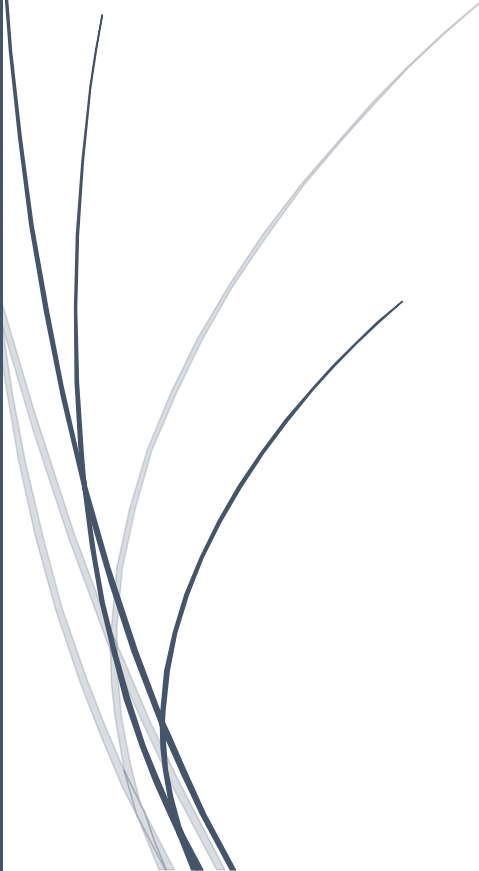
A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from the bar, containing the date.

25/2/2021

# SOFTWARE REQUIREMENTS SPECIFICATION

QR Code Scanner.

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Version 1.0 approved

---

**Prepared by:**

Ahmed Mohamed Hussein Hashem  
Mohamed Hussein Mohamed ABD-ELHAFEZ  
Amr Salamah Saree Youssef  
Mohamed Mohamed Syed Ali  
Mostafa Abdel Nasser Yahya Khalifa  
Mahmoud Mostafa Shafiq Ramadan

**Supervisor:**

Prof/Taysir Hassan

# Table of Contents

<b>Introduction.....</b>	<b>1</b>
Purpose.....	1
Product Scope.....	1
Actors.....	1
References.....	1
<b>Overall Description.....</b>	<b>1</b>
Product Perspective.....	1
Product Functions.....	1
User Classes and Characteristics.....	2
Operating Environment.....	2
Design and Implementation Constraints.....	2
User Documentation.....	2
Assumptions and Dependencies.....	2
<b>System Requirements specifications.....</b>	<b>2</b>
Functional Requirements.....	2
System Students Affairs.....	2
Security.....	2
Student.....	3
Non-Functional Requirements.....	3
Performance Requirements.....	3
Safety and Security Requirements.....	3
Reliability.....	3
Other Software Quality Attributes.....	3
<b>System Design.....</b>	<b>4</b>
UML Use Case Diagram.....	4
<b>Implementation.....</b>	<b>5</b>

Graphical user interface.....	5
database.....	13
Implementation.....	16
Code.....	18

## Introduction

The following section provides an overview of the software requirements specifications (SRS) for the QR Code Scanner.

## Purpose

*The purpose of SRS is to determine both functional and non-functional requirements of QR Code Scanner.*

*Also, the document provides an overall description about the QR Code Scanner with UML analysis models.*

## Product Scope

*Facilitating the university and the students in making the carnation and monitoring student activity in the university.*

*Converting paper cards to electronic cards.*

## Actors

<b>Software admin</b>	Someone who is responsible for maintaining, updating and checking errors of the software.
<b>System user</b>	Someone who use the application.

## References

1- <https://dart.dev/tutorials>

2- [https://www.tutorialspoint.com/dart\\_programming/index.htm](https://www.tutorialspoint.com/dart_programming/index.htm)

3- <https://riptutorial.com/dart>

## Overall Description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

## Product Perspective

*This system merges various hardware and software elements.*

## Product Functions

*System student will be able create QR code.*

*System Students Affairs will be able to add student to database .*

*System Security will be able to scanning qr code.*

## User Classes and Characteristics

*There are three types of users that interacts with the system*

*System Students Affairs: responsible for add student or edit him.*

*System Security: scanning qr code and store inter student.*

*System Students: generate qr code.*

## Operating Environment

- o *any mobile android.*
- o *Computer.*

## Design and Implementation Constraints

*Connect to internet.*

*Good source of light to scan qr code.*

## User Documentation

*-The Student simply needs to open the program to show the qr qr code.*

*-The security need to connect to internet and log in to account and scan the student code.*

## Assumptions and Dependencies

*One assumption about the software is that it will be always used on desktop computers or mobile so all the actors included in the software should be aware of the computers .*

*There must be a source of light for scan code.*

# System Requirements specifications.

## Functional Requirements

### System Students Affairs:

- 1- System must support add student.
- 2- System must support edit student.
- 3- System must support delete student.
- 4- System must support search for a student.

### Security:

- 1- System must support scan qr code.
- 2- System must support connect to database.

3- System must show student data after scan.

### Student:

1- System must create qr code by enter National identification number of student.

## Non-Functional Requirements

**Non-Functional requirements** are requirements that specify criteria for the developed system.

ensure that no one of app users loses any data while using app the developer team updates app regularly.app provides the users with both simple and advanced features. Due to its well designed and easy to use interface it can be used by both experts and typical users

## Performance Requirements

The system should be compatible with Android ,IOS and Web Browser .

The database must response to data entry fast.

The system must response to the user operations in less than 1 millisecond.

The system should be reliable.

There must be a light source for scan qr code.

## Safety and Security Requirements

System cannot affect, harm, damage to user.

It also cannot damage user's computer while accessing the system over a network.

## Reliability

System should be designed in modular manner to ease in software maintenance. By designing modularly, we can reduce coupling allowing each module to perform a specific function.

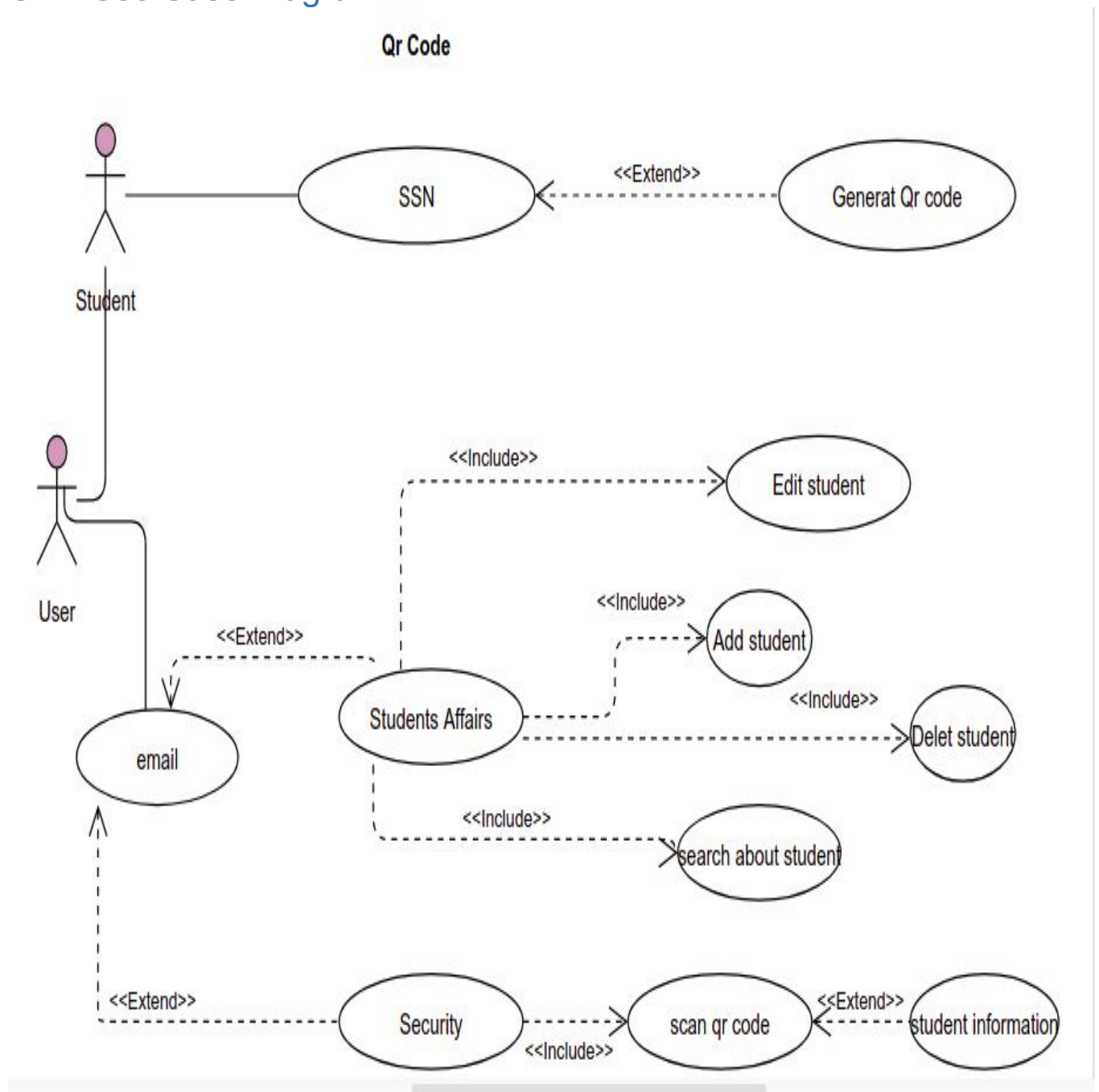
The program should be reliable and provide catching of exceptions, so the unintended results don't occur such as system crashes, or data validation failures

## Other Software Quality Attributes

The system should have user-friendly interface.

## System Design

### UML Use Case Diagram





## Implementation:

### 1- Graphical user interface:

1- When open the app this is first page it's content to log in email and password to the employ and log in student this don't need email and password



Login

Login as Student ?

2- When we click to icon Login as student ? the program display student form it's we write the National ID + student code to display qr code to student should the student show this code to security to scan this code.


Orange



qr data  
29704012613519162016029

Generate

3- When the employ need to log in enter the email and password if this account of security app open the security page when the log in is Students Affairs the app open the Students Affairs page.

email

ahmed@fci.com

password

.....

Login

Login as Student ?

4- When the log in the Students Affairs the app open this there no data yet this page enable to the Students Affairs to search about student by code or add student or delete or edit and log out.

⇨

student code

🔍

there is no students in database yet



5- When click to plus icon the app open this page this page content the some table name and phone and faculty and grade and National ID and student code and app offer to option update or delete if we click on delete this student delete from database if click for add icon this student add to database.



name

احمد محمد حسين هاشم

phone

01093506318

faculty

حاسبات ومعلومات

grade

4

ssn

29704012613519

student code

162016029



6- When click to add icon student add to database and display in this page in this page we can delete them or edit





احمد محمد حسين هاشم

code: 162016029

ssn: 29704012613519

faculty: حاسبات ومعلومات

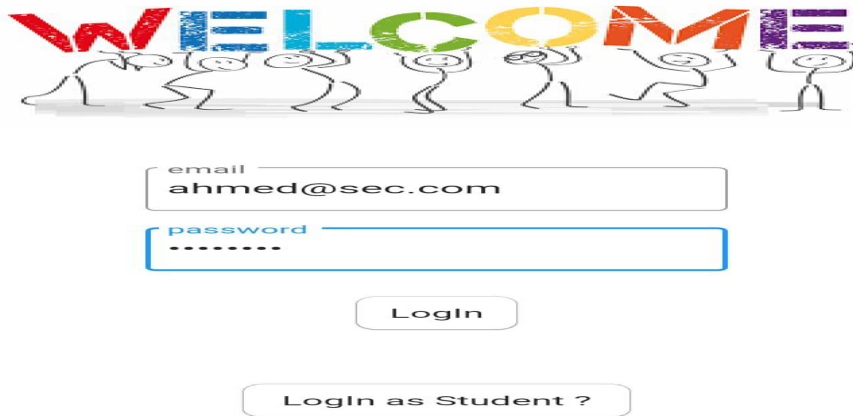
grade: 4

phone number: 01093506318





7- In this we log in with the security email .



A welcome screen featuring a colorful 'WELCOME' text at the top, where each letter is held up by a stick figure. Below this is a login form with two input fields: 'email' containing 'ahmed@sec.com' and 'password' containing seven dots. A 'Login' button is positioned below the password field, and a 'Login as Student ?' button is at the bottom.

WELCOME

email  
ahmed@sec.com

password  
.....

Login

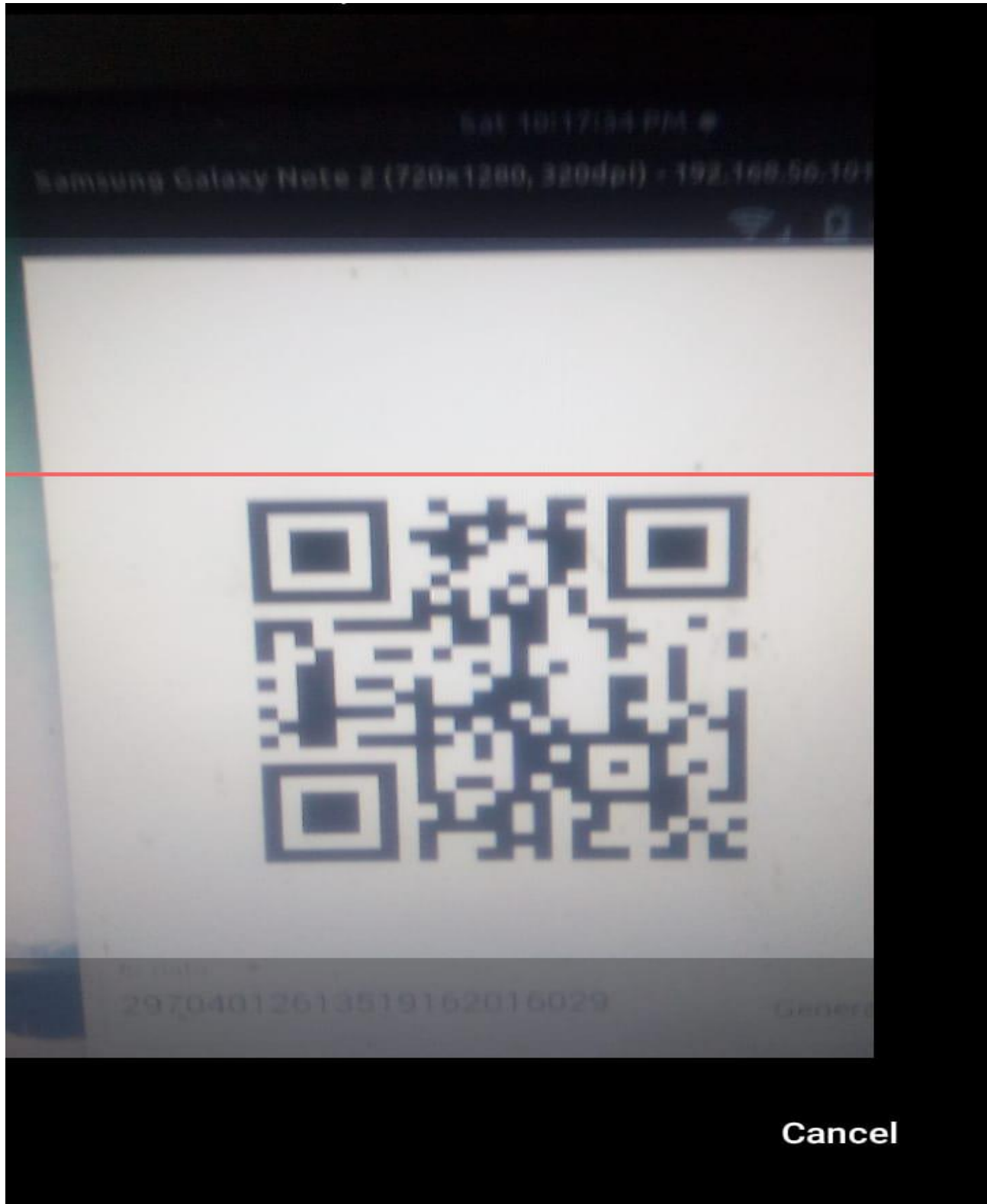
Login as Student ?

8- When log in with the security email the app display this page enable to the security to scan qr code from the student

sign out

start scanning

9- When click to start scanning icon the app open the camera of mobile to scan qr code this page is appear.



10- When scan code if the student is find in database the app display the all information about this student.



احمد محمد حسين هاشم

code: 162016029

ssn: 29704012613519

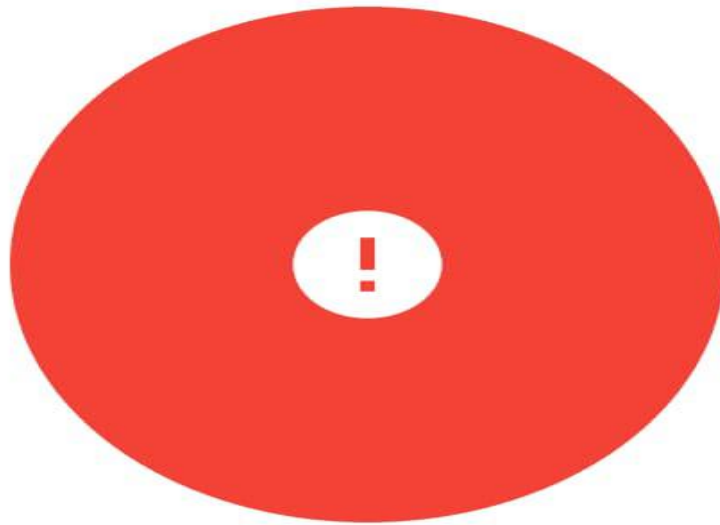
faculty: حاسبات ومعلومات

grade: 4

phone number: 01093506318



11- If student not find in database the app display this picture.





## 2- Database:

In this application we use Firebase Realtime Database: it's Affiliated to Google, is a cloud-hosted database, Data is stored as JSON and synchronized in realtime to every connected client. all of clients share one Realtime Database instance and automatically receive updates with the newest data.

Can use Firebase with build cross-platform apps with our iOS, Android, and JavaScript SDKs.

In addition to the development software packages, there is a library called FirebaseUI that provides a set of useful utilities to make developing with Firebase easier.

There are also projects like AngularFire that put web development software packages for use with the Angular website interface design framework. It is open source, by the way.

Firebase sends new data as soon as it is updated. When your user(Security, Students Affairs) saves a data change, all connected customers receive the updated data in real time.

### - advantages :

1- Realtime Database: It is useful for storing data on the server and the most characteristic of it is that it is Realtime, meaning that any change that occurs to the databases will change immediately in the application

2- Storage: store files and photos

3- Hosting: to host your site on Firebase

4- Notifications: Send notifications

In this application we use in Firebase :

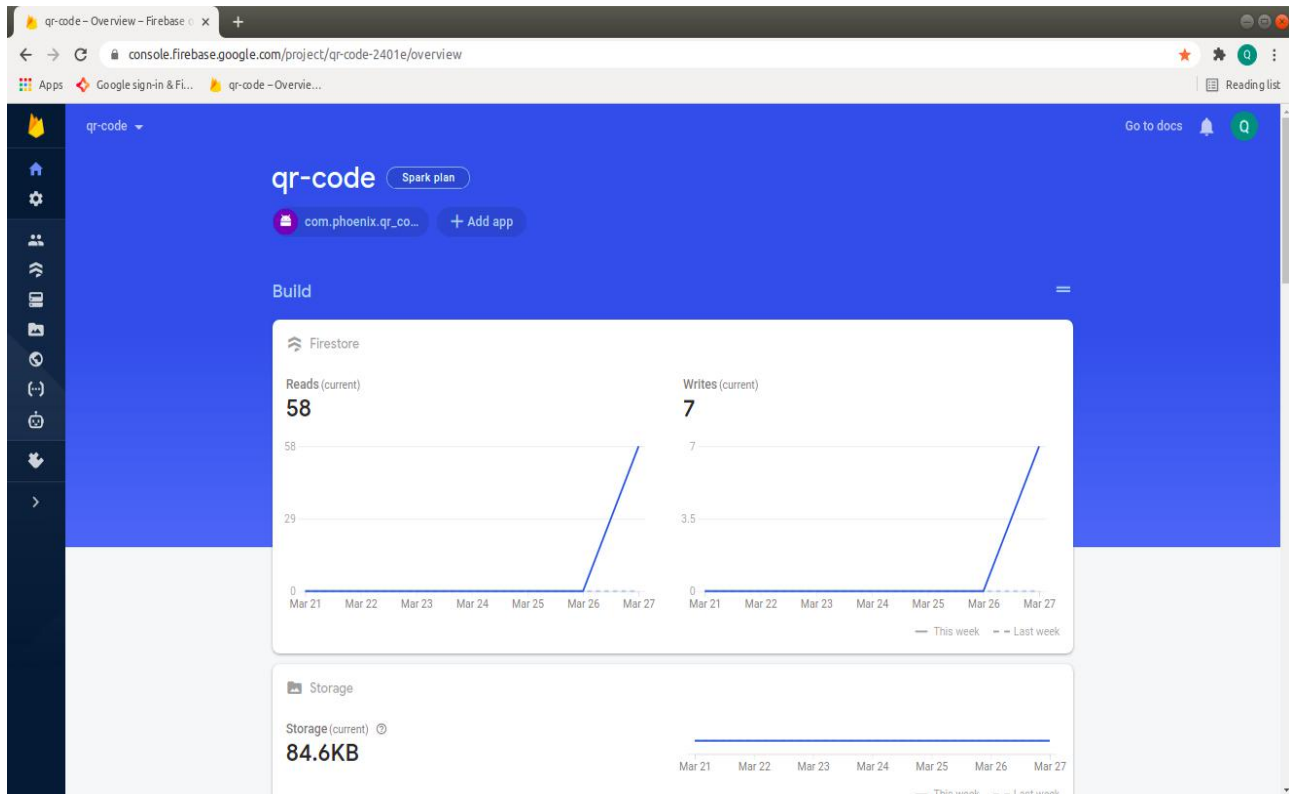
- **Firebase Authentication:** to provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

- **Cloud Firestore:** is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.

- **Cloud Storage for Firebase:** is a powerful, simple, and cost-effective object storage service built for Google scale. We use it our this app to store images of student.

We tack screen shot about Firebase of this project.

When we open Firebase we show first page it overview about this data it's appear number of time app read data from Firebase it's 58 and number of time app write data from Firebase it's 7, and storage usage.



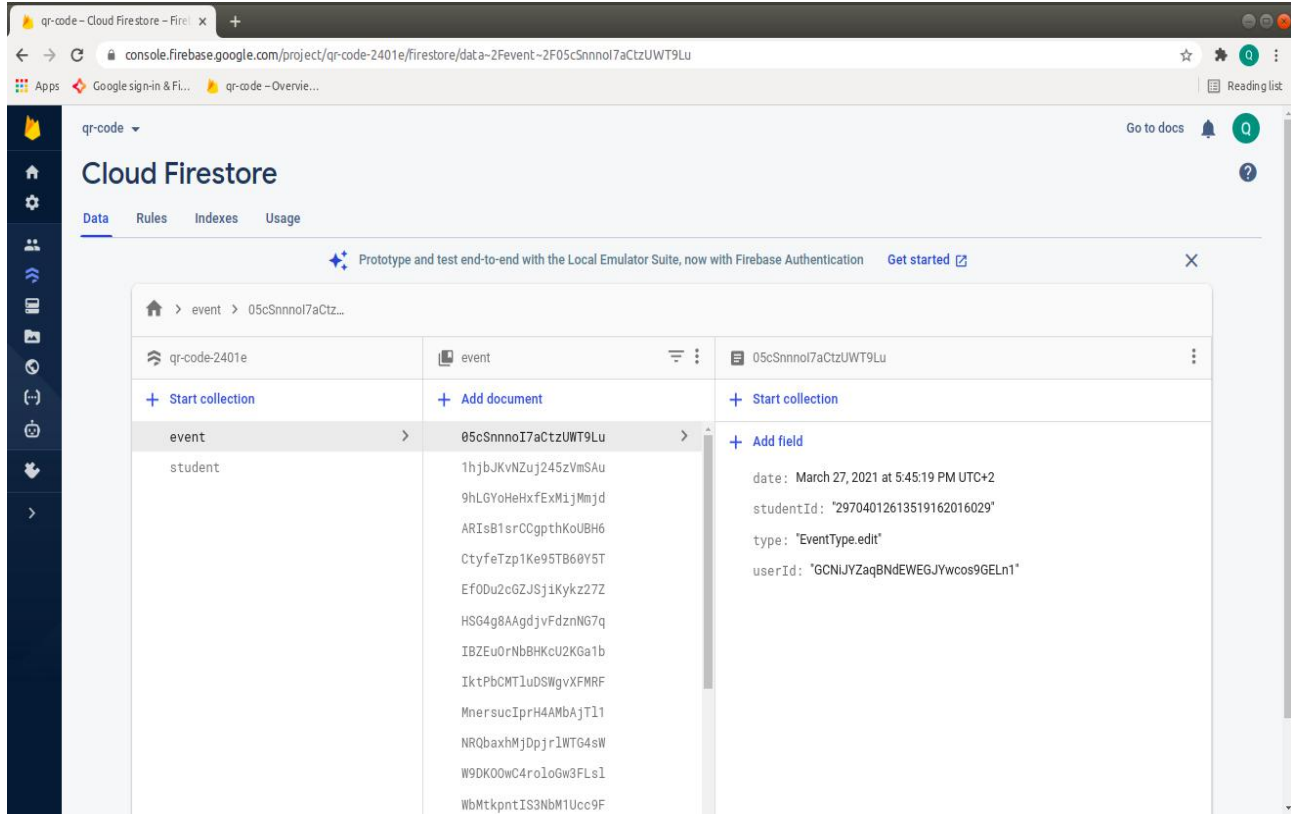
Next we go to authentication page we show this page

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. In this app we use authentication using email and password. This appears in the next picture where we add email one to Security and one to Students Affairs.

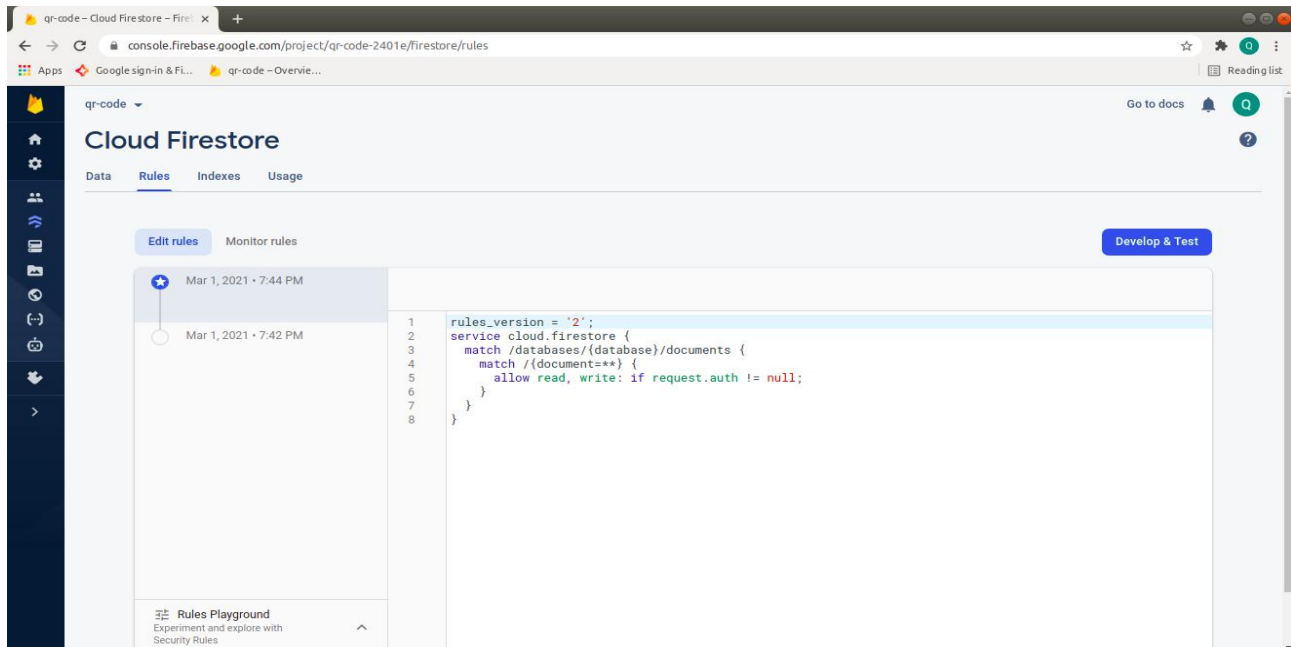
The screenshot shows the 'Users' page in the Firebase Authentication console. It features a search bar at the top and a table of users. The table has columns for Identifier, Providers, Created, Signed in, and User UID. Two users are listed: ahmed@fci.com and ahmed@sec.com. The page also includes a 'Get started' button and a 'Rows per page' dropdown.

Identifier	Providers	Created	Signed in	User UID
ahmed@fci.com	📧	Mar 1, 2021	Mar 27, 2021	GCNIJYZaqBNdEWEGJYwcos9GE...
ahmed@sec.com	📧	Mar 1, 2021	Mar 27, 2021	VvbMZ3SQNJRcNcl7i0JXGt1rYM...

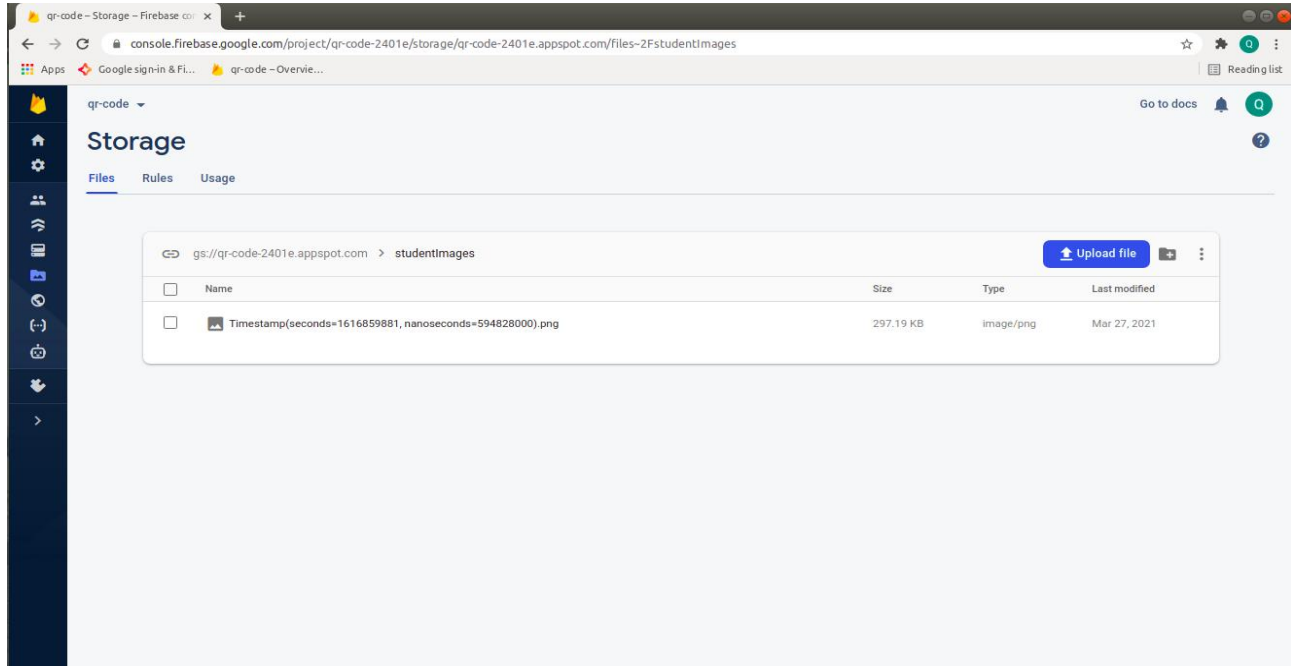
Cloud Firestore is a flexible, scalable database for mobile, web it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. In this app we store the student information and event (any employ add any thing this app save him ).



We add in this page the rules of data and access we here add read and Write access to any user have email.



Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. we use it to store images of students.



### 3- implementation :

In this app we use Flutter SDK Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. **Why we use flutter to this reason.**

**1- Flutter provides higher quality for multi-platform development :** Cross-platform development means that you can write your code once and run it across multiple platforms. is ideal for teams looking to reduce costs and build their products faster This adaptation can save the time of users. Unlike other cross-platform frameworks that look and feel the same on every platform, Flutter apps look and feel native both on iOS and Android. By supporting web and desktop as well, Flutter is uniquely positioned as a strong multi-platform UI toolkit. This means that you don't need to learn several languages and frameworks to bring your product to multiple platforms.

**2- Flutter apps use a single codebase, allowing faster time-to-market:** Compared to native platform development, writing multi-platform apps with a single codebase has many advantages:

- All your business logic and UI code live in one place.
- You can deliver apps with feature parity across all platforms.

With Flutter, you can get these benefits and deliver high-quality apps in a fraction of the time that it would take when using native platform development.

3- **The Dart programming language is very productive:** Flutter apps are built using the Dart language, which is an easy language to learn Dart language is a programming language for many applications and provides a bunch of platforms. It is created by Google and usually uses to produce servers, desktops, web applications, and many more useful tools used in devices.

4- **Flutter is open source :** Roughly every three months, a new stable Flutter release is published, including thousands of contributions from the community and the Flutter team. This means that Flutter is constantly improving to meet the developer's needs. While Flutter is evolving fast, it remains remarkably stable. Quality and performance have been high since day one, and severe issues are always addressed in a very timely manner.

that Flutter software helps to develop applications smoothly while operating in devices. Dart is a programming language that provides many services and users to code apps like Flutter. The usage of Dart in Flutter makes it more convenient to perform a specific function in the device. It also eases the development process of some applications.

## 5- Code:

### 1-login page:

In this picture define class login we define tow variables email and password and formkey is cookies variable

And function instate check from the cookies type of user to choose the page to open him .

```
class _LoginState extends State<Login> {
  //variables
  TextFieldModel email, password;
  final formKey = GlobalKey<FormState>();
  final authProvider = AuthProvider();

  //init functions
  @override
  void initState() {
    Future.delayed(Duration.zero, () async {
      if (authProvider.check()) {
        switch (authProvider.accountType()) {
          case AccountType.security:
            pushClear(context, QrScanner());
            break;
          case AccountType.manager:
            pushClear(context, HomeManager());
            break;
          case AccountType.employee:
            pushClear(context, Home());
            break;
        }
      }
    }); // Future.delayed
    super.initState(); //Called whenever the widget configuration changes
  }
}
```

In this page define text filed to tack email and password from user on submit check if the email or password is true or not.

```
void initModels() {
  email = email ??
    TextFieldModel(
      errorMsg: 'please enter valid email',
      keyboardType: TextInputType.emailAddress,
      label: 'email',
      regex: emailRegex,
      width: width(context, .6),
      onSubmit: () => FocusScope.of(context).nextFocus(),
    );

  password = password ??
    TextFieldModel(
      errorMsg: 'please enter valid password',
      keyboardType: TextInputType.visiblePassword,
      label: 'password',
      regex: passwordRegex,
      width: width(context, .6),
      onSubmit: () => FocusScope.of(context).unfocus(),
    );
}

//logic functions
void _login() {
  if (!formKey.currentState.validate()) return;
  authProvider?.login(
    email.text,
    password.text,
    (state, {error}) {

```

In login function first check if any cookies found if not find cookies tack email and password from user and check if any error happen or not and go to page by email type if security email go to QrScanner page if Students Affairs go to Home page.

```

//logic functions
void _login() {
  if (!formKey.currentState.validate()) return;
  authProvider?.login(
    email.text,
    password.text,
    (state, {error}) {
      switch (state) {
        case AuthProviderState.start:
          BotToast.showLoading();
          break;
        case AuthProviderState.success:
          if (authProvider.accountType() == AccountType.security)
            pushClear(context, QrScanner());
          else
            pushClear(context, Home());
          break;
        case AuthProviderState.fail:
          BotToast.showSimpleNotification(
            title: 'error please try again later',
            subTitle: error,
            duration: Duration(seconds: 3),
          );
          break;
        case AuthProviderState.finish:
          BotToast.closeAllLoading();
          break;
      }
    }
  );
}

```

In this part we define the text filed and button and location and size any information about the icon.

```

@override
Widget build(BuildContext context) {
  initModels();
  return Scaffold(
    body: SafeArea(
      child: Container(
        width: double.infinity,
        child: ScrollableColumn(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Image.asset(
              'assets/logo.png',
              height: height(context, 0.4),
            ), // Image.asset
            Form(
              key: formKey,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  MyTextField(model: email),
                  SizedBox(height: 10),
                  MyTextField(model: password),
                  SizedBox(height: 20),
                  MyButton(text: 'LogIn', onPressed: _login),
                  SizedBox(height: 60),
                  MyButton(
                    text: 'LogIn as Student ? ',
                    onPressed: () {
                      push(context, QrGenerator());
                    }
                  )
                ]
              )
            )
          ]
        )
      )
    )
  );
}

```



## 2- Qr\_scanner (security page):

In this picture we build the button and location of button and division the screen we have two button sign out button and scan button if press to sign out this delete cookies.

```

lib > src > screen > qr_scanner.dart > QrScanner

class QrScanner extends StatelessWidget {
  final auth = AuthProvider();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.start,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Expanded(
              child: Row(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: MyButton(
                      text: 'sign out',
                      onPressed: () async {
                        await auth.signOut();
                        pushClear(context, Login());
                      },
                    ), // MyButton
                  ) // Padding
                ],
            ), // Row // Expanded
            Expanded(
              child: Row(
                mainAxisAlignment: MainAxisAlignment.center

```

If we press in scan button we go to start\_scanning function this scan qr code and show if this code find in fire base or not and go to student viewer class.

```

lib > src > screen > qr_scanner.dart > ...

      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          MyButton(
            text: 'start scanning',
            onPressed: () => startScanning(context),
          ), // MyButton
        ],
      ), // Row
    ), // Expanded
  ), // Column
), // SafeArea
); // Scaffold
}

void startScanning(BuildContext context) async {
  try {
    var data = await FlutterBarcodeScanner.scanBarcode(
      "#ff6666", "Cancel", false, ScanMode.DEFAULT);

    if (data == null) {
      BotToast.showSimpleNotification(title: 'please try again later');
      return;
    }
    push(context, StudentViewer(data));
  } catch (e) {
    print(e.toString());
    BotToast.showSimpleNotification(title: 'please try again later');
  }
}

```



### 3- Student\_Viewer: when the security scan code from student

In this picture we check if the qr code is are find in Firebase or not if find this code show all information about this student.

```

lib > src > screen > student_viewer.dart > StudentViewer
class StudentViewer extends StatelessWidget {
  final String studentCode;
  final store = StoreProvider();

  StudentViewer(this.studentCode);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: FutureBuilder(
          future: store.getStudentById(studentCode),
          builder: (context, AsyncSnapshot<DocumentSnapshot> snapshot) {
            if (!snapshot.hasData)
              return Center(
                child: Text('loading ...'),
              ); // Center
            else {
              var doc = snapshot.data;
              if (doc.exists) {
                var event = EventTracker(AuthProvider().uid, studentCode,
                  EventType.allow.toString(), Timestamp.now()); // EventTracker
                store.addEvent(event);

                return StudentInfoCard(
                  student: Student.fromDocumentSnapshot(doc),
                ); // StudentInfoCard
              } else {
                return Center(
                  child: Container(
                    width: 200,

```

If the student don't find in Firebase show error picture.

```

lib > src > screen > student_viewer.dart > StudentViewer
); // Center
else {
  var doc = snapshot.data;
  if (doc.exists) {
    var event = EventTracker(AuthProvider().uid, studentCode,
      EventType.allow.toString(), Timestamp.now()); // EventTracker
    store.addEvent(event);

    return StudentInfoCard(
      student: Student.fromDocumentSnapshot(doc),
    ); // StudentInfoCard
  } else {
    return Center(
      child: Container(
        width: 200,
        height: 200,
        decoration:
          BoxDecoration(color: red, shape: BoxShape.circle),
        child: GestureDetector(
          onTap: () => pop(context),
          child: Icon(
            Icons.error,
            size: 50,
            color: white,
          ), // Icon
        ), // GestureDetector
      ), // Container
    ); // Center
  }
}

```

#### 4- Qr-Generator

This code for generate qr code for student by enter ssn+student-code

```

login.dart home.dart qr_generator.dart X auth_provider.dart
lib > src > screen > qr_generator.dart > QrGenerator

class QrGenerator extends StatefulWidget {
  @override
  _QrGeneratorState createState() => _QrGeneratorState();
}

class _QrGeneratorState extends State<QrGenerator> {
  String qrData;

  @override
  void initState() {
    Future.delayed(Duration.zero, () async {
      var pref = SharedProvider();
      qrData = await pref.getLastData();
      setState(() {
        if(qrData != null) qrModel.text = qrData;
      });
    }); // Future.delayed
    super.initState();
  }

  TextFieldModel qrModel;

  @override
  Widget build(BuildContext context) {
    qrModel = qrModel ??
      TextFieldModel(
        label: 'qr data',
        onSubmit: () => FocusScope.of(context).unfocus(),
        regex: '^[0-9]{14,}\\$',
        errorMsg: '',
      );
  }
}

```

Here this code check if enter correct ssn+student\_code don't enter character only enter number.

```

login.dart home.dart qr_generator.dart X auth_provider.dart
lib > src > screen > qr_generator.dart > QrGenerator

@override
Widget build(BuildContext context) {
  qrModel = qrModel ??
    TextFieldModel(
      label: 'qr data',
      onSubmit: () => FocusScope.of(context).unfocus(),
      regex: '^[0-9]{14,}\\$',
      errorMsg: '',
      width: width(context, 0.7),
      keyboardType: TextInputType.number,
    );

  return Scaffold(
    body: SafeArea(
      child: Container(
        width: double.infinity,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            if (qrData != null)
              BarcodeWidget(
                barcode: Barcode.qrCode(),
                color: black,
                data: qrData,
                width: 200,
                height: 200,
              ), // BarcodeWidget
            if(qrData != null) SizedBox(height: 50),
          ],
        ),
      ),
    ),
  );
}

```

In this code if press button generate go to generate-code function first check for text is not encloude any thing without number and generate qr code.

```

lib > src > screen > qr_generator.dart > QrGenerator
// // bar code widget
if(qrData != null) SizedBox(height: 50),
Row(
  children: [
    Expanded(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: MyTextField(
          model: qrModel,
        ), // MyTextField
      ), // Padding
    ), // Expanded
    MyButton(text: 'Generate', onPressed: generateCode,)
  ],
), // Row
],
), // Column
), // Container
), // SafeArea
); // Scaffold
}

void generateCode() {
  if(!RegExp(qrModel.regex).hasMatch(qrModel.text)){
    BotToast.showSimpleNotification(title: 'please enter valid qr data', subTitle: 'your ssn + your student code');
    return;
  }
  qrData = qrModel.text;
  SharedProvider().setData(qrData);
  setState() {

```

### 5- Home page for employ:

This page compile when the user enter Students Affairs email first we see the search text field in this field enter code student and the program show all information about student and logout icon.we define variable students to store the student with the id code .

```

lib > src > screen > home.dart > _HomeState > build
class _HomeState extends State<Home> {
  TextFieldModel search;
  var students = <Student>[];
  final auth = AuthProvider();
  final store = StoreProvider();

  @override
  Widget build(BuildContext context) {
    search = search ??
    TextFieldModel(
      label: 'student code',
      onSubmit: () => FocusScope.of(context).unfocus(),
      width: double.infinity,
      keyboardType: TextInputType.number,
      errorMsg: null,
      regex: null,
    );

    return Scaffold(
      appBar: PreferredSize(
        preferredSize: Size.fromHeight(100),
        child: SafeArea(
          child: Container(
            margin: EdgeInsets.all(8),
            child: Row(
              children: [
                IconButton(
                  icon: Icon(Icons.logout),
                  onPressed: () async {
                    await auth.signOut();

```

```
login.dart home.dart auth_provider.dart
lib > src > screen > home.dart > _HomeState > build
icon: Icon(Icons.logout),
onPressed: () async {
  await auth.signOut();
  pushClear(context, Login());
}, // IconButton
Expanded(
  child: MyTextField(
    model: search,
  ), // MyTextField
), // Expanded
IconButton(
  icon: Icon(Icons.search),
  onPressed: () {
    FocusScope.of(context).unfocus();
    var searchCode = search.text.trim();
    if (searchCode.isEmpty) return;
    var student = students.singleWhere(
      (element) => element.code == searchCode,
      orElse: () => null);
    if (student == null) {
      BotToast.showSimpleNotification(
        title: 'Student Not Found');
      return;
    }
    push(
      context,
      ManageStudent(
        student: student,
      )), then((value) { // ManageStudent
      setState(() {});
    });
  }
);
```

```
login.dart  home.dart  auth_provider.dart
lib > src > screen > home.dart > _HomeState > build

floatingActionButton: FloatingActionButton(
  child: Icon(Icons.add),
  onPressed: () {
    push(context, ManageStudent(student: Student())).then((value) {
      setState(() {});
    });
  },
), // FloatingActionButton
body: SafeArea(
  child: FutureBuilder(
    future: store.getAllStudents(),
    builder: (context, AsyncSnapshot<List<Student>> snapshot) {
      if (snapshot.hasData) {
        students = snapshot.data;
        if (students.isEmpty)
          return Center(
            child: Text('there is no students in database yet'),
          ); // Center
        return ListView.builder(
          itemCount: students.length,
          itemBuilder: (context, index) {
            var student = students[index];
            return StudentCard(
              student: student,
              delete: () {
                showDialog(context: context, builder: (context) {
                  return SimpleDialog(
                    contentPadding: EdgeInsets.only(top: 8, bottom: 0, left: 8, right: 8),
                    children: [
                      Container(

```



In this code when the employ select any student from app the app Lets to the employ to update and delete student or add student.

```

lib > src > screen > home.dart > _HomeState > build
children: [
  Text('Are You Sure ?', style: TextStyle(fontSize: 25)),
  SizedBox(height: 8),
  Text('delete student \${student.name}\'),
  SizedBox(height: 2),
  Text('with student code \${student.code}\'),
  FlatButton(
    alignment: MainAxisAlignment.end,
    children: [
      FlatButton(
        onPressed: () {
          store.deleteStudent(student.id).then((value) {
            setState(() {});
            pop(context);
          });
        }, child: Text('yes'), // FlatButton
      ), FlatButton(onPressed: () => pop(context), child: Text('no')),
    ], // FlatButton
  ), // FlatButton
), // Column
), // Container
), // SimpleDialog
});
edit: () {
  push(context, ManageStudent(student: student))
    .then((value) {
      setState(() {});
    });
}, // StudentCard
}); // ListView.builder
} else
  return Center(child: Text('Loading ...'));
},

```

## 6- AuthProvider page to connect to firebase and login into app.

This code for connect to firebase and login into app or logout from app. First define variable to Conect to firebase and check if any user login before or not .if not tack email and password log in into firebase check type of email and change type of account.

```

lib > src > provider > auth_provider.dart > AuthProvider
class AuthProvider {
  var _auth = FirebaseAuth.instance;

  bool check() => _auth.currentUser != null;

  String get uid => _auth.currentUser.uid;

  void login(
    String email,
    String password,
    Function(AuthProviderState, {String error}) observe,
  ) {
    async {
      try {
        observe(AuthProviderState.start);
        await _auth.signInWithEmailAndPassword(email: email, password: password);
        observe(AuthProviderState.success);
      } catch (e) {
        observe(AuthProviderState.fail, error: e.toString());
      } finally {
        observe(AuthProviderState.finish);
      }
    }
  }

  AccountType accountType() => _auth.currentUser.email.contains('sec')
    ? AccountType.security
    : _auth.currentUser.email.contains('manager')
    ? AccountType.manager
    : AccountType.employee;
}

```

In this if user need to sign-out we delete connection between app and firebase and we define enum to account type and AuthProvider state .

```

login.dart  home.dart  auth_provider.dart X
lib > src > provider > auth_provider.dart > AuthProvider

AccountType accountType() => _auth.currentUser.email.contains('sec')
    ? AccountType.security
    : auth.currentUser.email.contains('manager')
    ? AccountType.manager
    : AccountType.employee;

Future<void> signOut() => _auth.signOut();

enum AccountType {
  security,
  manager,
  employee,
}

enum AuthProviderState {
  start,
  success,
  fail,
  finish,
}

```

## 7- StoreProvider this class responsible to read and write from firebase.

This enable app to access to firebase to add or delete or update student in this picture we see function upload new student to upload student information to firebase.

```

login.dart  home.dart  store_provider.dart X
lib > src > provider > store_provider.dart > StoreProvider > editStudentData

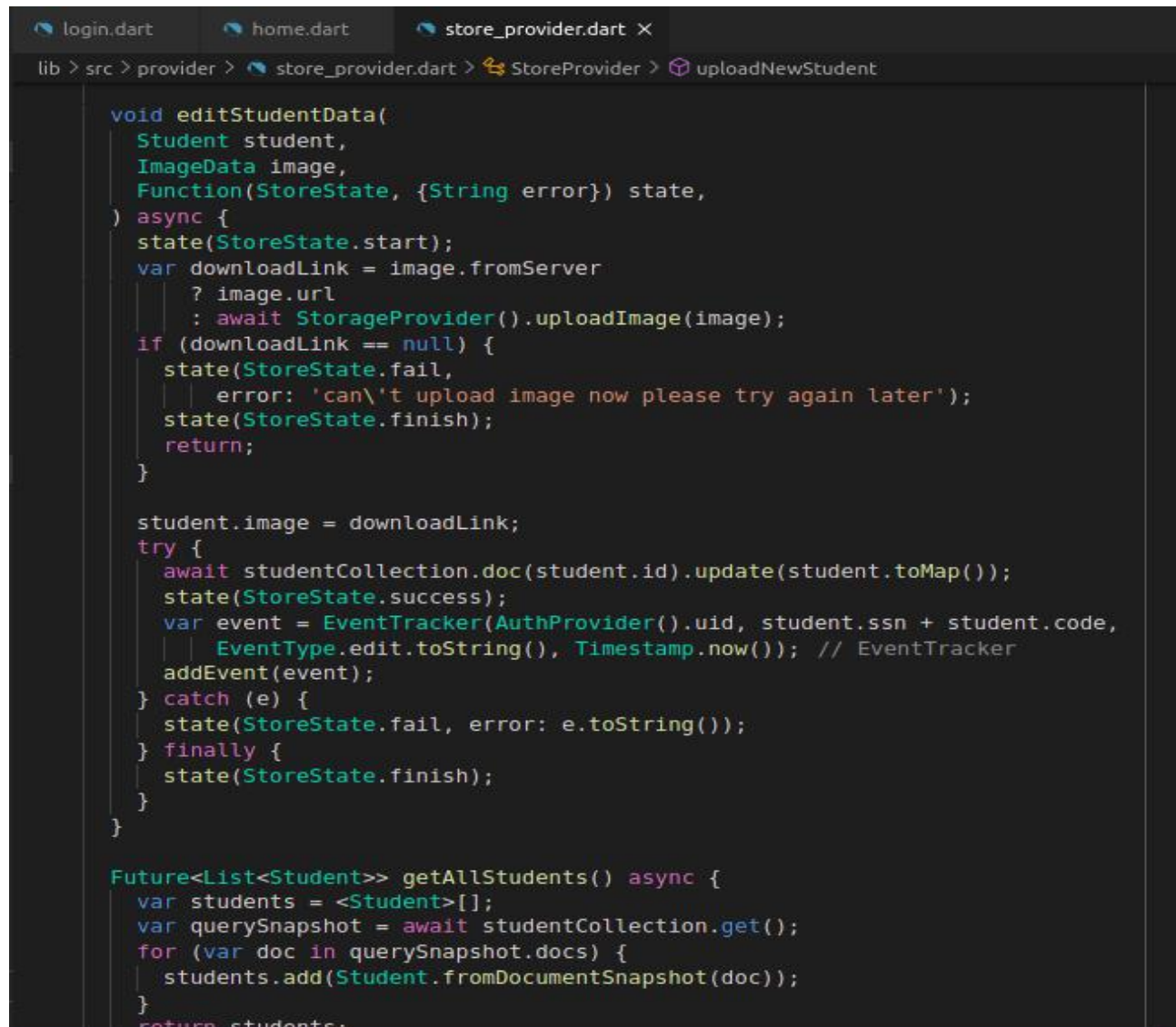
class StoreProvider {
  var _store = FirebaseFirestore.instance;
  CollectionReference get studentCollection => _store.collection('student');
  CollectionReference get eventCollection => _store.collection('event');

  void uploadNewStudent(
    Student student,
    ImageData image,
    Function(StoreState, {String error}) state,
  ) async {
    state(StoreState.start);
    var downloadLink = await StorageProvider().uploadImage(image);
    if (downloadLink == null) {
      state(StoreState.fail,
        error: 'can\'t upload image now please try again later');
      state(StoreState.finish);
      return;
    }

    student.image = downloadLink;
    try {
      await studentCollection
        .doc(student.ssn + student.code)
        .set(student.toMap());
      var event = EventTracker(AuthProvider().uid, student.ssn + student.code,
        EventType.add.toString(), Timestamp.now()); // EventTracker
      addEvent(event);
      state(StoreState.success);
    } catch (e) {
      state(StoreState.fail, error: e.toString());
    } finally {
      state(StoreState.finish);
    }
  }
}

```

This edit function first check if employ change picture of student if change tack the URL and add to student information and upload new data .

A screenshot of an IDE window showing Dart code. The top bar has tabs for 'login.dart', 'home.dart', and 'store\_provider.dart'. The breadcrumb navigation shows the path: 'lib > src > provider > store\_provider.dart > StoreProvider > uploadNewStudent'. The code defines an 'editStudentData' function and a 'getAllStudents' function. The 'editStudentData' function takes a 'Student' object, an 'ImageData' object, and a 'state' object as parameters. It first checks if the 'image' has a 'url' and if it's not null. If it is, it updates the 'student.image' with the 'downloadLink'. If not, it shows an error message. Then, it updates the 'studentCollection.doc' with the 'student.toMap()' and adds an event to the 'EventTracker'. The 'getAllStudents' function is an asynchronous function that returns a 'Future<List<Student>>'. It fetches the 'studentCollection.get()' and iterates over the 'docs' to add them to the 'students' list.

```
void editStudentData(
  Student student,
  ImageData image,
  Function(StoreState, {String error}) state,
) async {
  state(StoreState.start);
  var downloadLink = image.fromServer
    ? image.url
    : await StorageProvider().uploadImage(image);
  if (downloadLink == null) {
    state(StoreState.fail,
      error: 'can\'t upload image now please try again later');
    state(StoreState.finish);
    return;
  }

  student.image = downloadLink;
  try {
    await studentCollection.doc(student.id).update(student.toMap());
    state(StoreState.success);
    var event = EventTracker(AuthProvider().uid, student.ssn + student.code,
      EventType.edit.toString(), Timestamp.now()); // EventTracker
    addEvent(event);
  } catch (e) {
    state(StoreState.fail, error: e.toString());
  } finally {
    state(StoreState.finish);
  }
}

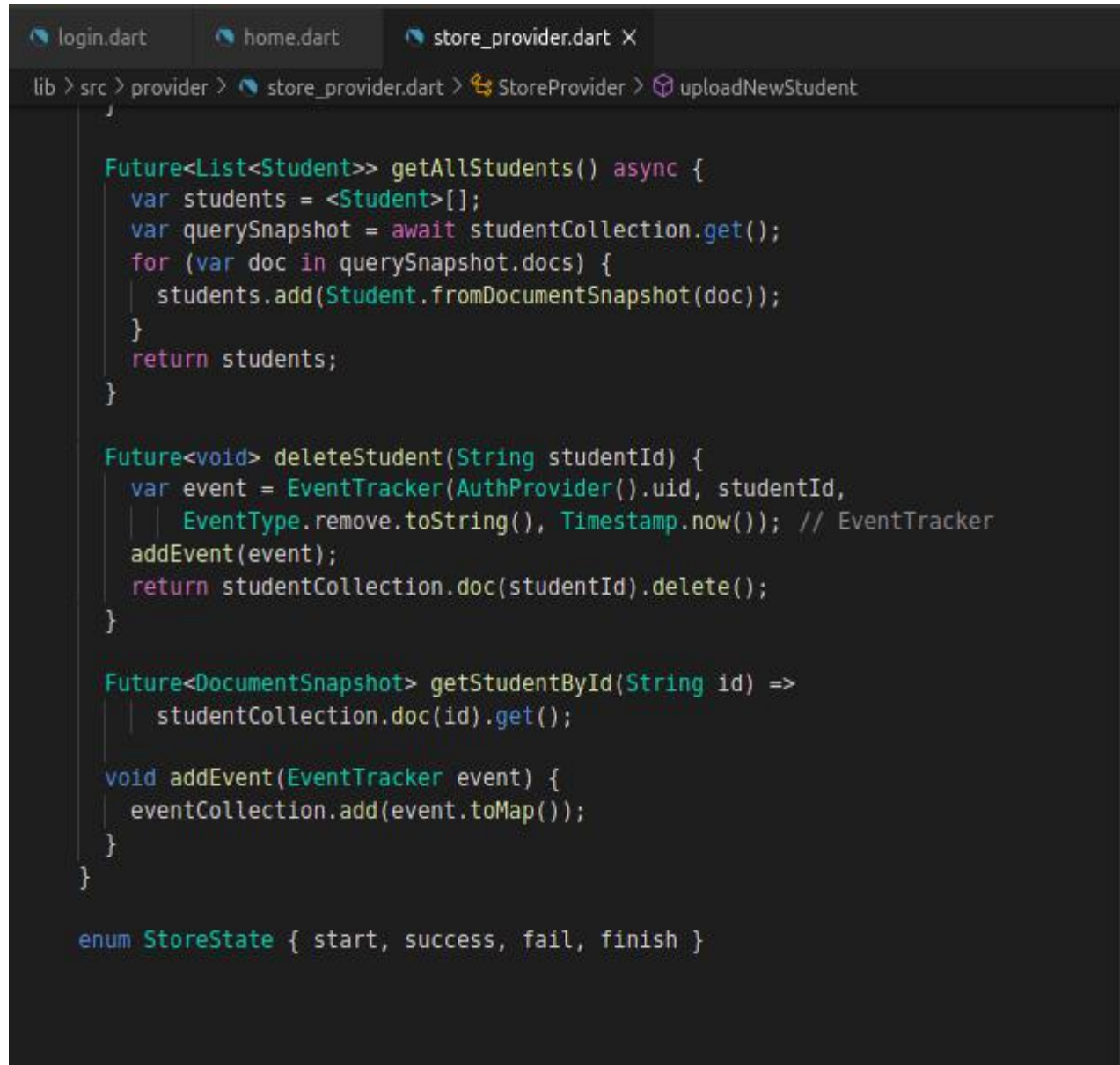
Future<List<Student>> getAllStudents() async {
  var students = <Student>[];
  var querySnapshot = await studentCollection.get();
  for (var doc in querySnapshot.docs) {
    students.add(Student.fromDocumentSnapshot(doc));
  }
  return students;
}
```

Function get all student return list of student get all student and show in app.

Function delete student delete by id of student.

Function get student for search icon this by id.

Function add event to store everything an employee does.



```
lib > src > provider > store_provider.dart > StoreProvider > uploadNewStudent

Future<List<Student>> getAllStudents() async {
  var students = <Student>[];
  var querySnapshot = await studentCollection.get();
  for (var doc in querySnapshot.docs) {
    students.add(Student.fromDocumentSnapshot(doc));
  }
  return students;
}

Future<void> deleteStudent(String studentId) {
  var event = EventTracker(AuthProvider().uid, studentId,
    EventType.remove.toString(), Timestamp.now()); // EventTracker
  addEvent(event);
  return studentCollection.doc(studentId).delete();
}

Future<DocumentSnapshot> getStudentById(String id) =>
  studentCollection.doc(id).get();

void addEvent(EventTracker event) {
  eventCollection.add(event.toMap());
}

enum StoreState { start, success, fail, finish }
```