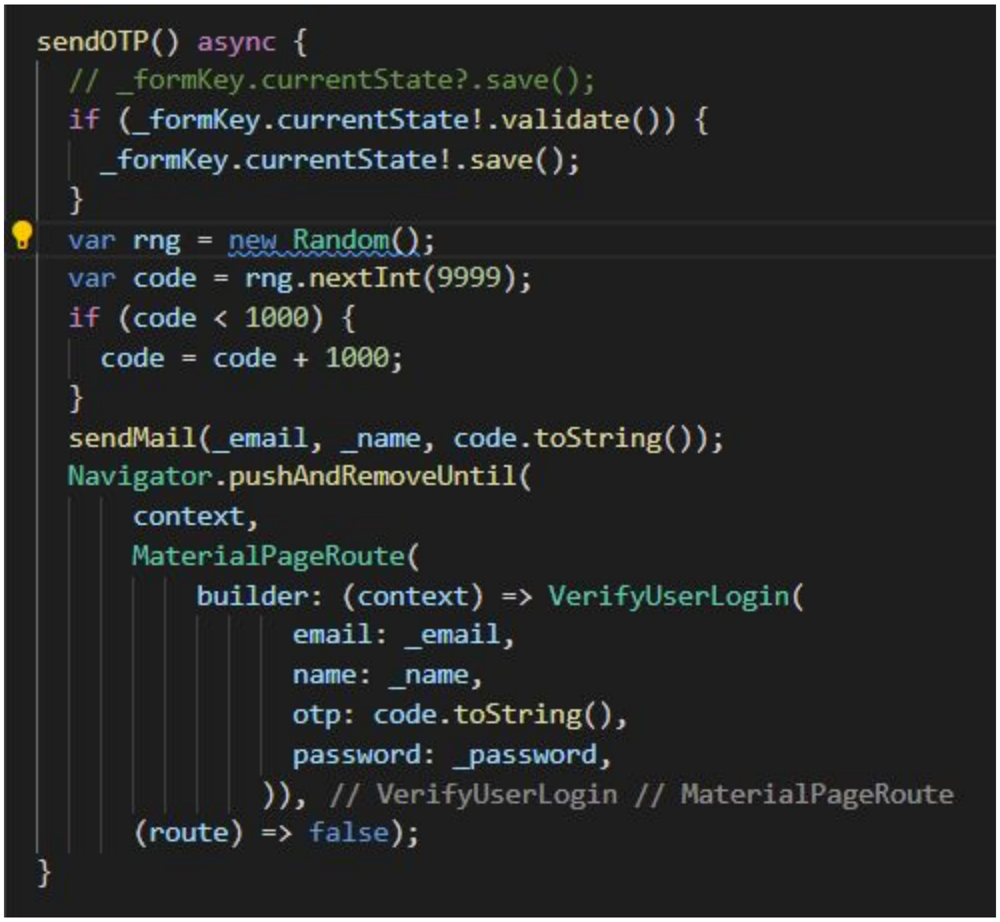


```dart
getUser() async {
  User? firebaseUser = _auth.currentUser;
  await firebaseUser.reload();
  firebaseUser = _auth.currentUser;

  if (firebaseUser != null) {
    setState(() {
      this.user = firebaseUser!;
      this.isloggedin = true;
      Navigator.push(
          context, MaterialPageRoute(builder: (context) => Index()));
    });
  }
}
```



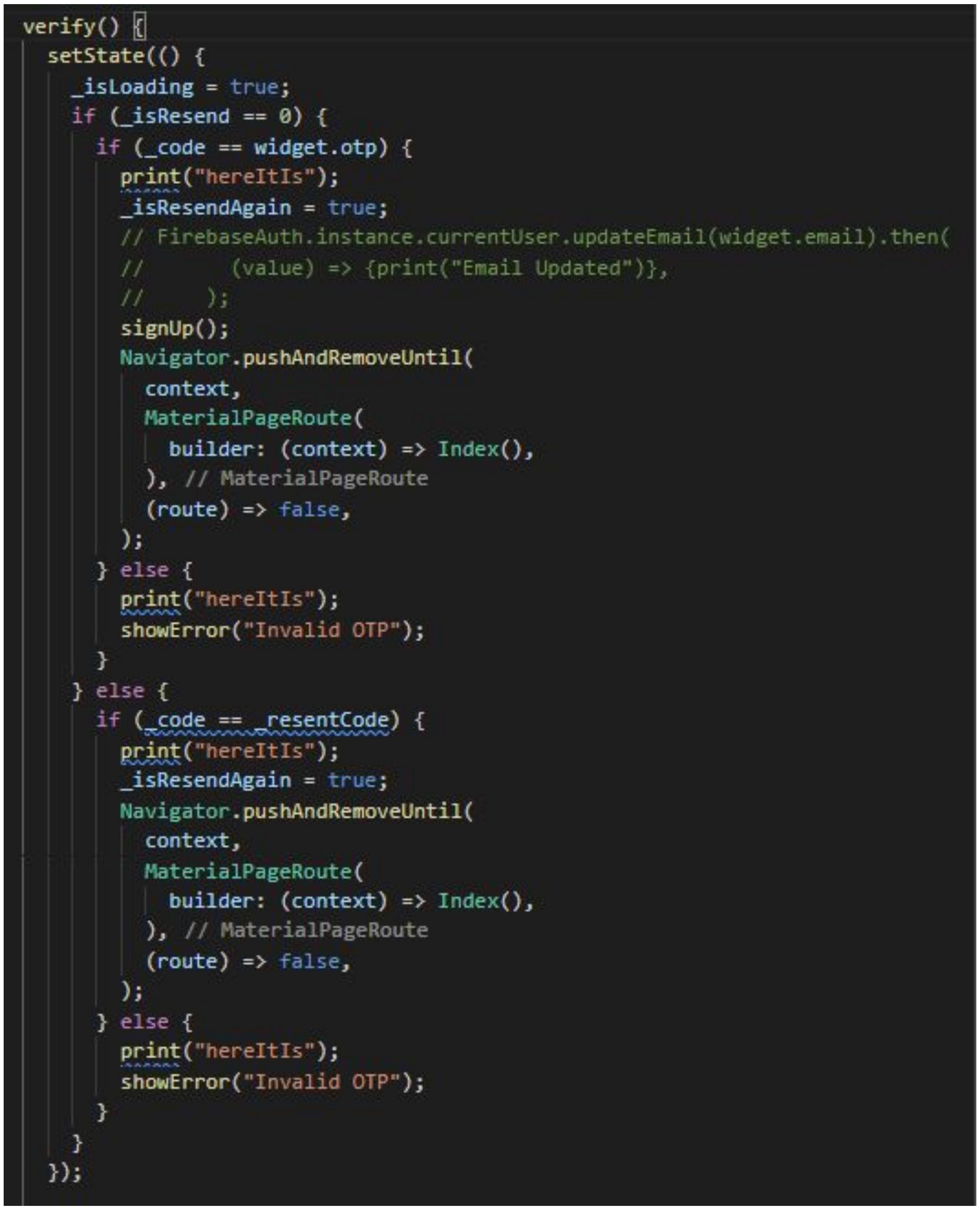
```dart
login() async {
  if (_formKey != null &&
      _formKey.currentState != null &&
      _formKey.currentState!.validate()) {
    _formKey.currentState!.save();
    try {
      await _auth.signInWithEmailAndPassword(
          email: _email, password: _password);
      Navigator.push(
          context, MaterialPageRoute(builder: (context) => Index()));
    } catch (e) {
      showError("Email and Password donot match!!");
    }
  }
}
```

Login Page Checks!!



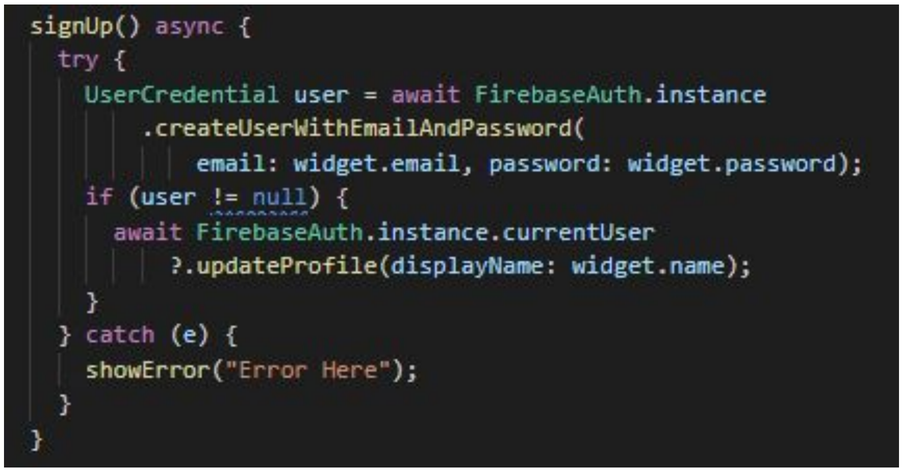
```dart
sendOTP() async {
  // _formKey.currentState?.save();
  if (_formKey.currentState!.validate()) {
    _formKey.currentState!.save();
  }
  var rng = new Random();
  var code = rng.nextInt(9999);
  if (code < 1000) {
    code = code + 1000;
  }
  sendMail(_email, _name, code.toString());
  Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(
          builder: (context) => VerifyUserLogin(
                email: _email,
                name: _name,
                otp: code.toString(),
                password: _password,
              )), // VerifyUserLogin // MaterialPageRoute
      (route) => false);
}
```

Random OTP generation on signup!



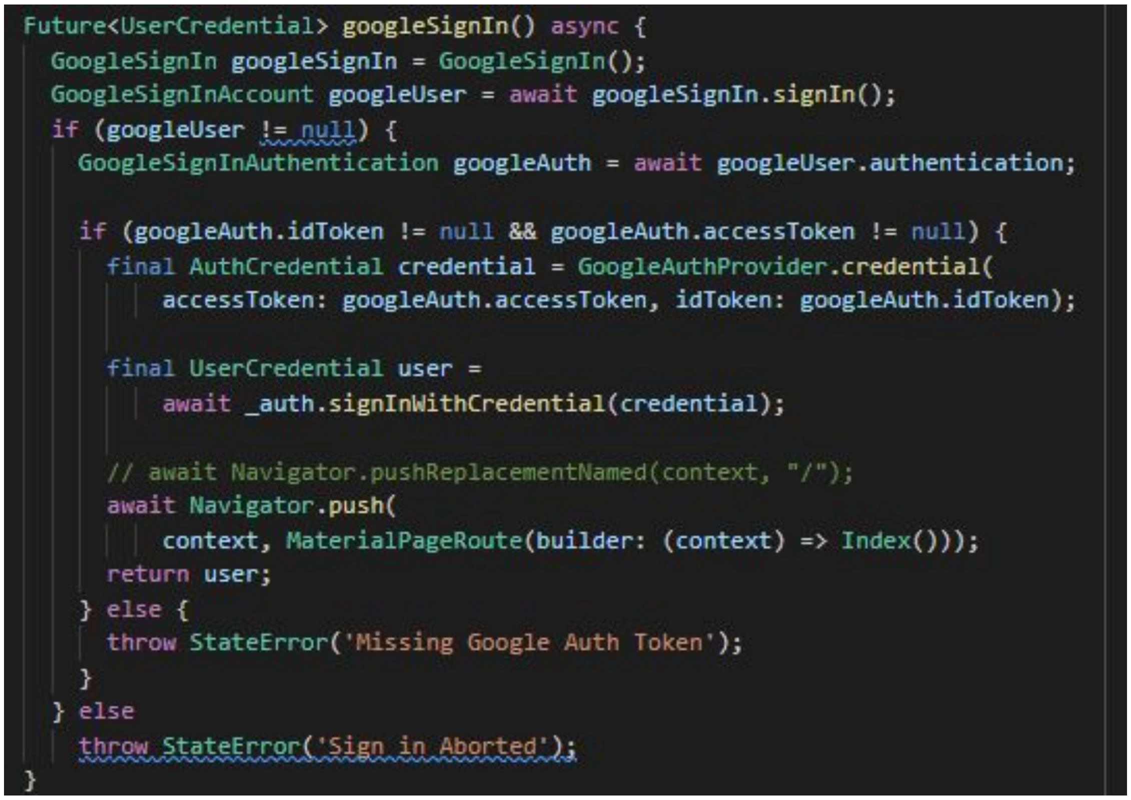
```
verify() {
  setState(() {
    _isLoading = true;
    if (_isResend == 0) {
      if (_code == widget.otp) {
        print("hereItIs");
        _isResendAgain = true;
        // FirebaseAuth.instance.currentUser.updateEmail(widget.email).then(
        //       (value) => {print("Email Updated")},
        //     );
        signUp();
        Navigator.pushAndRemoveUntil(
          context,
          MaterialPageRoute(
            builder: (context) => Index(),
          ), // MaterialPageRoute
          (route) => false,
        );
      } else {
        print("hereItIs");
        showError("Invalid OTP");
      }
    } else {
      if (_code == _resentCode) {
        print("hereItIs");
        _isResendAgain = true;
        Navigator.pushAndRemoveUntil(
          context,
          MaterialPageRoute(
            builder: (context) => Index(),
          ), // MaterialPageRoute
          (route) => false,
        );
      } else {
        print("hereItIs");
        showError("Invalid OTP");
      }
    }
  });
```

OTP Verification



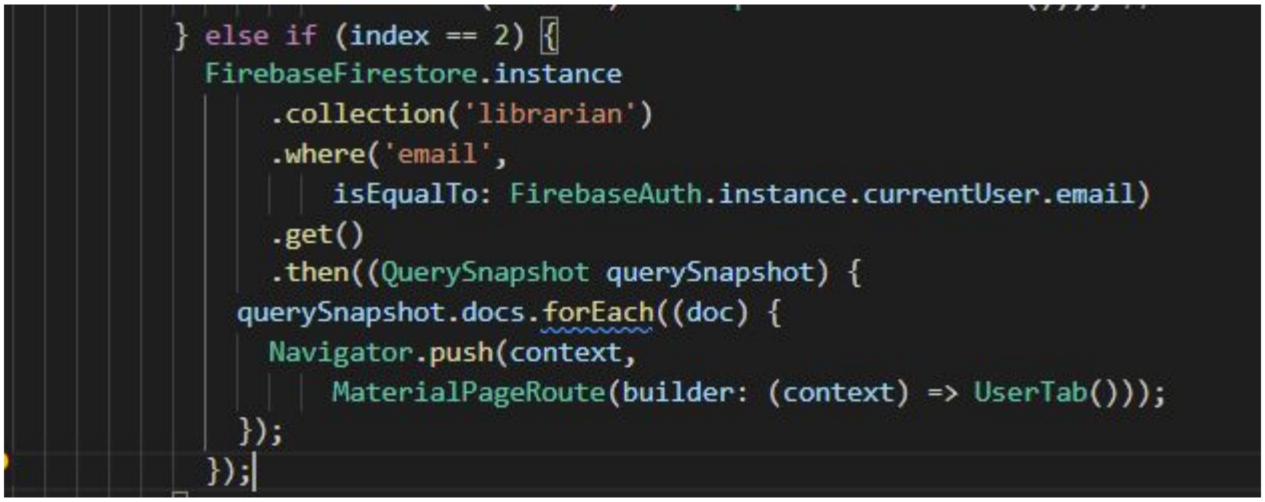
```
signUp() async {
  try {
    UserCredential user = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
            email: widget.email, password: widget.password);
    if (user != null) {
      await FirebaseAuth.instance.currentUser
          ?.updateProfile(displayName: widget.name);
    }
  } catch (e) {
    showError("Error Here");
  }
}
```

SignUp



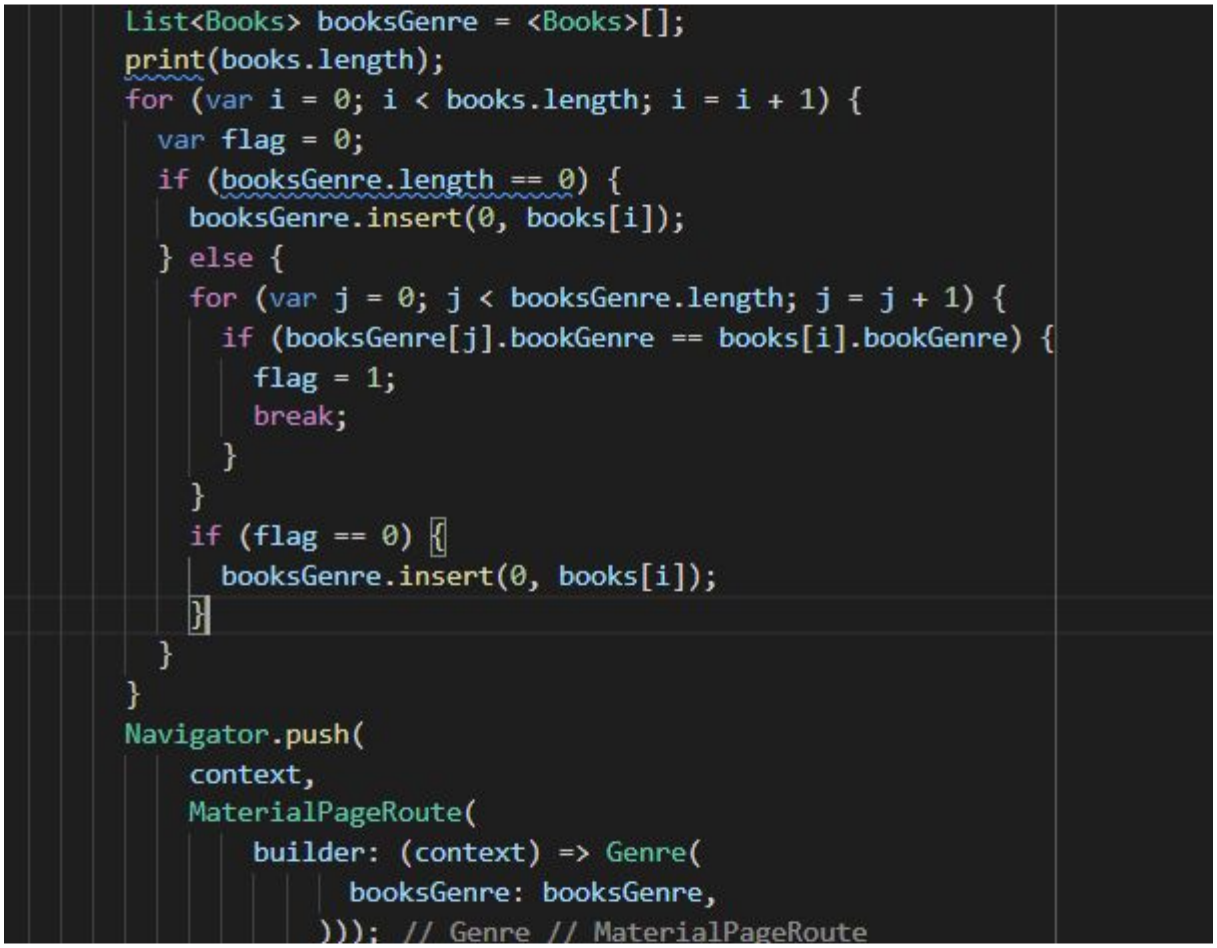
```dart
Future<UserCredential> googleSignIn() async {
  GoogleSignIn googleSignIn = GoogleSignIn();
  GoogleSignInAccount googleUser = await googleSignIn.signIn();
  if (googleUser != null) {
    GoogleSignInAuthentication googleAuth = await googleUser.authentication;

    if (googleAuth.idToken != null && googleAuth.accessToken != null) {
      final AuthCredential credential = GoogleAuthProvider.credential(
          accessToken: googleAuth.accessToken, idToken: googleAuth.idToken);

      final UserCredential user =
          await _auth.signInWithCredential(credential);

      // await Navigator.pushReplacementNamed(context, "/");
      await Navigator.push(
          context, MaterialPageRoute(builder: (context) => Index()));
      return user;
    } else {
      throw StateError('Missing Google Auth Token');
    }
  } else
    throw StateError('Sign in Aborted');
}
```

Google Signup



```dart
} else if (index == 2) {
  FirebaseFirestore.instance
      .collection('librarian')
      .where('email',
          isEqualTo: FirebaseAuth.instance.currentUser.email)
      .get()
      .then((QuerySnapshot querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      Navigator.push(context,
          MaterialPageRoute(builder: (context) => UserTab()));
    });
  });
```

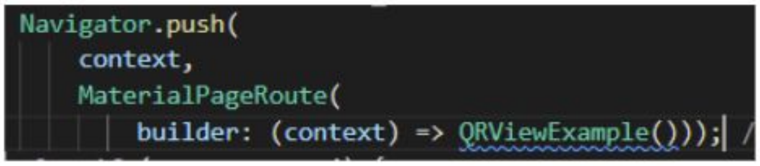
Admin Panel Access Test



```
List<Books> booksGenre = <Books>[];
print(books.length);
for (var i = 0; i < books.length; i = i + 1) {
  var flag = 0;
  if (booksGenre.length == 0) {
    booksGenre.insert(0, books[i]);
  } else {
    for (var j = 0; j < booksGenre.length; j = j + 1) {
      if (booksGenre[j].bookGenre == books[i].bookGenre) {
        flag = 1;
        break;
      }
    }
    if (flag == 0) {
      booksGenre.insert(0, books[i]);
    }
  }
}
Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => Genre(
              booksGenre: booksGenre,
            ))); // Genre // MaterialPageRoute
```

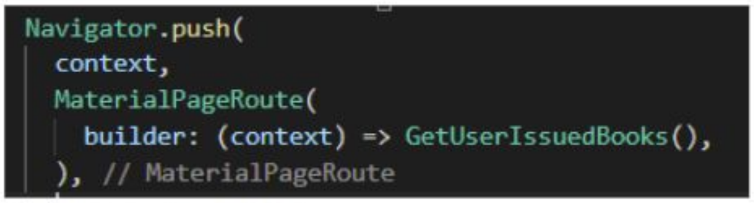
Navigate to Genre List Page

- Navigate to QR Code Scan



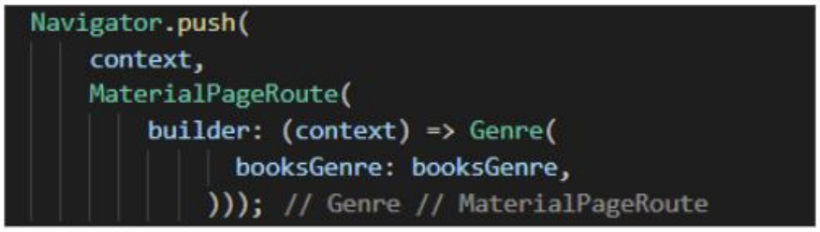
```
Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => QRViewExample())); /
```

- Navigate to My Issued Books



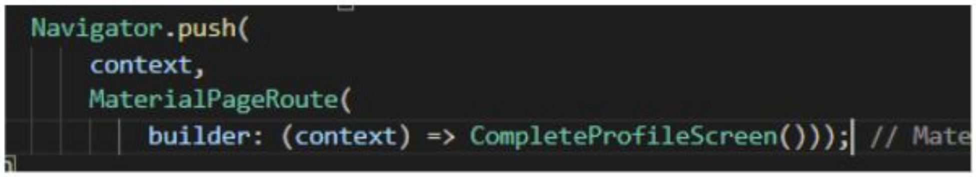
```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => GetUserIssuedBooks(),
  ), // MaterialPageRoute
```

- Navigate to Genre List



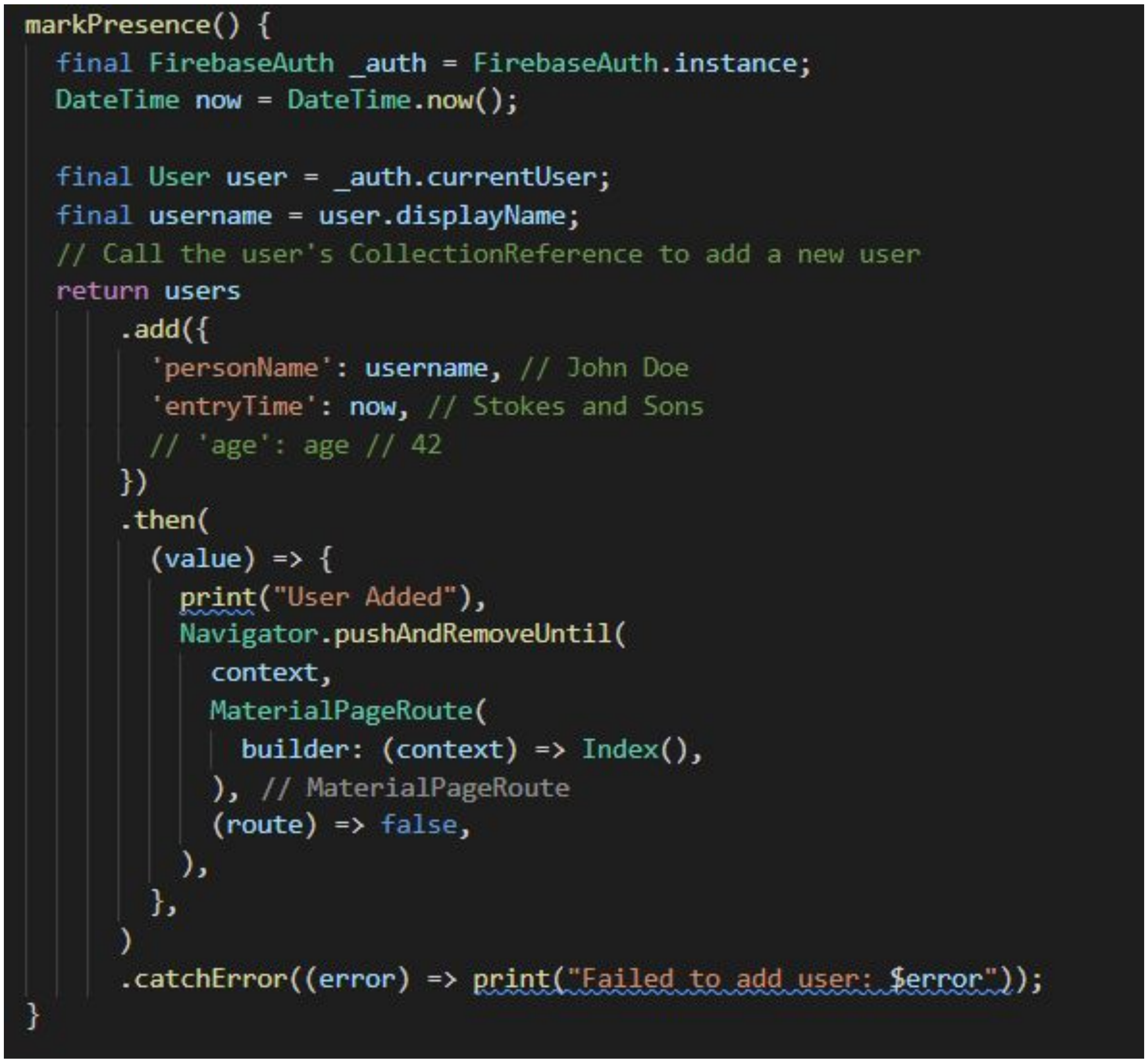
```
Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => Genre(
              booksGenre: booksGenre,
            ))); // Genre // MaterialPageRoute
```

- Navigate to Profile Page



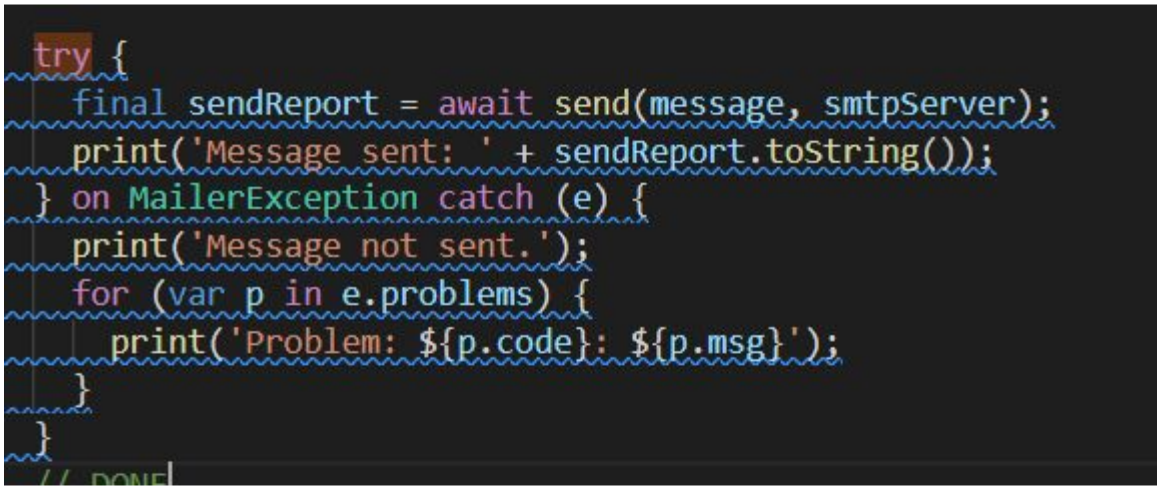
```
Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => CompleteProfileScreen())); // Mate
```

- And so on for navigating to different pages



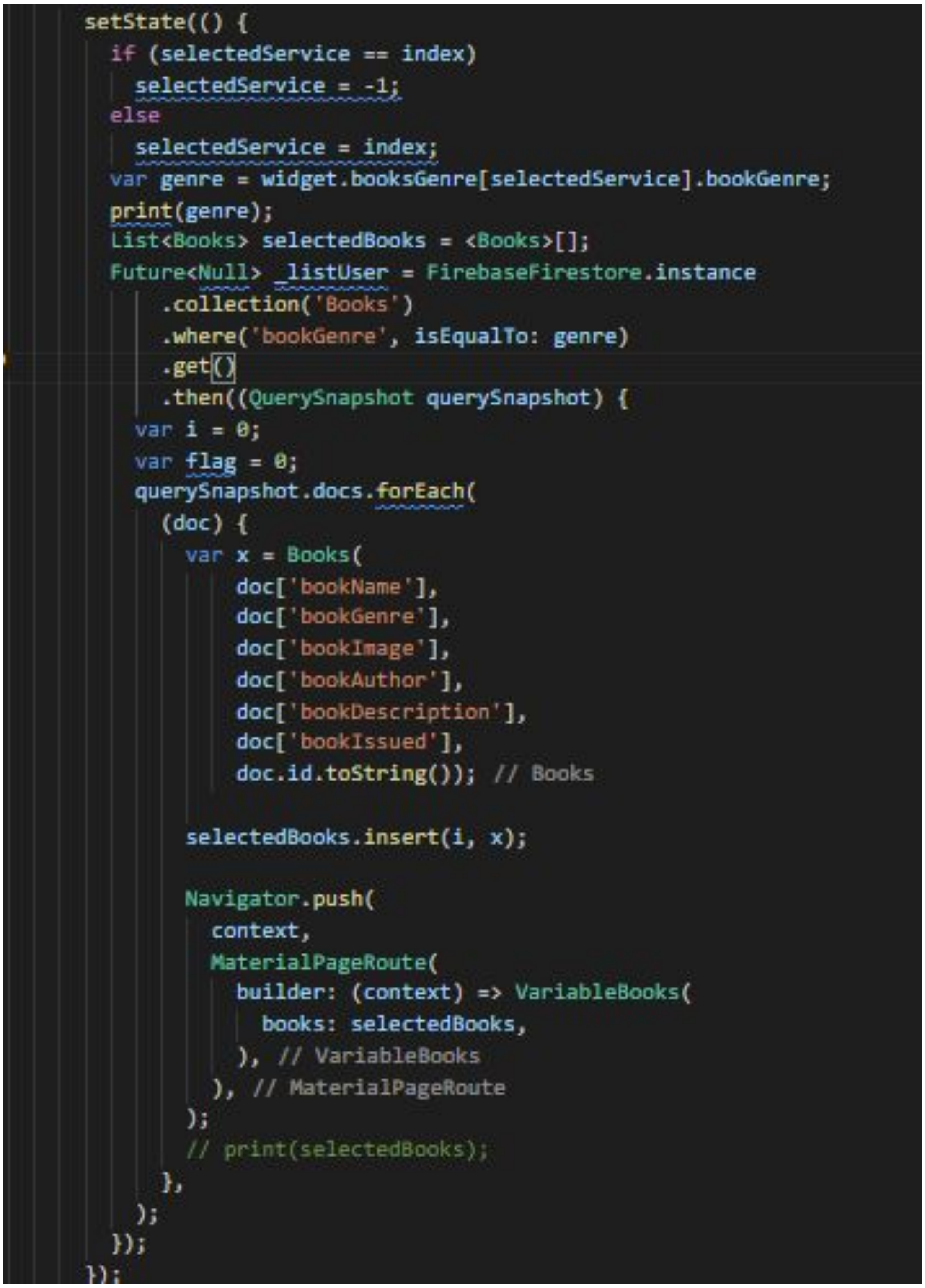
```
markPresence() {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  DateTime now = DateTime.now();

  final User user = _auth.currentUser;
  final username = user.displayName;
  // Call the user's CollectionReference to add a new user
  return users
      .add({
        'personName': username, // John Doe
        'entryTime': now, // Stokes and Sons
        // 'age': age // 42
      })
      .then(
        (value) => {
          print("User Added"),
          Navigator.pushAndRemoveUntil(
            context,
            MaterialPageRoute(
              builder: (context) => Index(),
            ), // MaterialPageRoute
            (route) => false,
          ),
        },
      )
      .catchError((error) => print("Failed to add user: $error"));
}
```

QR Code Scan Page



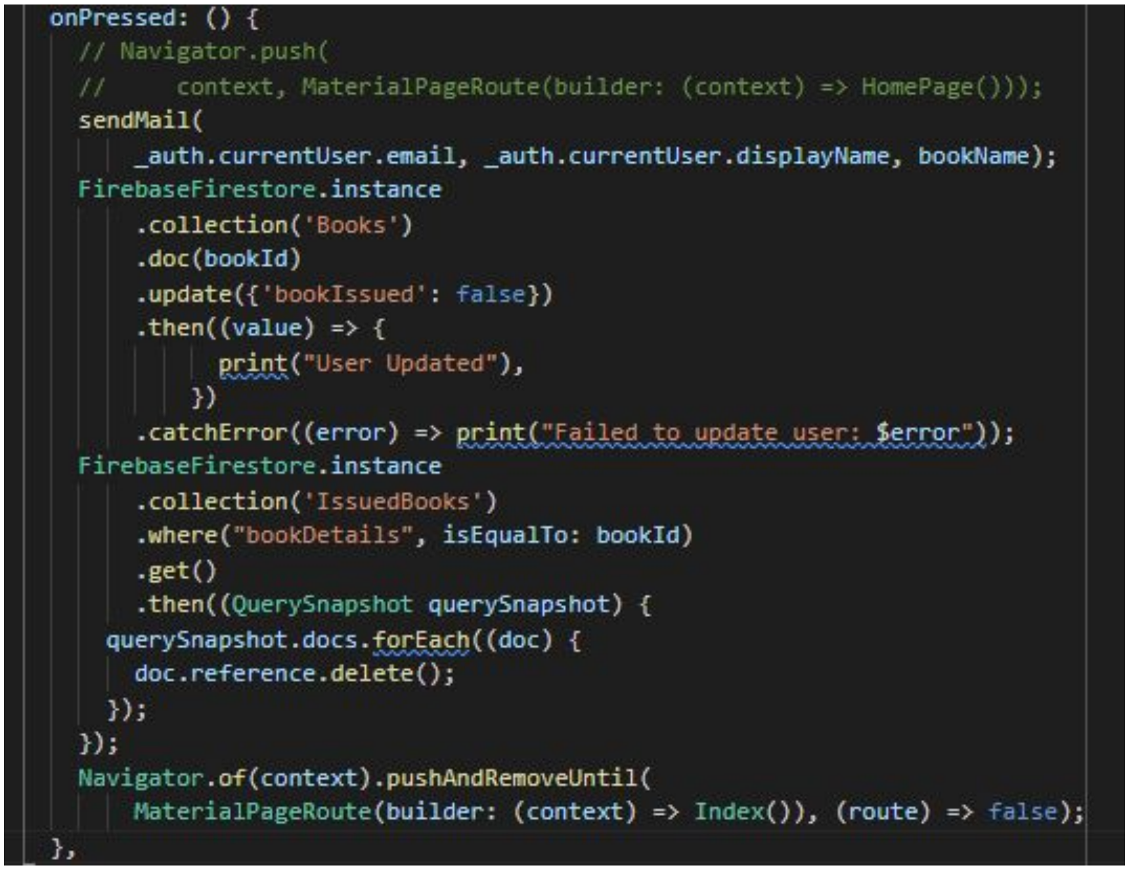
```
try {
  final sendReport = await send(message, smtpServer);
  print('Message sent: ' + sendReport.toString());
} on MailerException catch (e) {
  print('Message not sent.');
  for (var p in e.problems) {
    print('Problem: ${p.code}: ${p.msg}');
  }
}
// DONE
```

Issued Books Page:



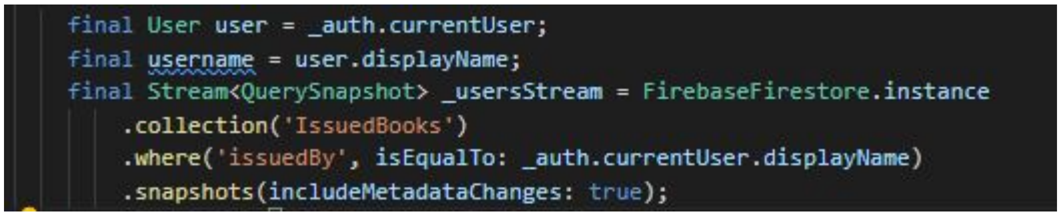
```
setState(() {
  if (selectedService == index)
    selectedService = -1;
  else
    selectedService = index;
  var genre = widget.booksGenre[selectedService].bookGenre;
  print(genre);
  List<Books> selectedBooks = <Books>[];
  Future<Null> _listUser = FirebaseFirestore.instance
      .collection('Books')
      .where('bookGenre', isEqualTo: genre)
      .get()
      .then((QuerySnapshot querySnapshot) {
    var i = 0;
    var flag = 0;
    querySnapshot.docs.forEach(
      (doc) {
        var x = Books(
            doc['bookName'],
            doc['bookGenre'],
            doc['bookImage'],
            doc['bookAuthor'],
            doc['bookDescription'],
            doc['bookIssued'],
            doc.id.toString()); // Books

        selectedBooks.insert(i, x);

        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => VariableBooks(
              books: selectedBooks,
            ), // VariableBooks
          ), // MaterialPageRoute
        );
        // print(selectedBooks);
      },
    );
  });
});
```

Genre List Page



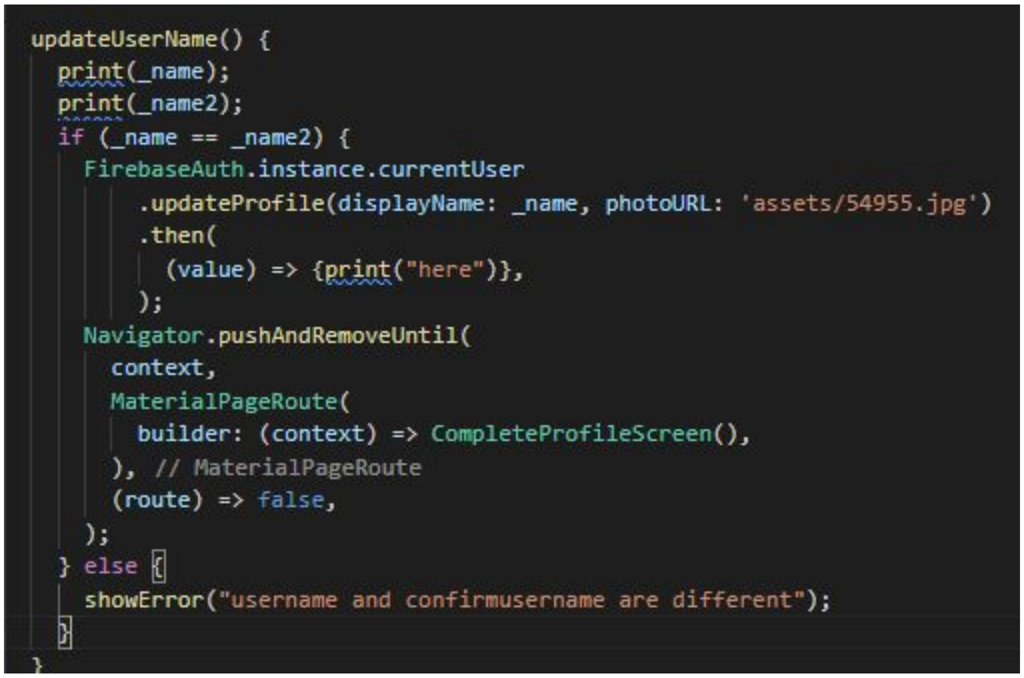
```
onPressed: () {
  // Navigator.push(
  //     context, MaterialPageRoute(builder: (context) => HomePage()));
  sendMail(
      _auth.currentUser.email, _auth.currentUser.displayName, bookName);
  FirebaseFirestore.instance
      .collection('Books')
      .doc(bookId)
      .update({'bookIssued': false})
      .then((value) => {
            print("User Updated"),
          })
      .catchError((error) => print("Failed to update user: $error"));
  FirebaseFirestore.instance
      .collection('IssuedBooks')
      .where("bookDetails", isEqualTo: bookId)
      .get()
      .then((QuerySnapshot querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      doc.reference.delete();
    });
  });
  Navigator.of(context).pushAndRemoveUntil(
      MaterialPageRoute(builder: (context) => Index()), (route) => false);
},
```

Return Book Button

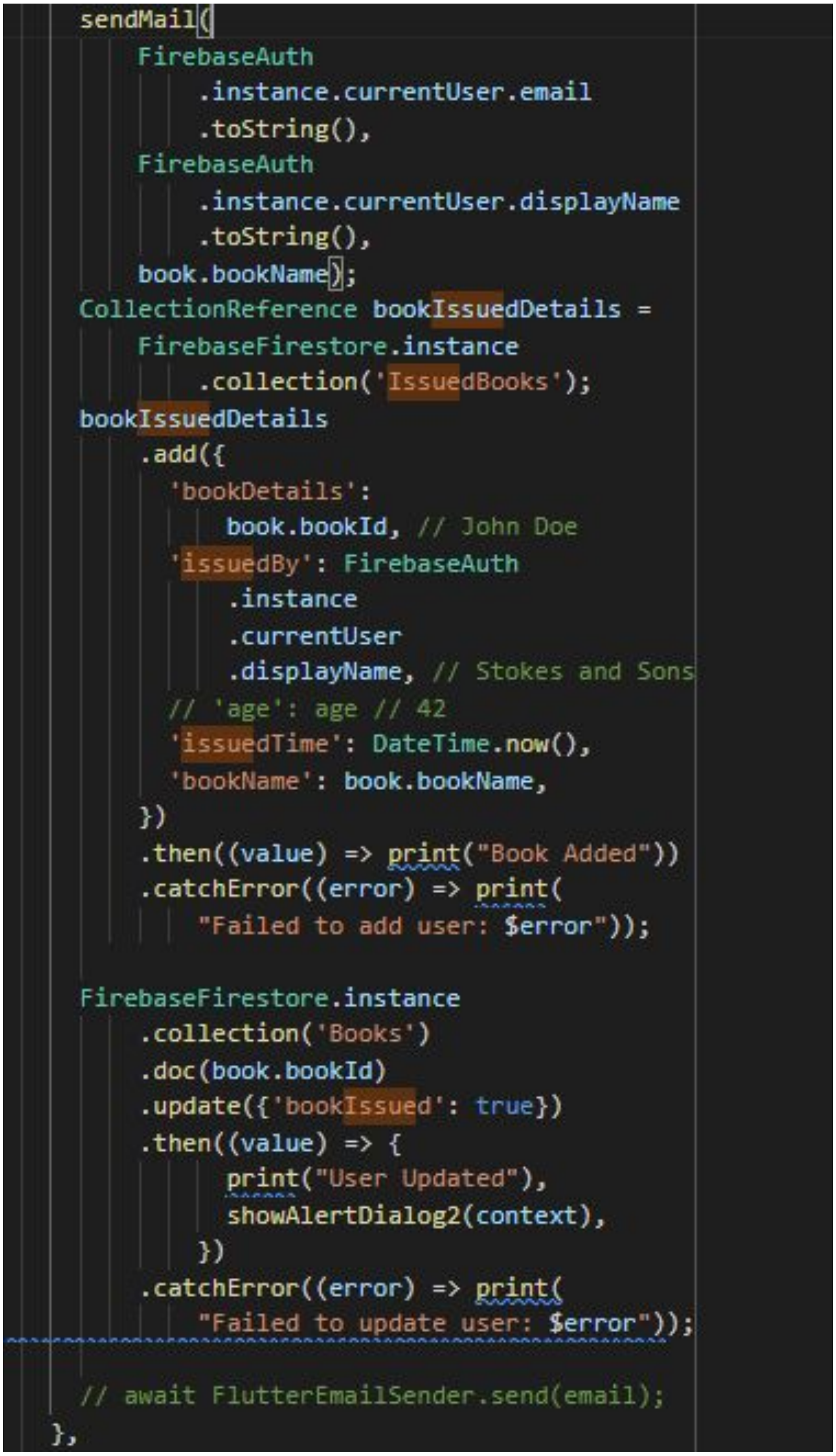


```
final User user = _auth.currentUser;
final username = user.displayName;
final Stream<QuerySnapshot> _usersStream = FirebaseFirestore.instance
    .collection('IssuedBooks')
    .where('issuedBy', isEqualTo: _auth.currentUser.displayName)
    .snapshots(includeMetadataChanges: true);
```

Fetch user issued books

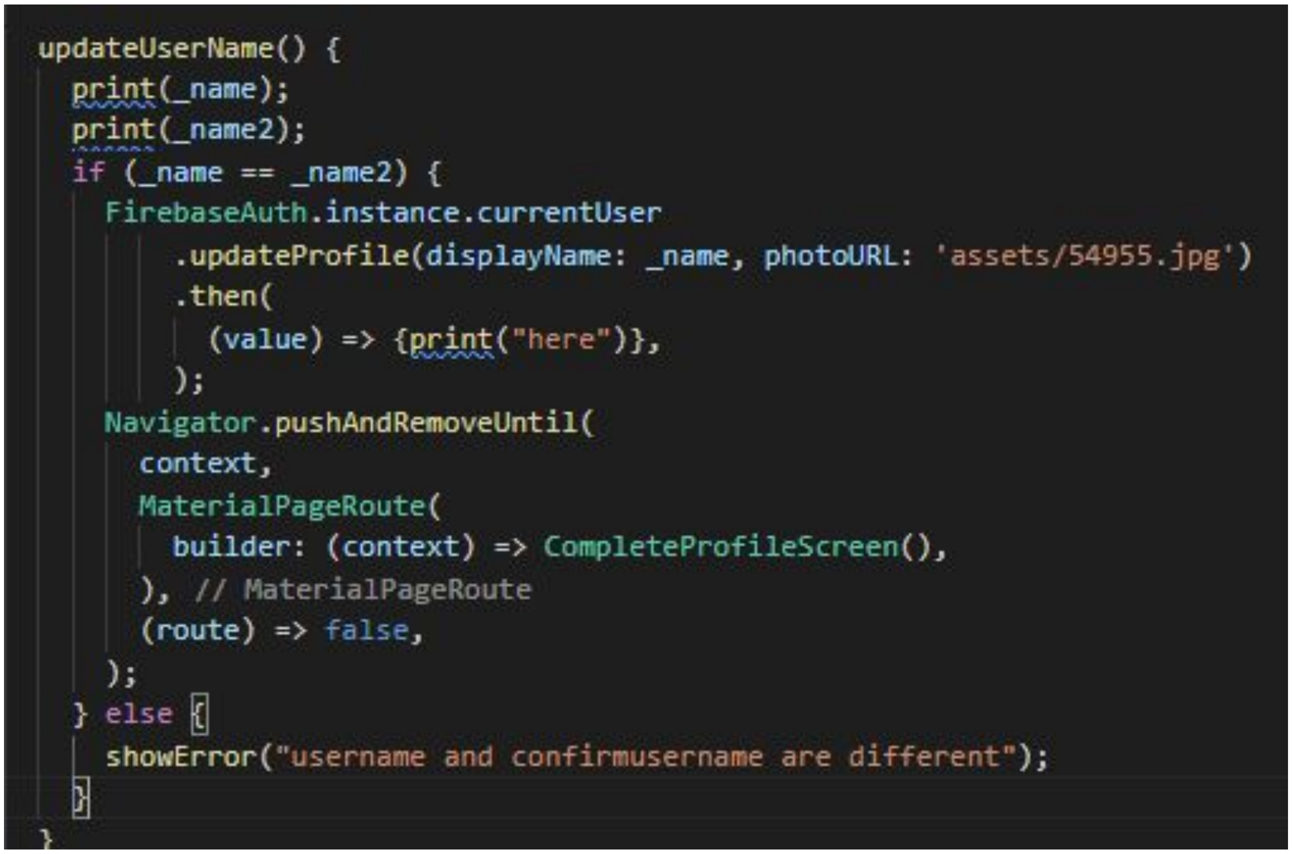


```
updateUserName() {
  print(_name);
  print(_name2);
  if (_name == _name2) {
    FirebaseAuth.instance.currentUser
        .updateProfile(displayName: _name, photoURL: 'assets/54955.jpg')
        .then(
          (value) => {print("here")},
        );
    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(
        builder: (context) => CompleteProfileScreen(),
      ), // MaterialPageRoute
      (route) => false,
    );
  } else {
    showError("username and confirmusername are different");
  }
}
```



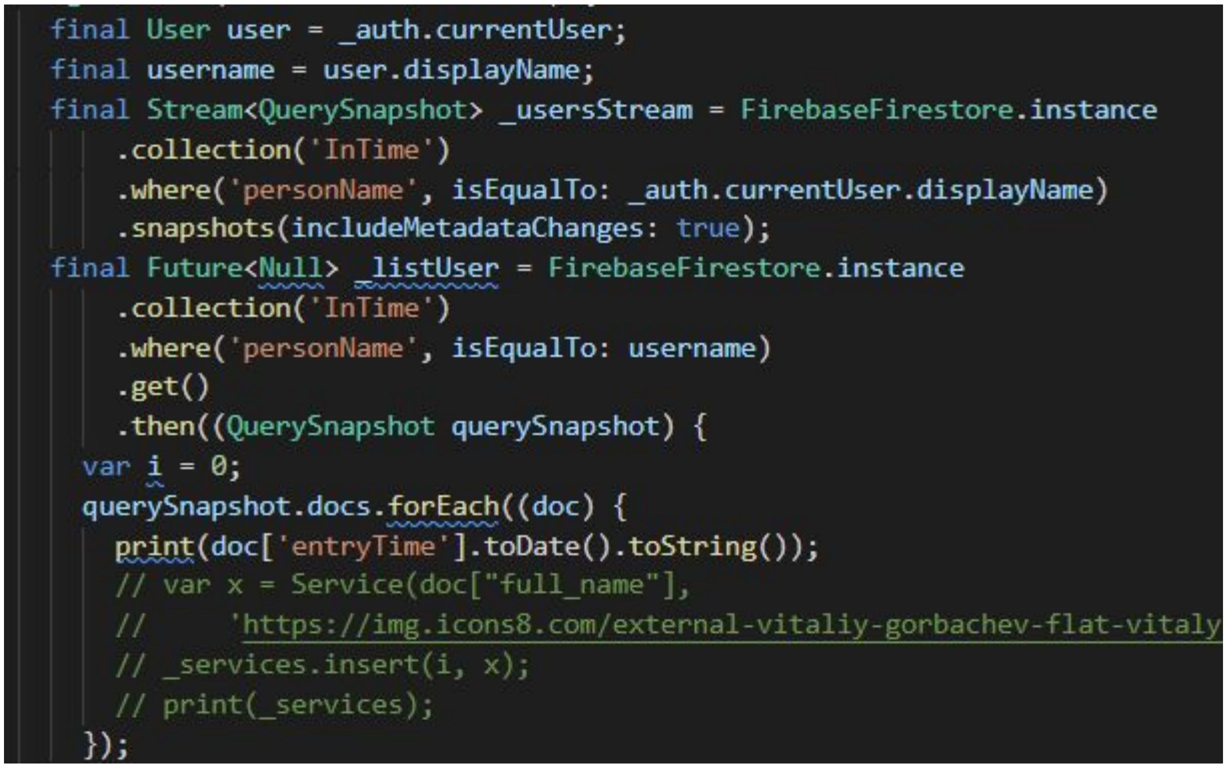
```
    sendMail(
        FirebaseAuth
            .instance.currentUser.email
            .toString(),
        FirebaseAuth
            .instance.currentUser.displayName
            .toString(),
        book.bookName);
    CollectionReference bookIssuedDetails =
        FirebaseFirestore.instance
            .collection('IssuedBooks');
    bookIssuedDetails
        .add({
          'bookDetails':
              book.bookId, // John Doe
          'issuedBy': FirebaseAuth
              .instance
              .currentUser
              .displayName, // Stokes and Sons
          // 'age': age // 42
          'issuedTime': DateTime.now(),
          'bookName': book.bookName,
        })
        .then((value) => print("Book Added"))
        .catchError((error) => print(
            "Failed to add user: $error"));

    FirebaseFirestore.instance
        .collection('Books')
        .doc(book.bookId)
        .update({'bookIssued': true})
        .then((value) => {
              print("User Updated"),
              showAlertDialog2(context),
            })
        .catchError((error) => print(
            "Failed to update user: $error"));

    // await FlutterEmailSender.send(email);
  },
```

Issue Book page



```
updateUserName() {
  print(_name);
  print(_name2);
  if (_name == _name2) {
    FirebaseAuth.instance.currentUser
        .updateProfile(displayName: _name, photoURL: 'assets/54955.jpg')
        .then(
          (value) => {print("here")},
        );
    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(
        builder: (context) => CompleteProfileScreen(),
      ), // MaterialPageRoute
      (route) => false,
    );
  } else {
    showError("username and confirmusername are different");
  }
}
```

Update Username Page



```
  final User user = _auth.currentUser;
  final username = user.displayName;
  final Stream<QuerySnapshot> _usersStream = FirebaseFirestore.instance
      .collection('InTime')
      .where('personName', isEqualTo: _auth.currentUser.displayName)
      .snapshots(includeMetadataChanges: true);
  final Future<Null> _listUser = FirebaseFirestore.instance
      .collection('InTime')
      .where('personName', isEqualTo: username)
      .get()
      .then((QuerySnapshot querySnapshot) {
    var i = 0;
    querySnapshot.docs.forEach((doc) {
      print(doc['entryTime'].toDate().toString());
      // var x = Service(doc["full_name"],
      //     'https://img.icons8.com/external-vitaliy-gorbachev-flat-vitaly
      // _services.insert(i, x);
      // print(_services);
    });
```

User activity page

<u>And many other pages include testing scripts like this, which can be found in the code in github repository.</u>