GroupGram

social media

Helwan University

Faculty of Computers and artificial intelligence
Information Systems Department

Supervised by:
Dr.Doaa Saad

Team members:

1- Mohamed Nabil Hemid Salem (20170476)

2- Mostafa Ahmed Gomaa Mohamed (20170528)

3- Mina Youssef Alber George (20170583)

4- Nagy Mostafa Hussein Ibrahim (20170584)

5- Momen Ali Assar (20170572)

6- Mohamed Hamdy Saber (20170435)

Acknowledgement

First, we would express our deepest gratitude and appreciation to our advisor Dr.Doaa Saad for her support, outstanding guidance and encouragement throughout our graduation project and her continuous support and help as she was always there for us providing more than the needed from her since the very beginning of our project.

We would also like to thank our families, especially our parents, for their encouragement, patience, and assistance over the years. We are forever indebted to our parents, who have always kept us in their prayers and pushed us to Success. Finally, for our faculty for providing the suitable environment that leaded us to represent the best image that computer science graduates of Helwan University are supposed to represent

Table of Contents

Abstract

Main idea:

This application is made to help people communicate with each other easily and exchange information about anything they want to, and since communication affect our lives in a great way then this application is very useful and important to all people who uses mobile phones today, the main focus of this application is making communication easier for people by providing the suitable environment for them. This application also has a shop, chat which support sending instant messages to other people, navigation page which allow people to help each other by opening google maps application to know the exact locations of people who needs help then I can go to help him with his problem

Structure:

A social media application with advanced technologies like flutter and firebase to make people communicate and help each other with their car problems

Technical architecture:

1- Application service – Android & IOS
2- Flutter framework
3- Firebase firestore
4- Firebase storage
5- Google maps

- Our application is built with flutter which is portable for high-quality mobile applications. It implements flutter's core libraries including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a dart runtime and toolchain for developing, compiling, and running flutter applications

- Supporting it with firebase which have a database that provides a real-time database and backend and Firebase Auth that can authenticate users using only client-side code

- We are using Android Studio for developing

1. Introduction

In this chapter, we will discuss the project's objectives, purpose, scope of the project, and work needed to build the project

Overview:

This application is a social media application which allow people to communicate with each other easily by following each other, making posts, commenting on posts, liking posts, chatting with each other, and there is a shop which allow users to sell or buy anything from it

Project motivation:

- The reason behind developing this application is to make communication between people easier and faster than before

- This application is important because it deals with a critical issue and facilitates the process of communication between people

- The new idea in this project is creating an environment which facilitates communication and also allowing people who have a problem in their cars to make a post with their location so that anyone using the application and has the ability to go and help will go and solve the problem for him

Problem statement:

This application helps anyone who have a car problem to solve his problem easily by communicating with other people which makes the process of solving car problems easier and faster than before, also the application facilitates buying and selling cars by providing a car shop where people can post their cars for sale or can search for cars to buy, the conditions that must be improved upon this project is facilitating the process of communication between people and especially people with problems in their cars so that all people who is interested in cars can communicate with each other and exchange their ideas and experiences

Project aim and objectives:

- The main goal that this application wants to achieve is to create an environment where all people can communicate with each other easily

- This project can achieve this goal by creating an application that can be used by people to communicate with each other by making posts that other people can interact with, making posts in the shop if a person wants to sell his car, or if you have a critical problem with your car then you can make posts which include your current location so that anyone who have the ability to help you can easily know your location and come to help you with solving your problem

Project scope:

This application creates an environment where all people can communicate with each other easily by making posts, commenting on posts, liking posts, sending messages, also the users can ask for help if their car crashed suddenly, and if you have the ability to help someone to solve his car problem then you can help him easily because the application will show you his location so that you can go and help him by solving the problem. The user can login into the application and perform tasks such as making a post, see other people's posts, see the problems written by other users in the application, see the shop if he wants to buy or sell anything, and editing his profile information

The application contains two modules: User – Admin

Project Software and Hardware Requirements:

Hardware requirements:

The application needs direct hardware device which is a mobile

Software requirements:

The hardware uses the android operating system to run the application

Project limitations:

The limitations of this application are:

- The problems section is only for users with problems in cars
- The application doesn't provide help in all kinds of problems

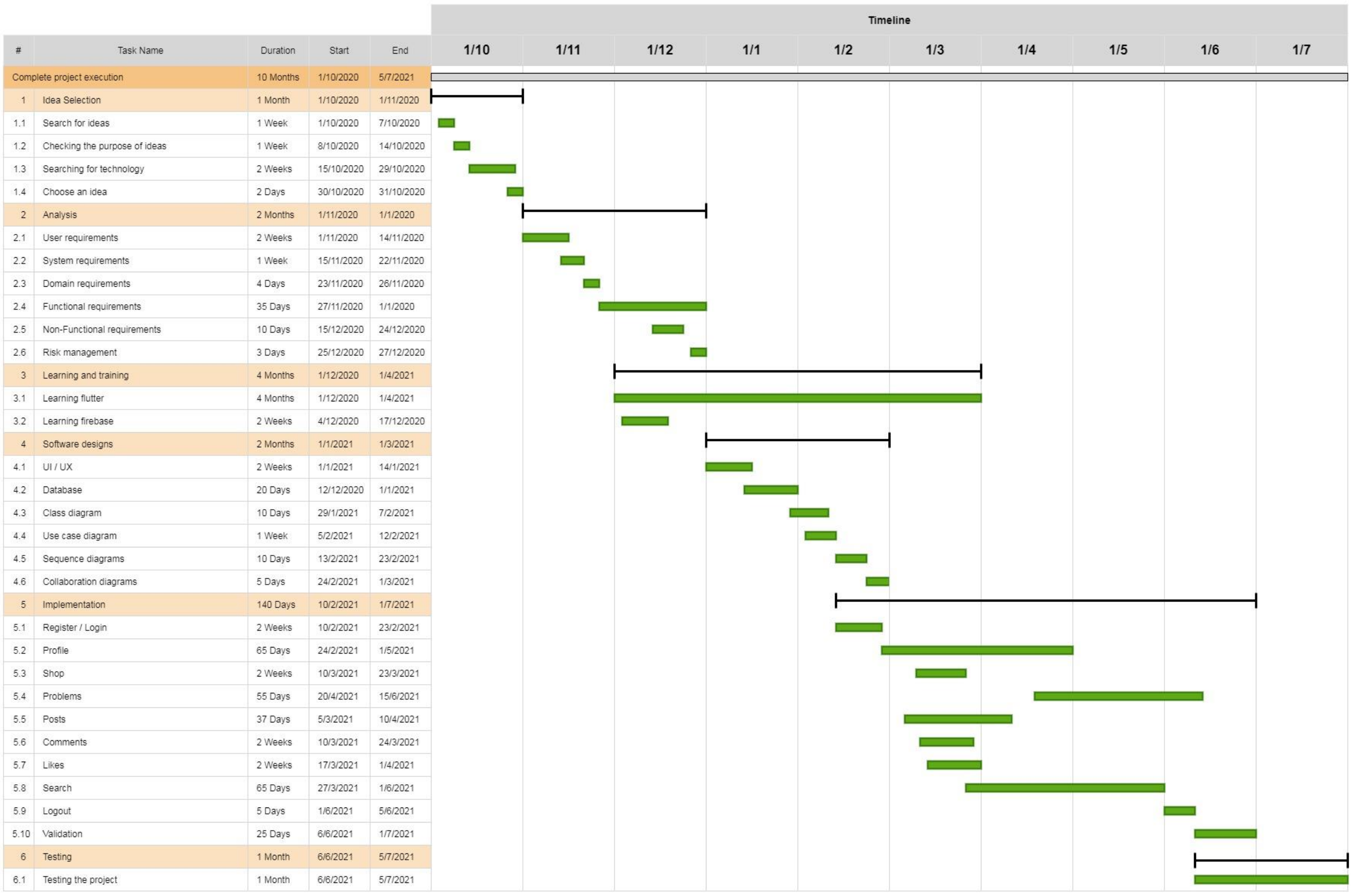
These limitations can be improved by adding more functionalities and features to the application

Project expected output:

The desired output of the project is to make a social network where all people can communicate with each other easily, also it includes a shop where people can buy or sell anything. This application includes a section which allows people who have problems in their cars to solve their problems by asking for help from other people

The project milestone is to create an environment which facilitates the process of communication between all people, selling or buying anything, also the project will cover the scope of helping others in solving their car problems

Project schedule:



#	Task Name	Duration	Start	End
Complete project execution		10 Months	1/10/2020	5/7/2021
1	Idea Selection	1 Month	1/10/2020	1/11/2020
1.1	Search for ideas	1 Week	1/10/2020	7/10/2020
1.2	Checking the purpose of ideas	1 Week	8/10/2020	14/10/2020
1.3	Searching for technology	2 Weeks	15/10/2020	29/10/2020
1.4	Choose an idea	2 Days	30/10/2020	31/10/2020
2	Analysis	2 Months	1/11/2020	1/1/2020
2.1	User requirements	2 Weeks	1/11/2020	14/11/2020
2.2	System requirements	1 Week	15/11/2020	22/11/2020
2.3	Domain requirements	4 Days	23/11/2020	26/11/2020
2.4	Functional requirements	35 Days	27/11/2020	1/1/2020
2.5	Non-Functional requirements	10 Days	15/12/2020	24/12/2020
2.6	Risk management	3 Days	25/12/2020	27/12/2020
3	Learning and training	4 Months	1/12/2020	1/4/2021
3.1	Learning flutter	4 Months	1/12/2020	1/4/2021
3.2	Learning firebase	2 Weeks	4/12/2020	17/12/2020
4	Software designs	2 Months	1/1/2021	1/3/2021
4.1	UI / UX	2 Weeks	1/1/2021	14/1/2021
4.2	Database	20 Days	12/12/2020	1/1/2021
4.3	Class diagram	10 Days	29/1/2021	7/2/2021
4.4	Use case diagram	1 Week	5/2/2021	12/2/2021
4.5	Sequence diagrams	10 Days	13/2/2021	23/2/2021
4.6	Collaboration diagrams	5 Days	24/2/2021	1/3/2021
5	Implementation	140 Days	10/2/2021	1/7/2021
5.1	Register / Login	2 Weeks	10/2/2021	23/2/2021
5.2	Profile	65 Days	24/2/2021	1/5/2021
5.3	Shop	2 Weeks	10/3/2021	23/3/2021
5.4	Problems	55 Days	20/4/2021	15/6/2021
5.5	Posts	37 Days	5/3/2021	10/4/2021
5.6	Comments	2 Weeks	10/3/2021	24/3/2021
5.7	Likes	2 Weeks	17/3/2021	1/4/2021
5.8	Search	65 Days	27/3/2021	1/6/2021
5.9	Logout	5 Days	1/6/2021	5/6/2021
5.10	Validation	25 Days	6/6/2021	1/7/2021
6	Testing	1 Month	6/6/2021	5/7/2021
6.1	Testing the project	1 Month	6/6/2021	5/7/2021

Project, product, and schedule risks:

If any defect is found while testing the overall system after finishing the development phase, we will not be able to return and solve it because of the lack of time, and if this defect doesn't affect the main functionalities in the application then we will write a list by this defects but will not solve it, and if this defect has a negative effect on the main functionalities then we will try to solve it during the production and in this case if the defect is critical then the application may fail and not perform what is required

2. Related existing systems

In this chapter, we will discuss the existing systems that perform the same functions or something related to the main functions of our application

Introduction:

In this chapter, we discuss the construction of the baseline models of related existing systems. This activity relies on knowledge of the software related to our system. It is common that all people wants to communicate with each other easily and also the very common scenario that someone's car might be broken down on his way, so in this situation he can't get a direct and quick solution because it's difficult to repair it with himself or get help to repair it in a short time. So in this case he can use the problems section in the application to contact someone to reach him and get the help, he would use two things to get the help through the application and these two things are:

- The GPS to know the current location
- Making a post with what happened to his car and this post will include his location so anyone using the application can come and help him in solving his problem

So this application is different from any other application because there is no specific application that does these things together efficiently and in a small amount of time like our application

Existing systems:

Social media applications:

Name	Description and audience	Features
Facebook	The most popular social network in the world but restricted in some countries like china, user must be at least 13 years	News feed, profile, friends, find friends, search, pages, groups, events, add photos, update status
Instagram	Social network which focus on sharing photos, innovative by using hashtags which help users spot on specific kinds of pictures that they wish to see	Profile, followers, following, posts, explore, search, shop, direct messaging
Twitter	Generally it is a social network for people who are willing to connect with other people to express their feelings and opinions about any subject in any field	Profile, followers, following, home, search, trending, hashtags, search, tweeting, retweeting, chat

Car repair applications:

Name	Description and audience	Features
Carefer	Free interactive electronic platform that helps you get the nearest car maintenance workshop, it also offers consulting and provide a service of receiving the client's car to maintain it professionally then getting it back to the client for a very small fee	Invite friends, wallet, add your workshop in the application, booking appointments, shock repair
Asale7lak	The company that owns the application is the first of its kind in the Arabic world to serve cars on the road, the company has a team that aims to revolutionize the world of car repair and deliver every vehicle owner to the ideal vehicle with only a click of a button	Comprehensive insurance, compulsory insurance, ask for a mechanic, fix your car, asking for a winch
Morni	A network to provide transportation services and roadside assistance, you can request their services when needed or subscribe for an annual membership that allows you to request services for free when needed in exchange for a nominal amount paid once for a full year	Transport, fuel delivery, battery, punching, vehicle locks opening, comprehensive solution service for accidents

Overall problems of existing systems:

- Social media applications like Facebook, Instagram, and Twitter focus on communication between the general public at large and not a specific niche or specialty

- Auto repair applications like Carefer, asale7lak, and Morni focus on fixing cars just by sending a mechanic to the scene of the accident, and mostly the application is a property of the repair center

- The problem here is that it does not benefit from people's communication helping each other and strengthening the bond between people

Overall solution approach:

Our solution approach represents two ideas:

- How to deal with a car breakdown in one single application that you should use to get help to repair your car when you are far away from your home

- Make an online community where you can share anything related to cars, also you can describe your problem and get a solution from another people using the application

In addition to:

- You can sell or buy a car
- You can sell or buy car's accessories

3. Analysis

In this chapter, we will discuss the analysis of our system from the point of view of all the stakeholders who will have an effect on the system

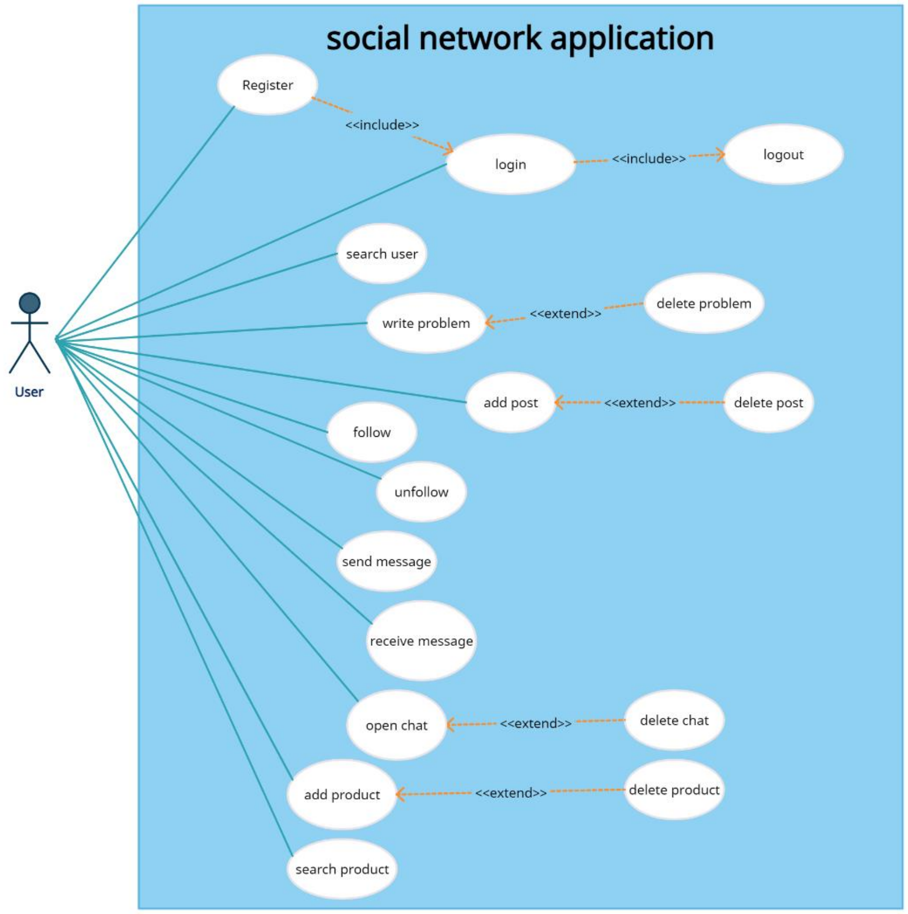
Stakeholders:

Primary stakeholders: users

Secondary stakeholders: competitors – advertisers – developers

Use case diagram:

A use case diagram is a graphical representation of a user's possible interactions with a system, a use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well, the use cases are represented by either circles or ellipses, the actors are often shown as stick figures



Functional requirements:

- **Register:** user can register with his google account
- **Login:** user can login with his google account
- **Add post:** users can make a post about anything they want to
- **Delete post:** users can delete their posts
- **Add comment:** users can add comments on the posts of people they follow
- **Delete comment:** users can delete their comments
- **Search user:** users can search for another users, follow them, send message to them, and see their posts
- **Follow:** users can follow anyone and the posts of the followed user will appear in the home page of the user
- **Unfollow:** users can unfollow anyone from their following list then the posts of the unfollowed user will disappear from the home page of the user
- **Send message:** users can send message to anyone
- **Receive message:** users can receive message from anyone
- **Delete chat:** users can delete any chat
- **Add product:** users can add a product for sale in the shop
- **Delete product:** users can delete a posted product from the shop
- **Search product:** users can search for a specific product
- **Add problem:** users can post a problem with their location in the problems section so that another users can find this person and help him
- **Delete problem:** users can delete their posted problems from the problems section if their problem is solved
- **Logout:** users can logout from their account

Non-functional requirements:

Execution qualities:

1- **Security:**
The system has high level of security because it uses passwords

2- **Usability:**
- Well-structured user manual
- Well-formed graphical user interface
- Simple and users can use and understand it without training
- Home pages is simple with clear buttons

Evolution qualities:

1- **Performance:**
- Scalability
- The capability of the system to increase the total throughput under pressure
- Increased load when resources (typically hardware) are added

2- **Maintainability:**
Our system is responsible for providing support, maintenance and services to users and there is guarantee for emergencies

3- **Portability:**
Portability is a characteristic attributed to a computer program if it can be used in operating systems different from the one in which it was created without requiring major network

4- **Testability:**
Used to understand whether your system meets quality constraints

5- **Extensibility:**
It is a characteristic where the architecture, design, and implementation actively caters to future business needs

6- **Scalability:**

- It is the ability of the application to handle an increase in the workload without performance degeneration
- It is the ability of the application to quickly enlarge

Constraints:

- The problems section is only for users who have problems with their cars
- Users can send message to anyone using the application without having to follow them

4. Software design

In this chapter, we will go deeper in the application's design, and present its diagrams and database

Overview:

This document consists of seven chapters:

- The first chapter contains a general description about the main idea of the project, project aim and objectives, project scope, project limitations, the project expected output, and the risks that may face us during the development phase of the project

- The second chapter talks about related existing systems that may have similar functionalities like our project, this chapter also talks about the problems that face these existing systems

- The third chapter talks about the analysis of our application which includes identifying all the stakeholders that deal or have an interest in the application, use case diagram, functional requirements, non-functional requirements, and the constraints that may affect the operation of our application

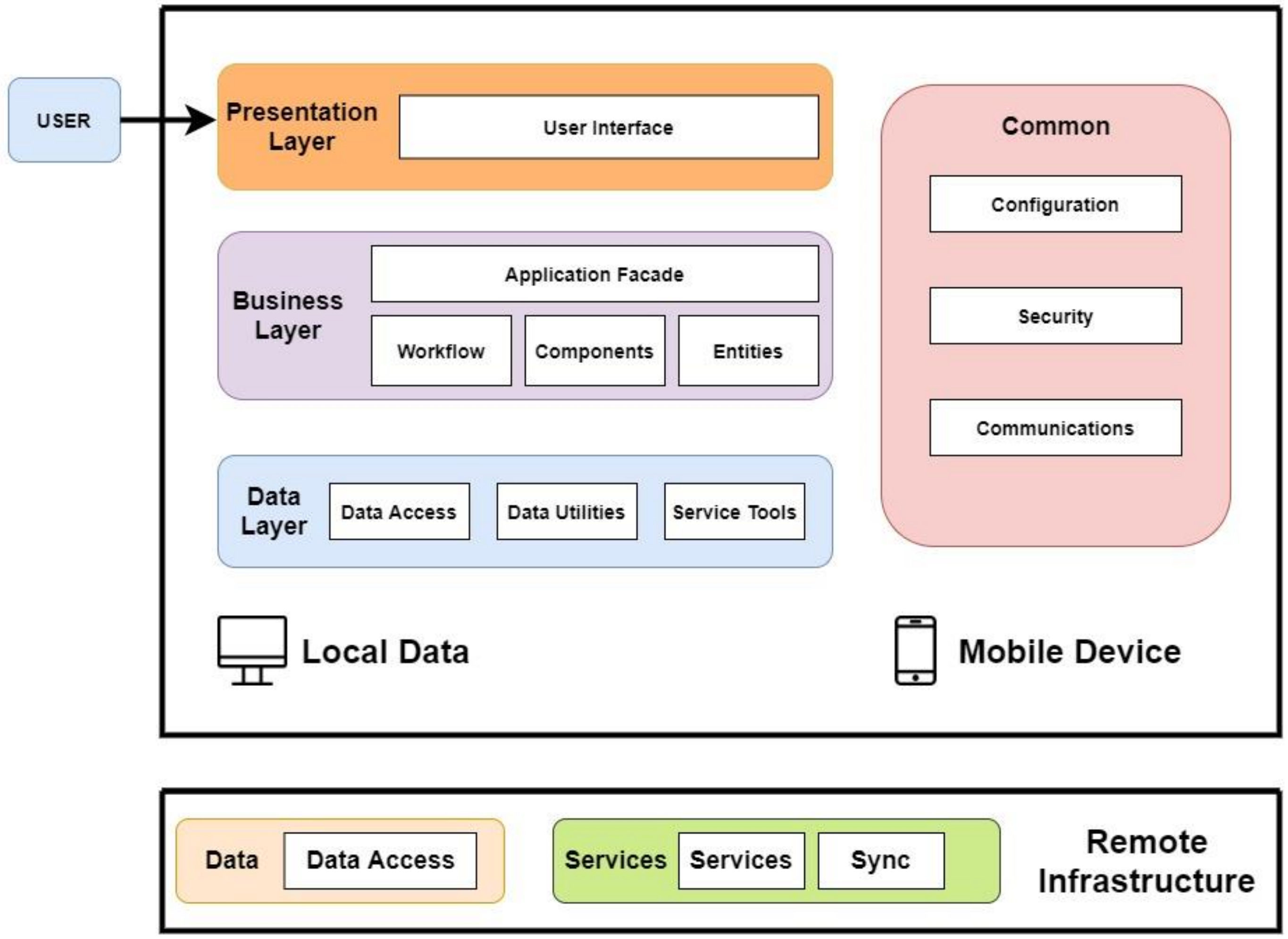
- The fourth chapter talks about software design which includes the software architecture of the system, UML diagrams like sequence and class diagram, database design which includes ER diagram and database schema (mapping), and this chapter also includes the prototype of our application which is the user interface design

- The fifth chapter talks about the details of implementation of the application which includes description of implementation, the used programming language and technology

- The sixth chapter talks about testing phase which includes black-box testing and white-box testing of the application to see if there are any errors so that we can fix it before the production phase

- The seventh chapter includes a conclusion of the overall details of the application and the results that we reached upon this application

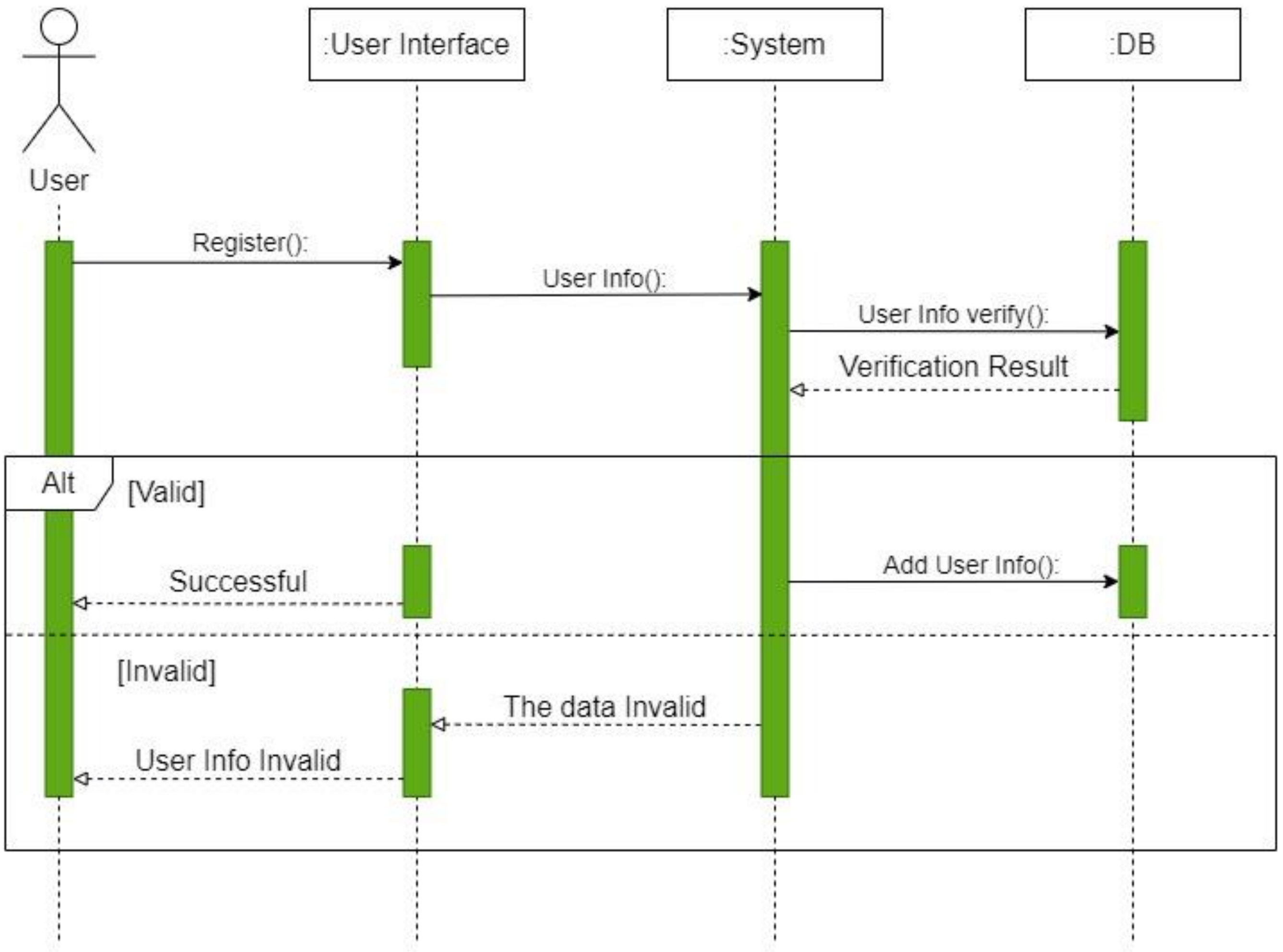
Software architecture:



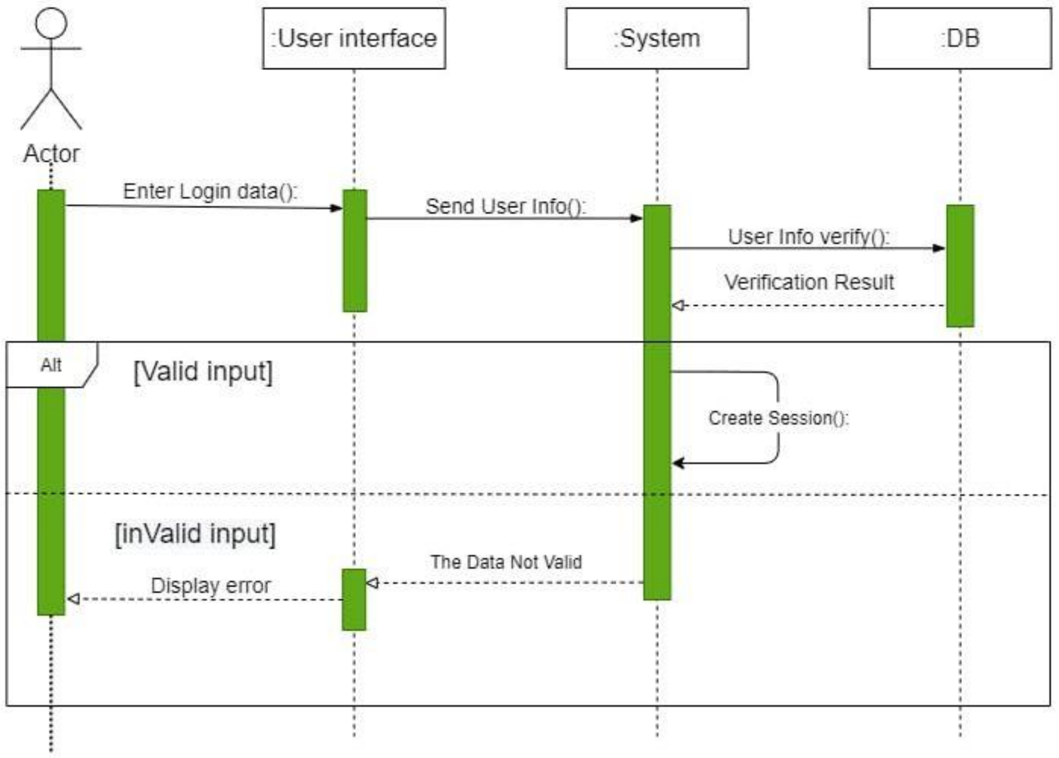
UML Diagrams:

UML Sequence:

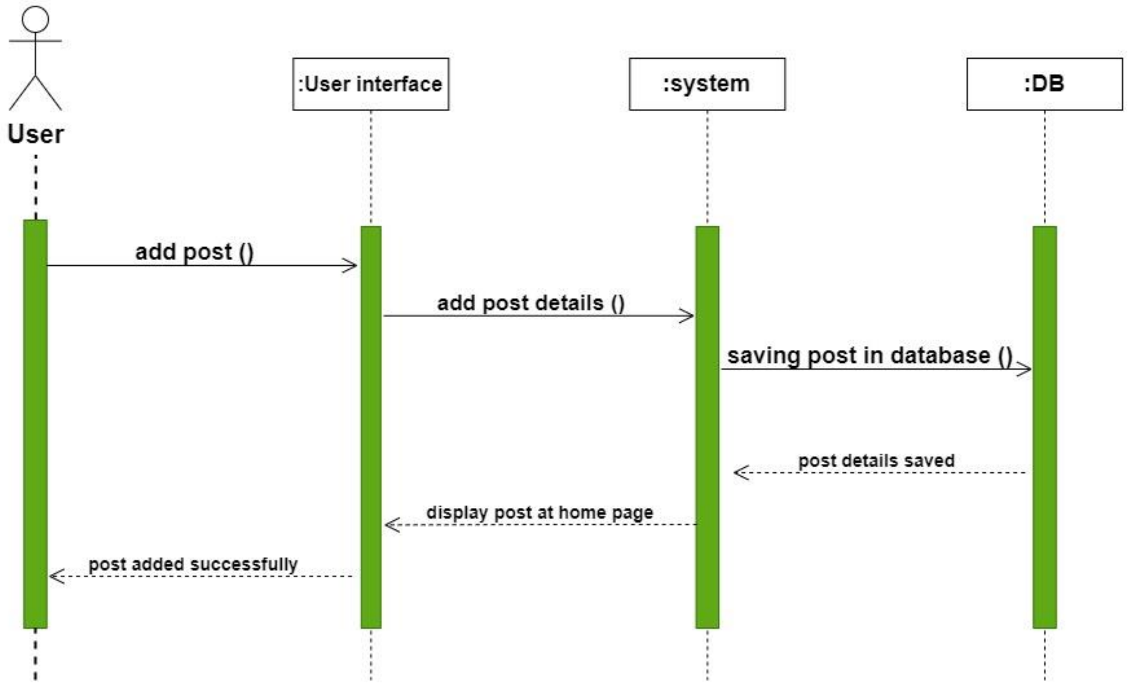
1- Register:



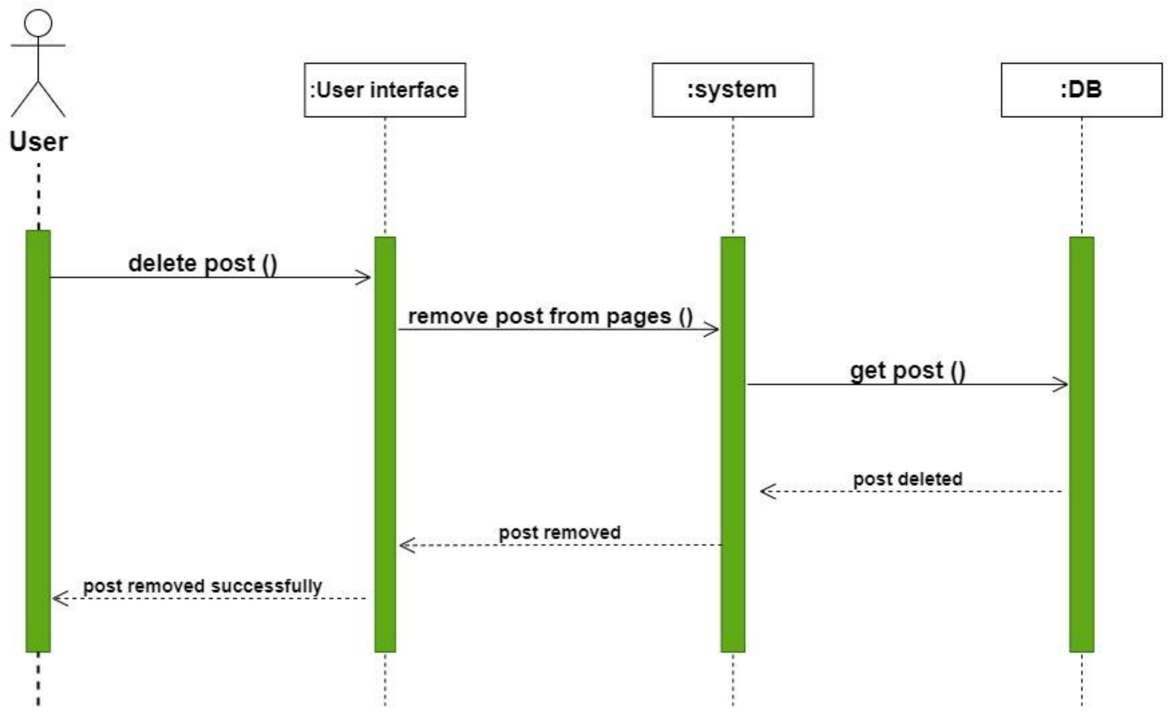
2- Login:



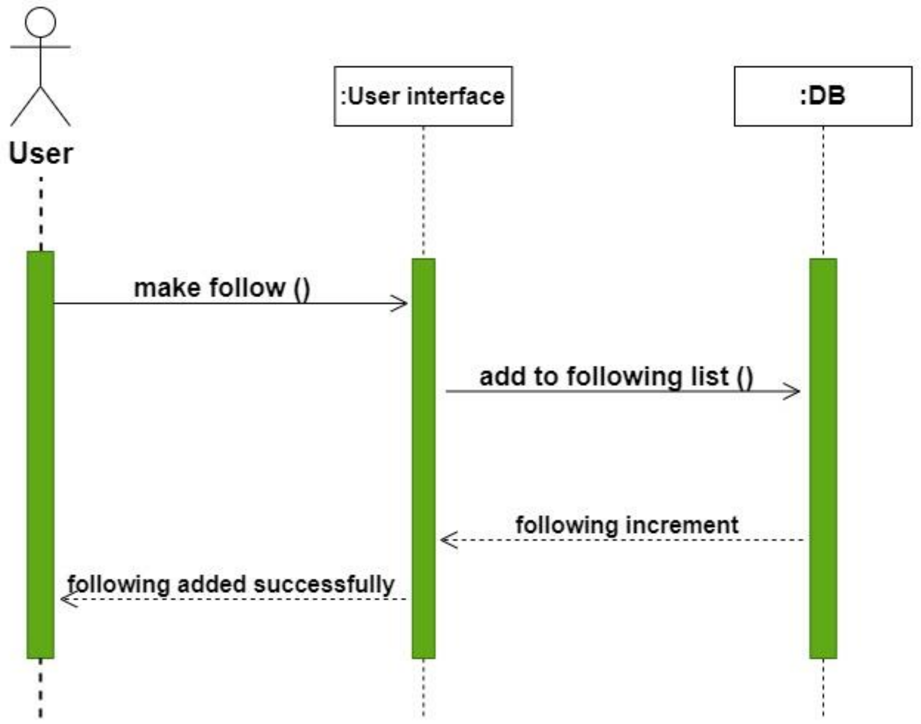
3- Add post:



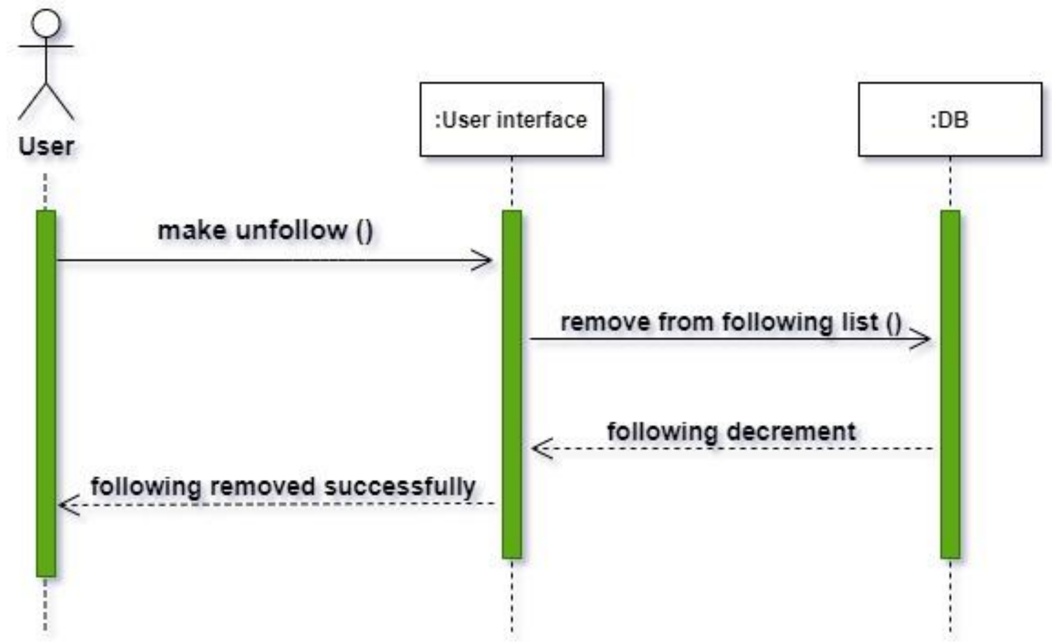
4- Delete post:



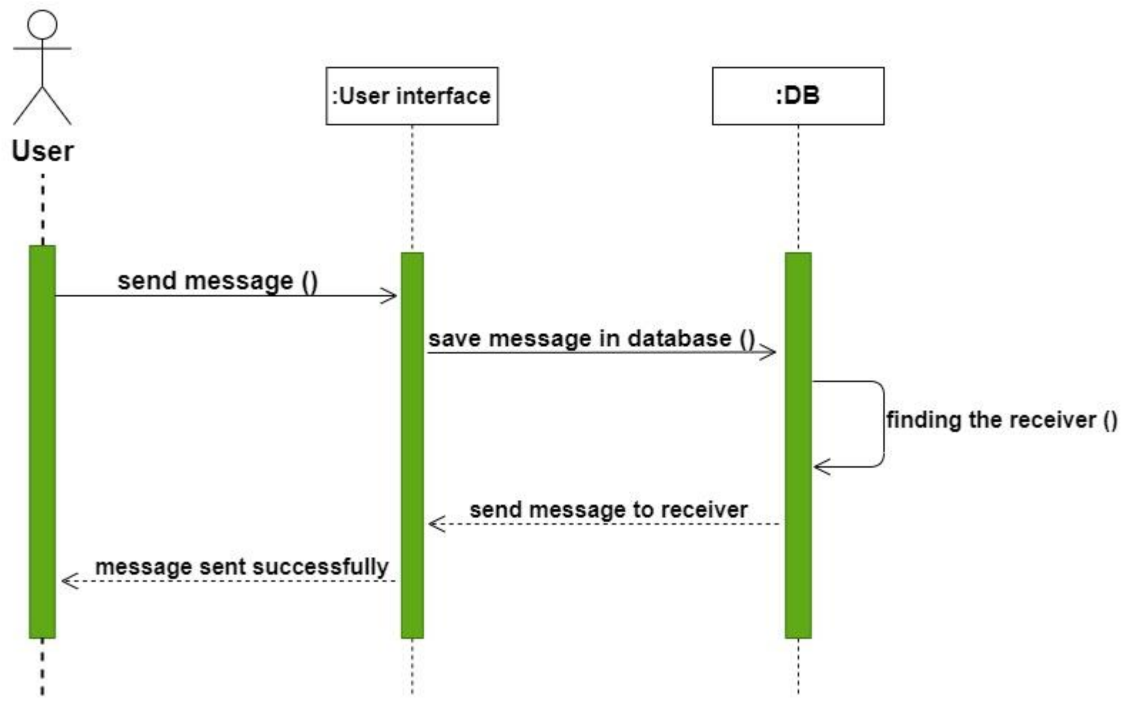
5- Follow:



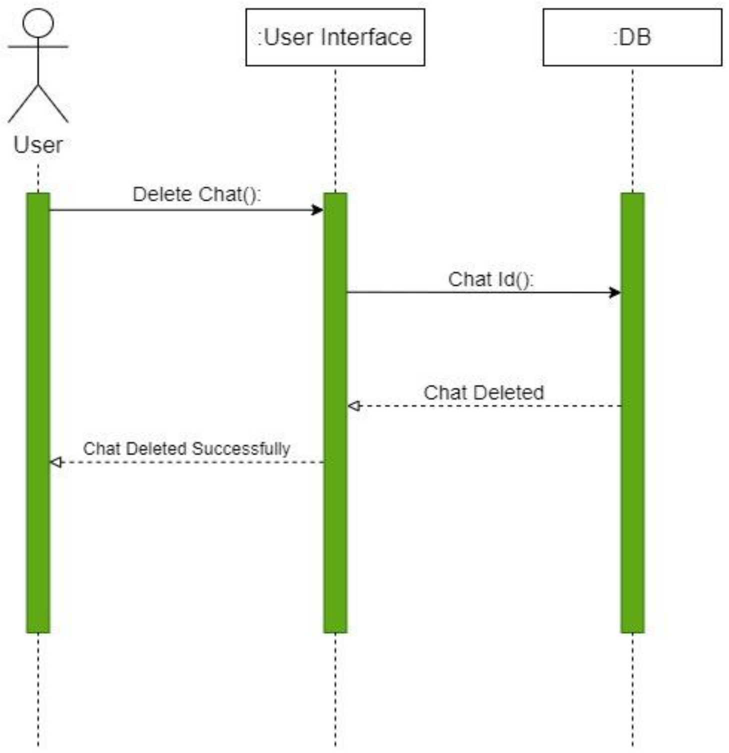
6- Unfollow:



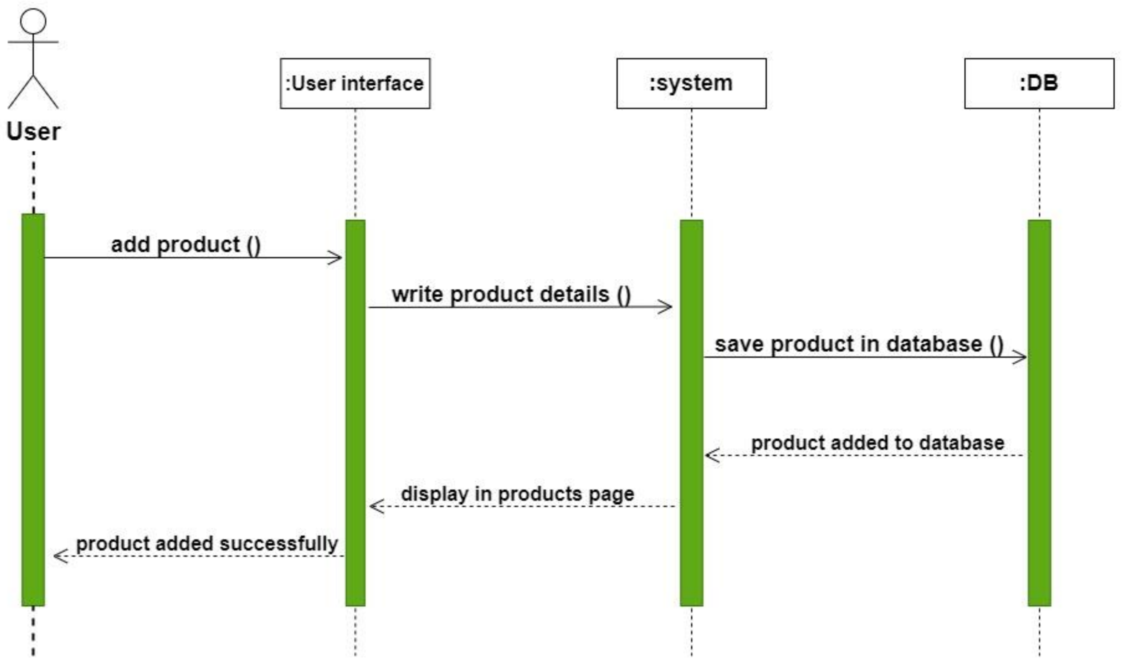
7- Send message:



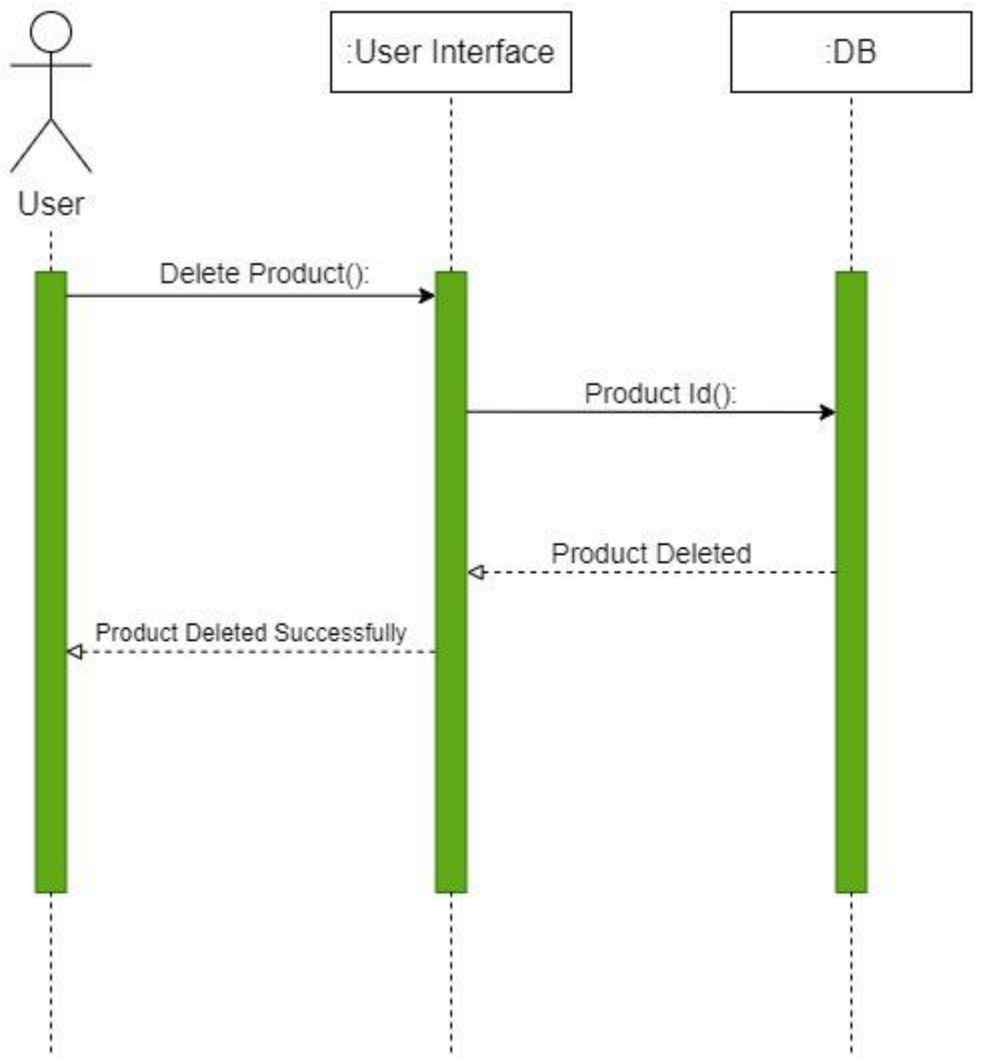
8- Delete chat:



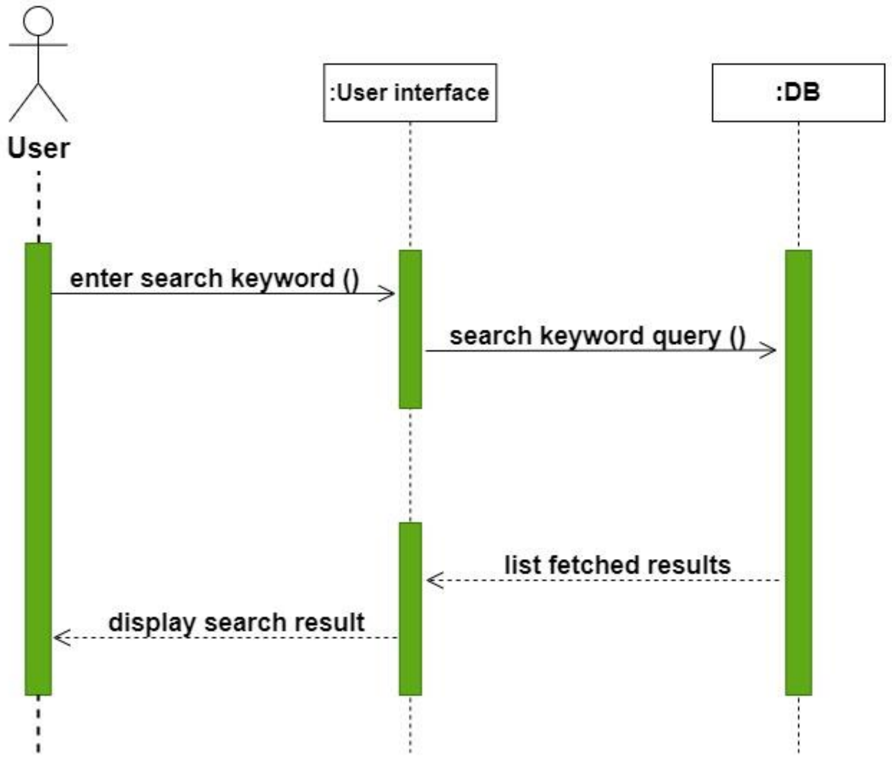
9- Add product:



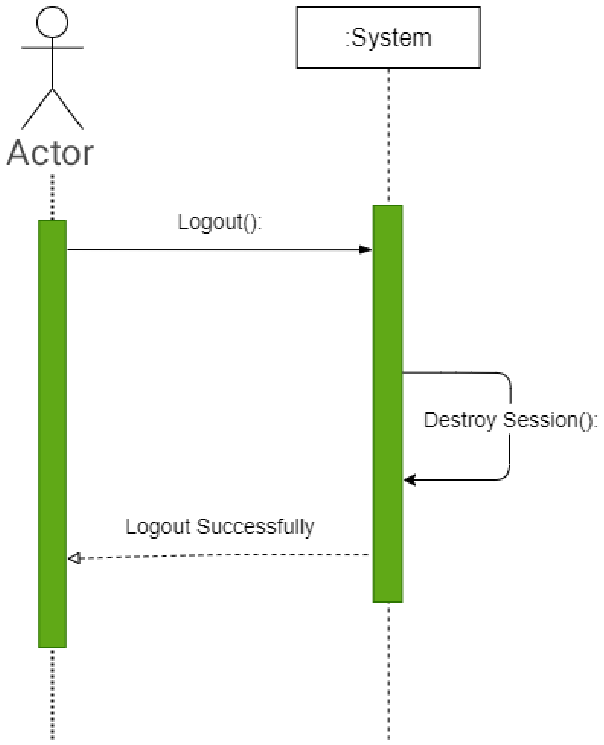
10- Delete product:



11- Search:

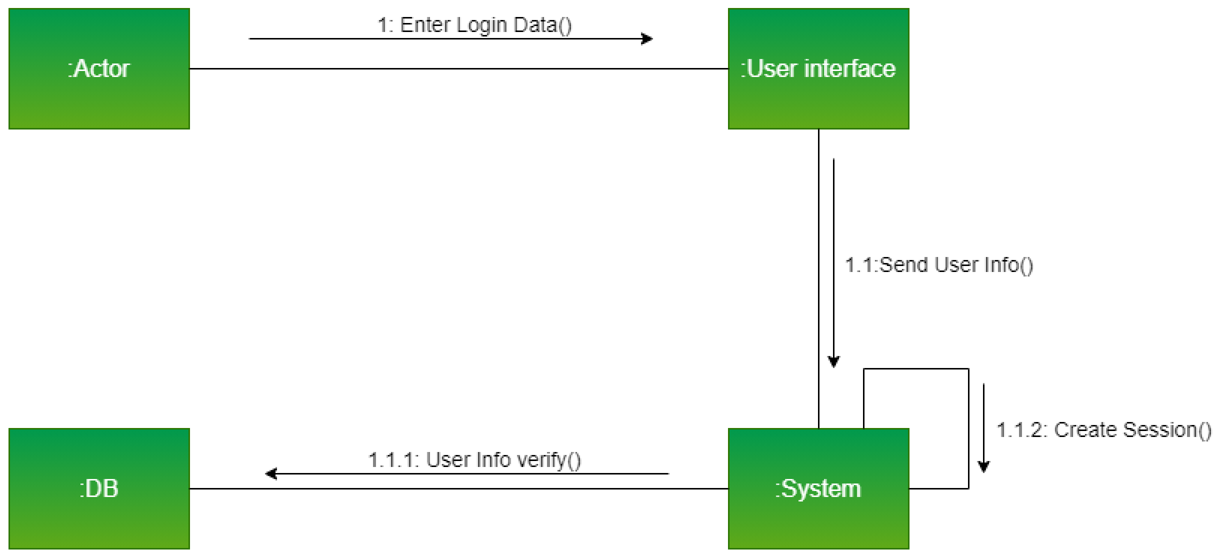


12- Logout:

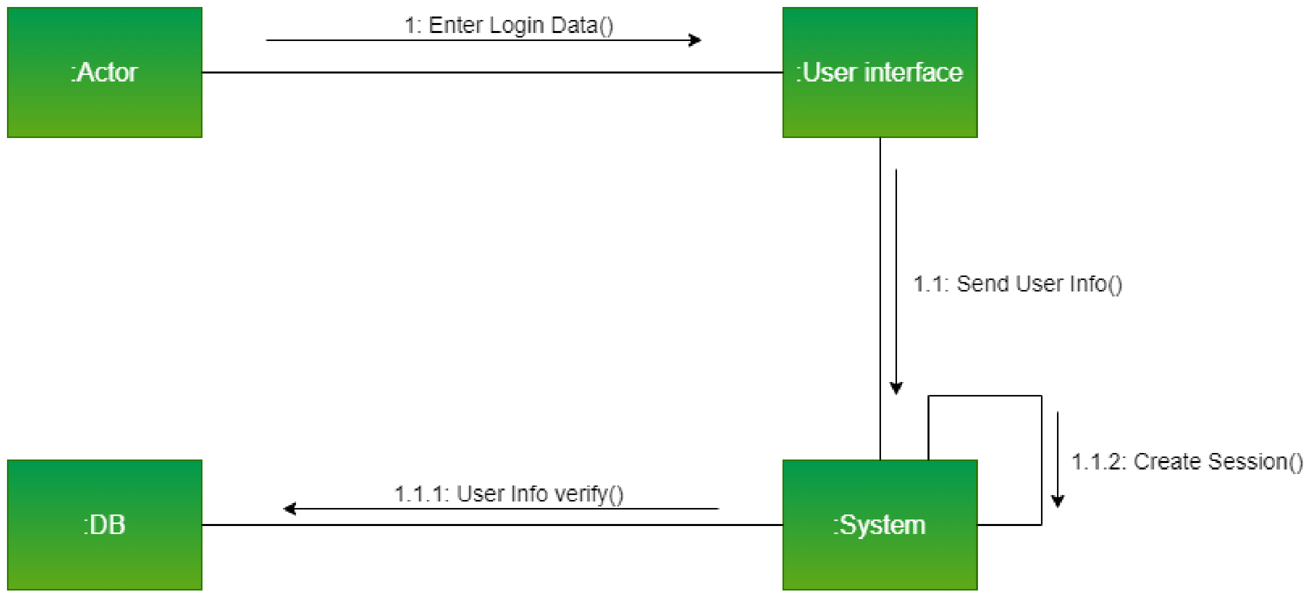


UML Collaboration:

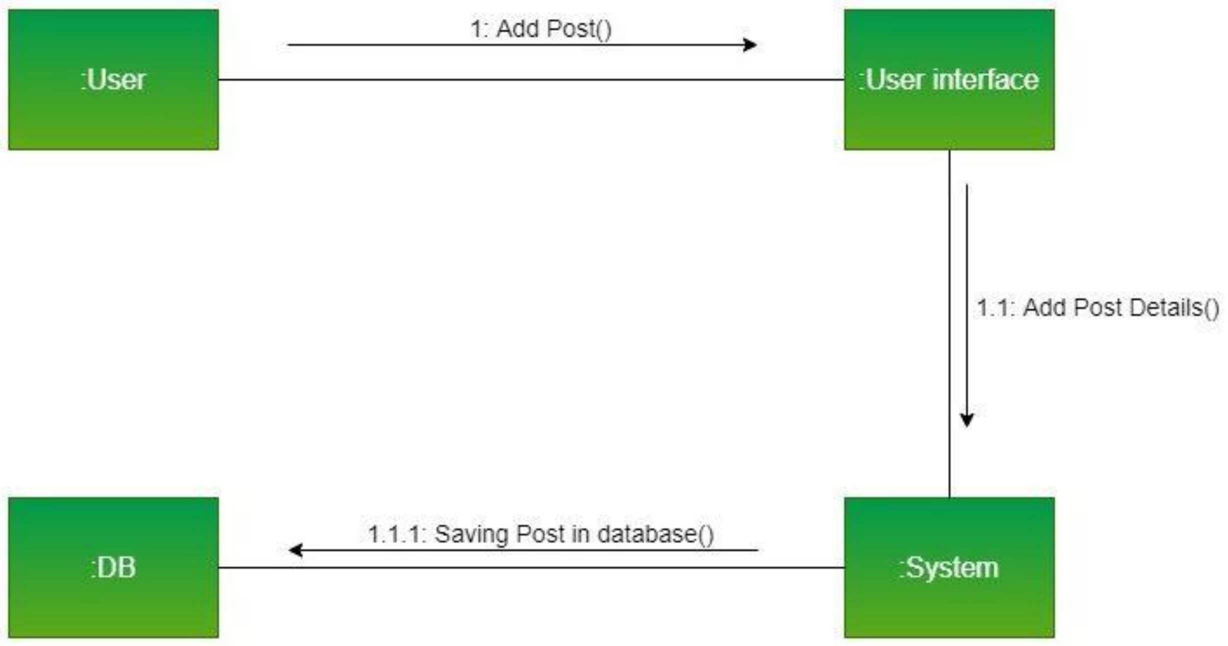
1- Register:



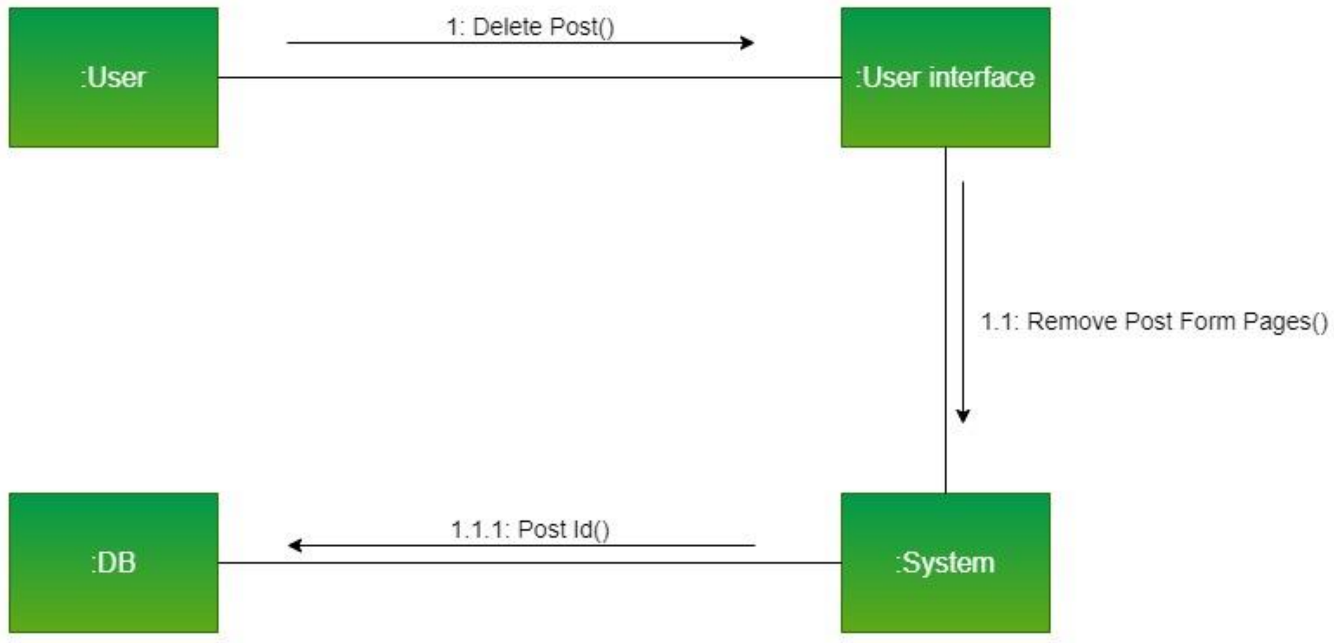
2- Login:



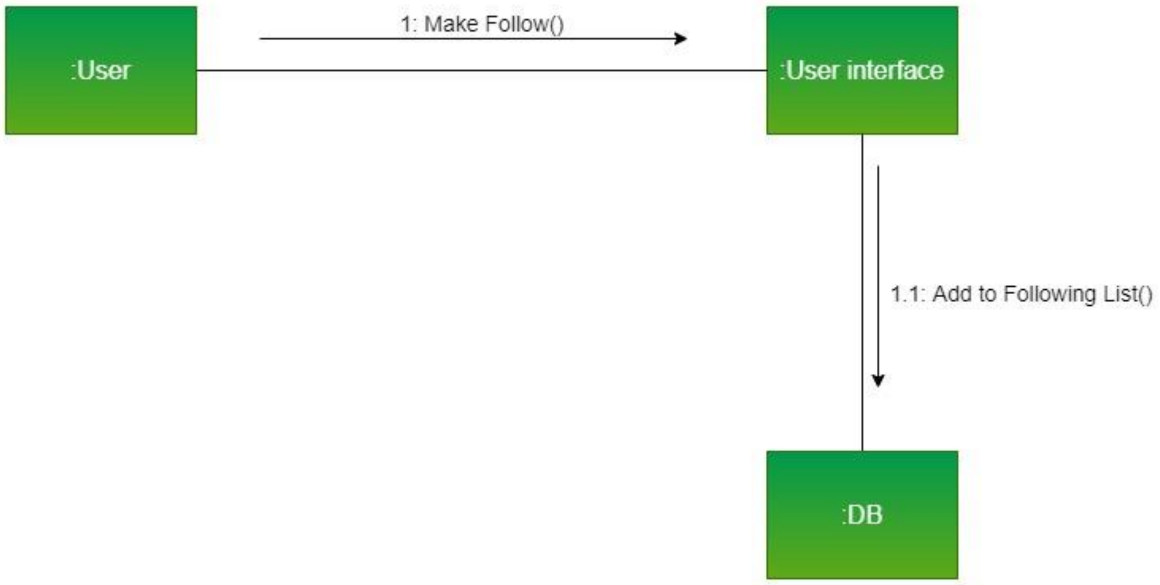
3- Add post:



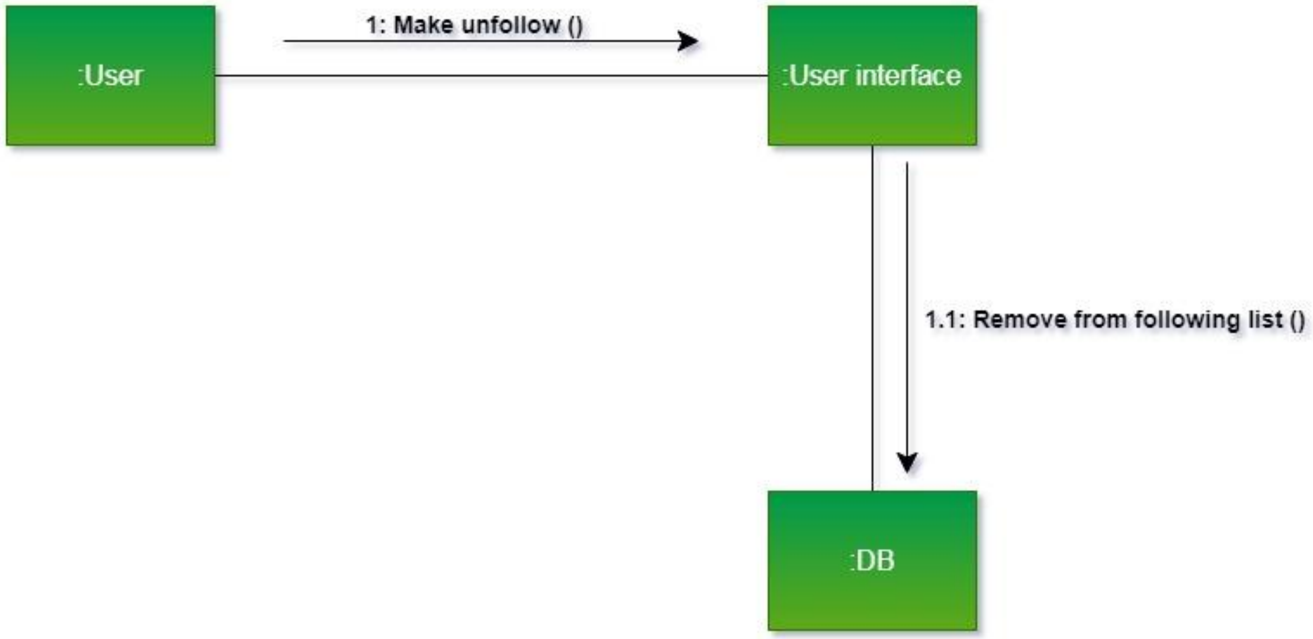
4- Delete post:



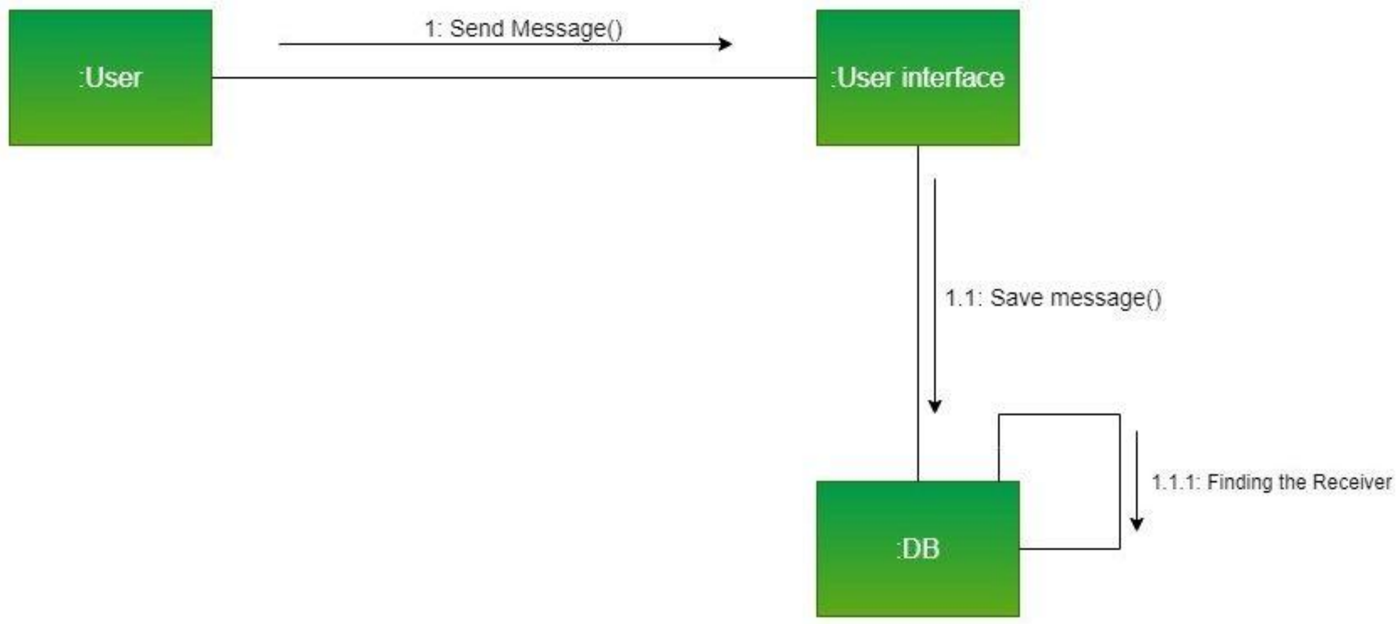
5- Follow:



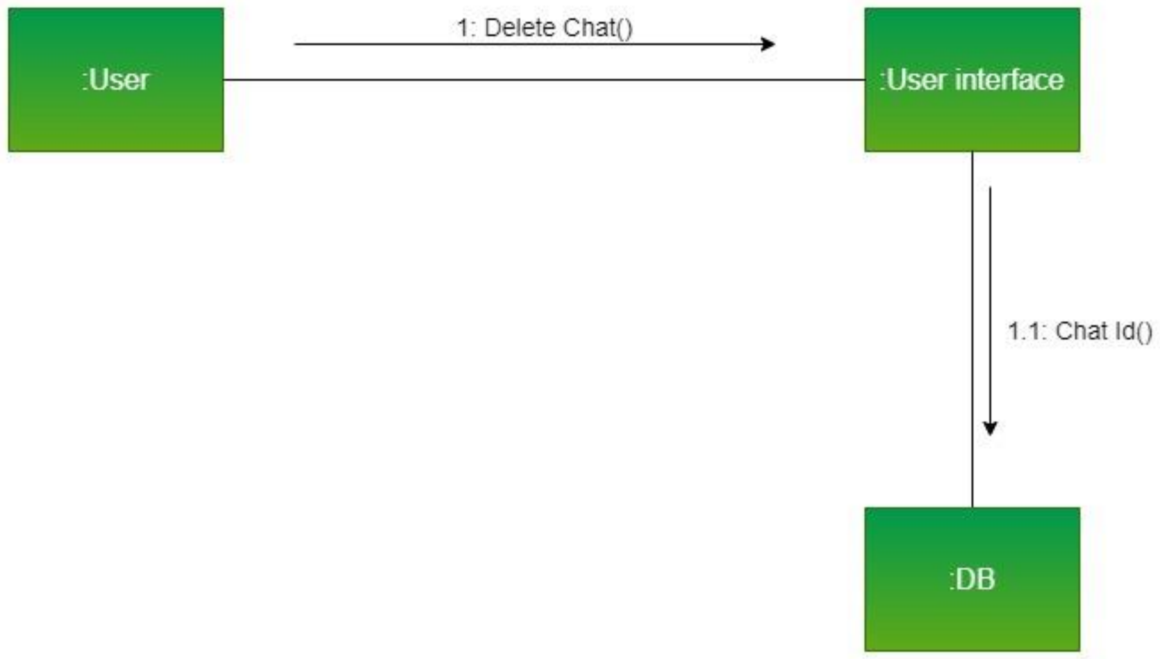
6- Unfollow:



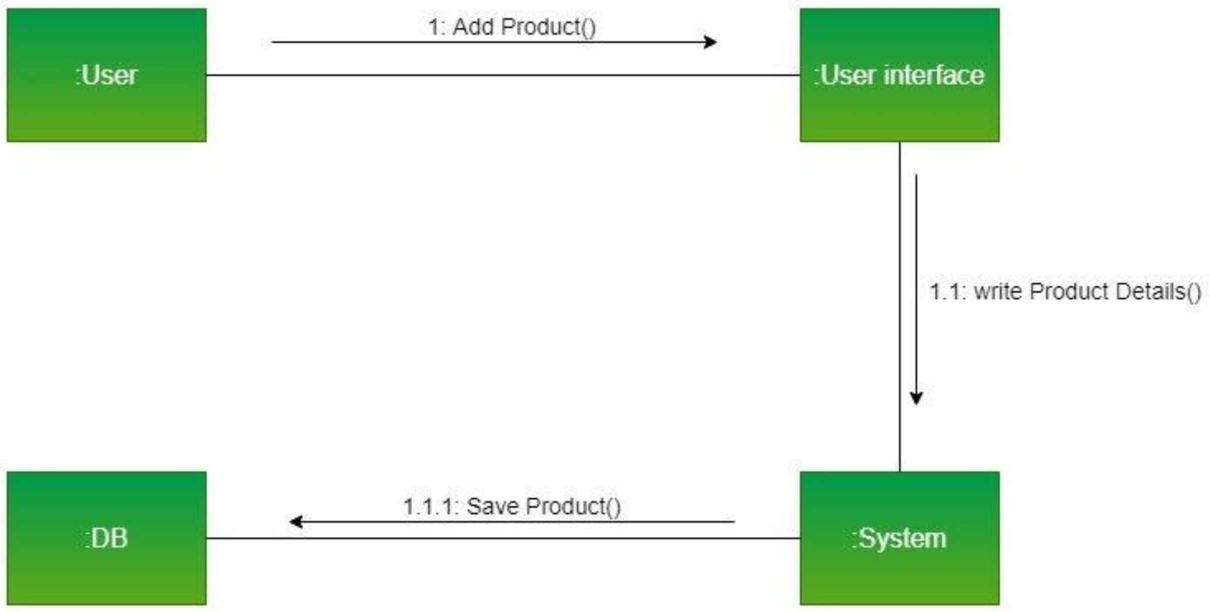
7- Send message:



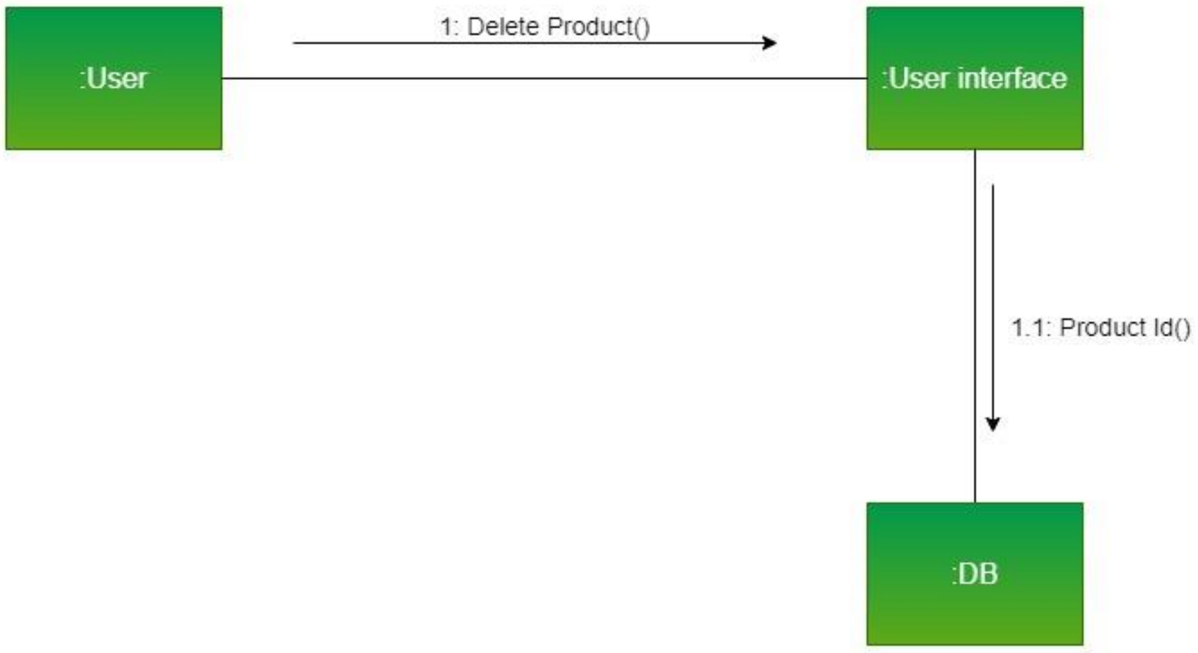
8- Delete chat:



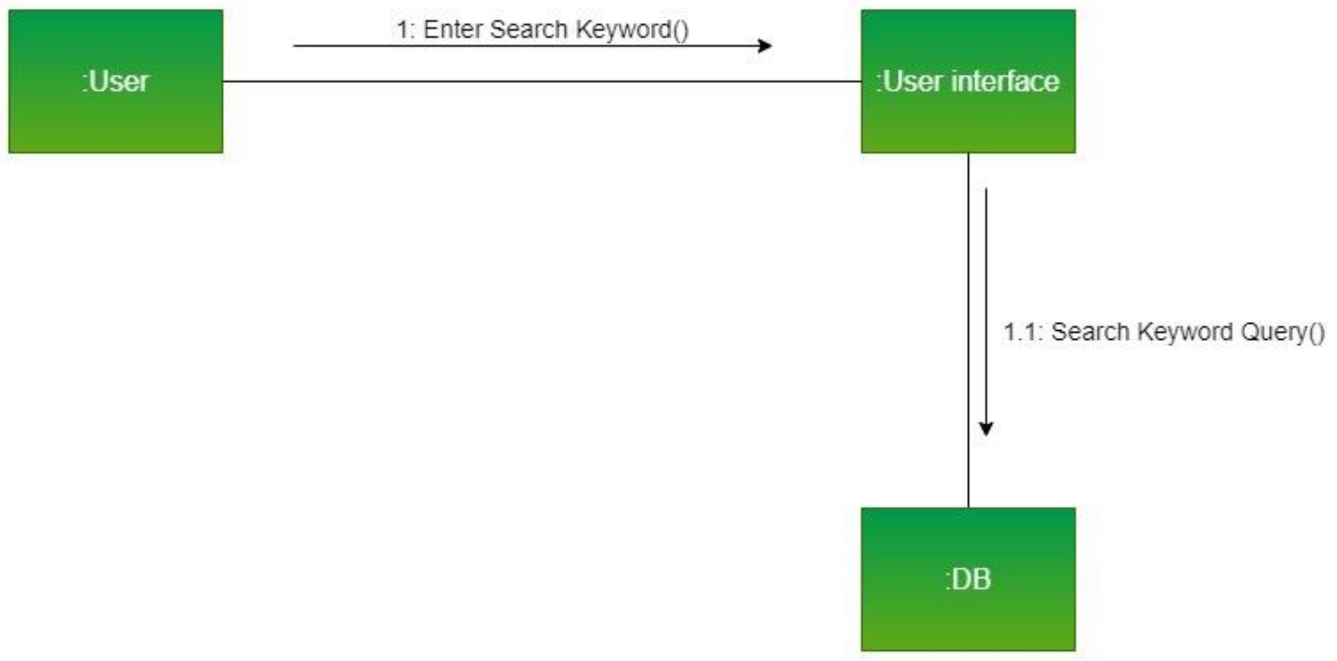
9- Add product:



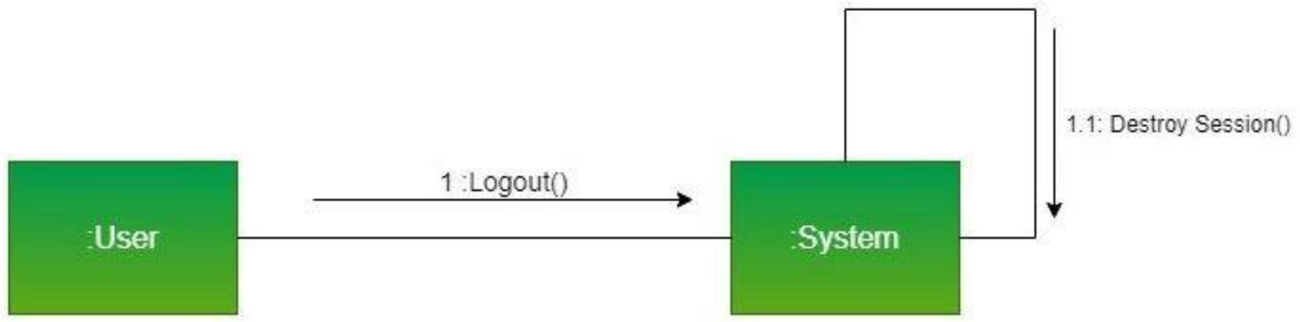
10- Delete product:



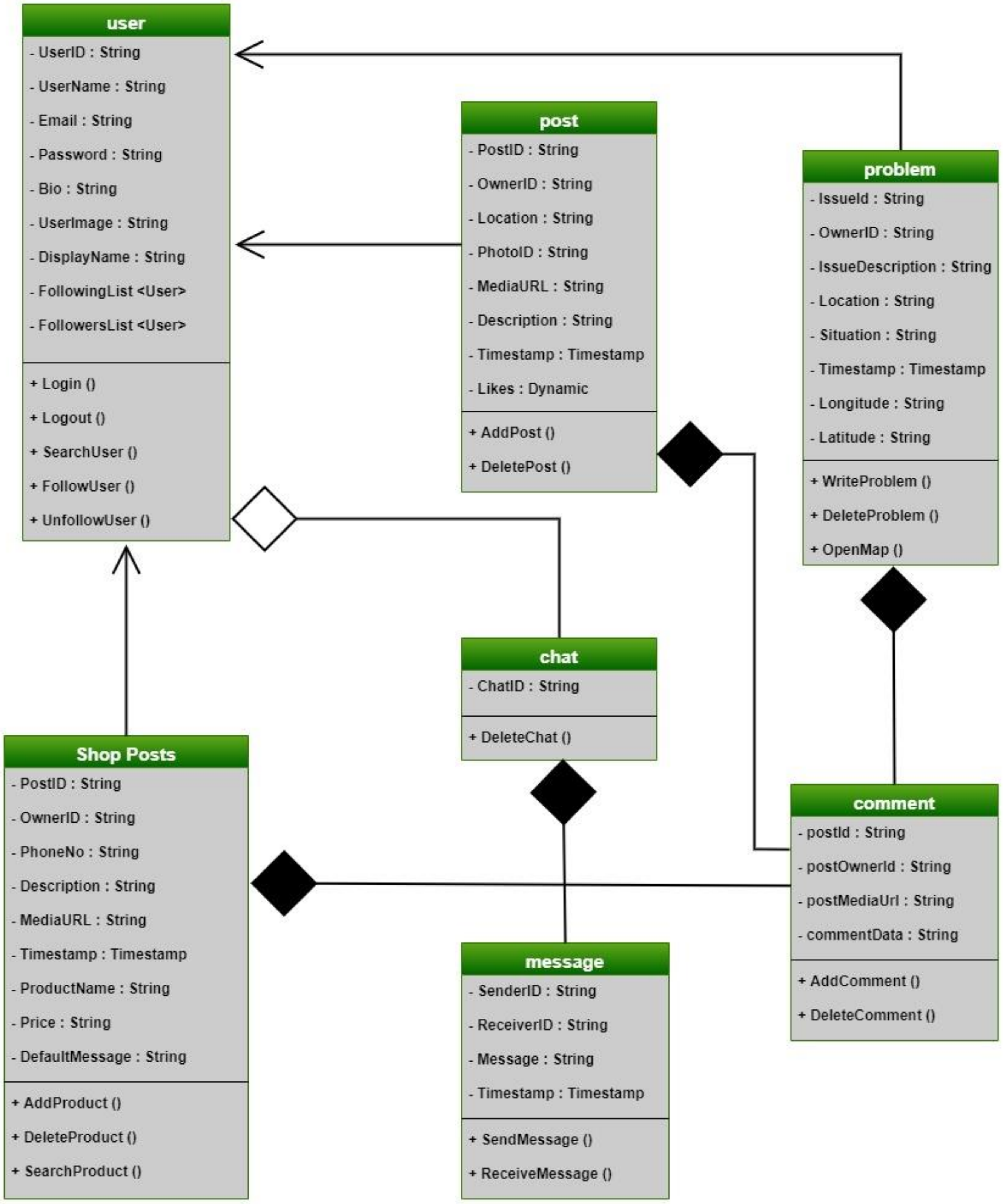
11- Search:



12- Logout:

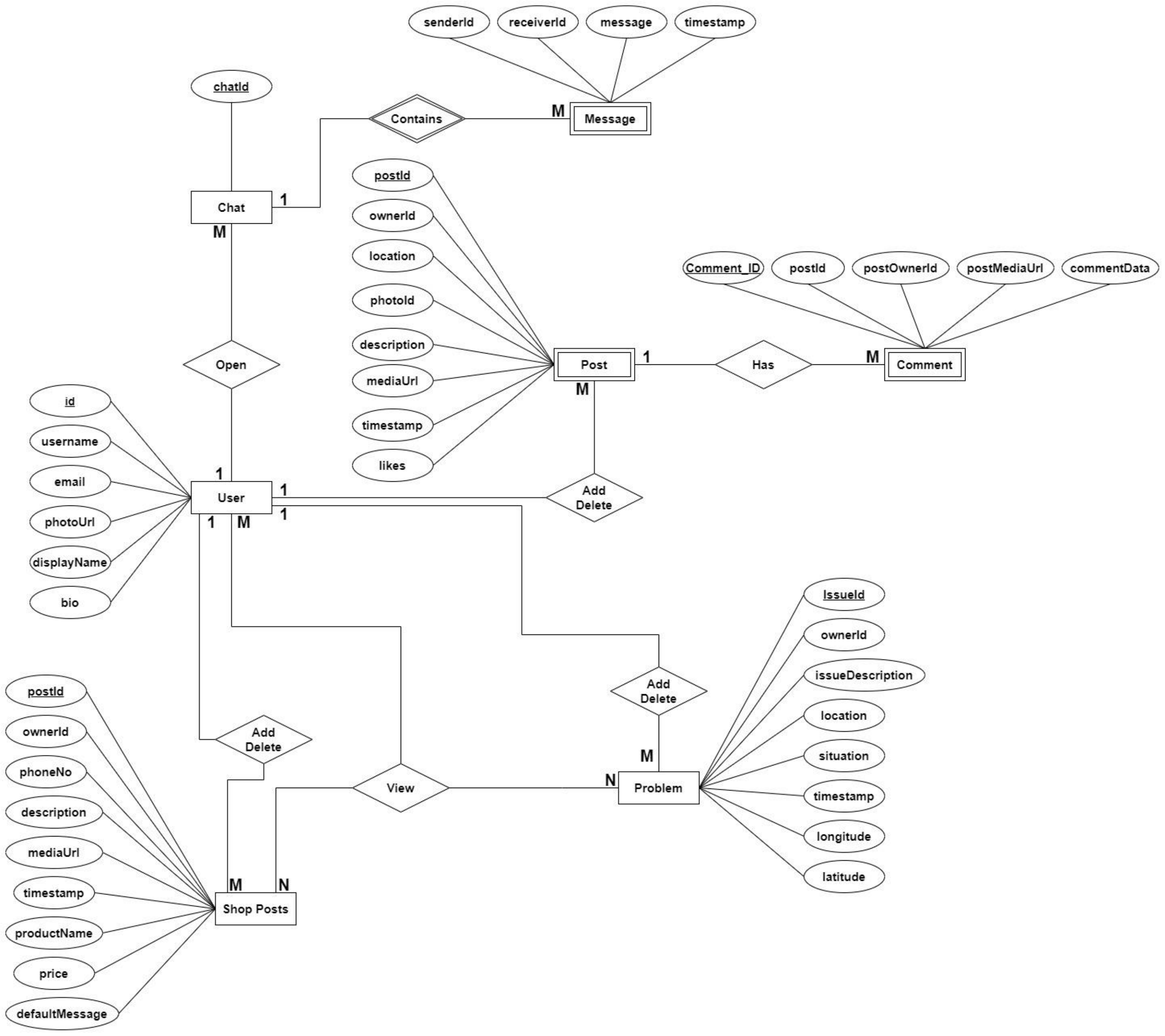


Class diagram:

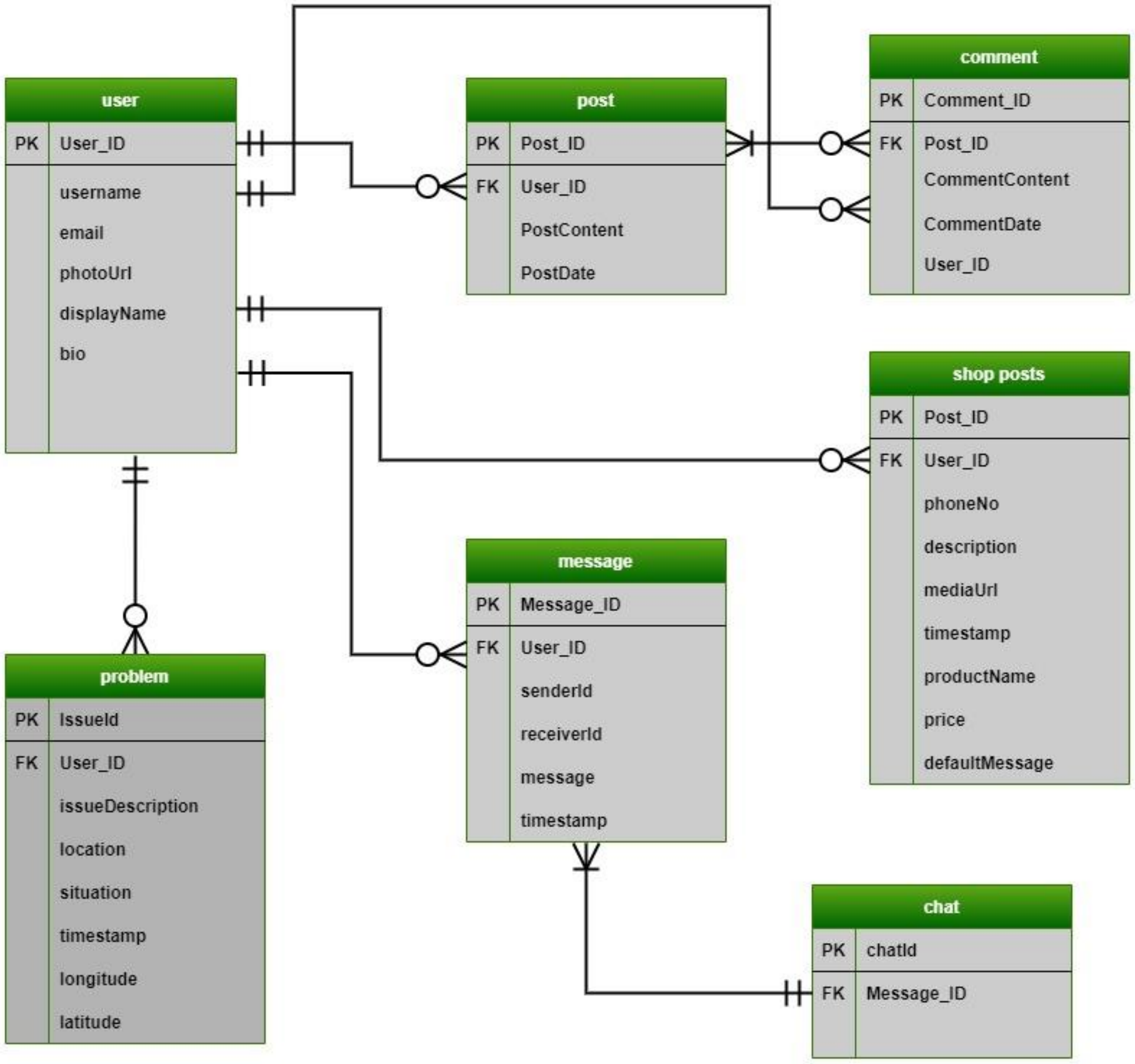


Database design:

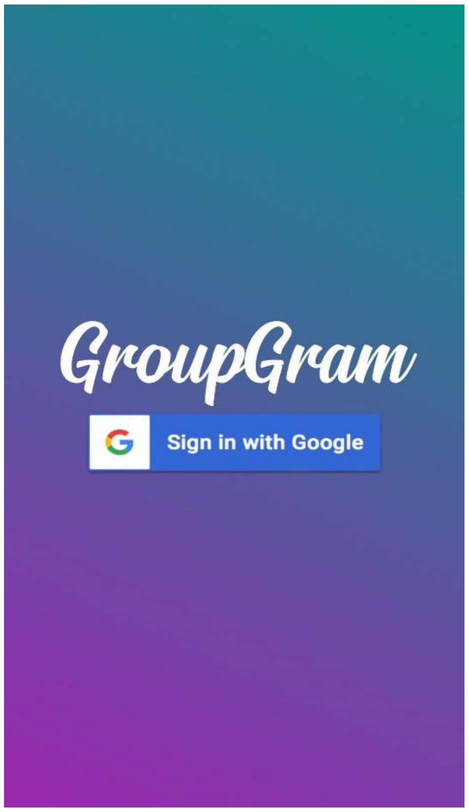
1- ERD:

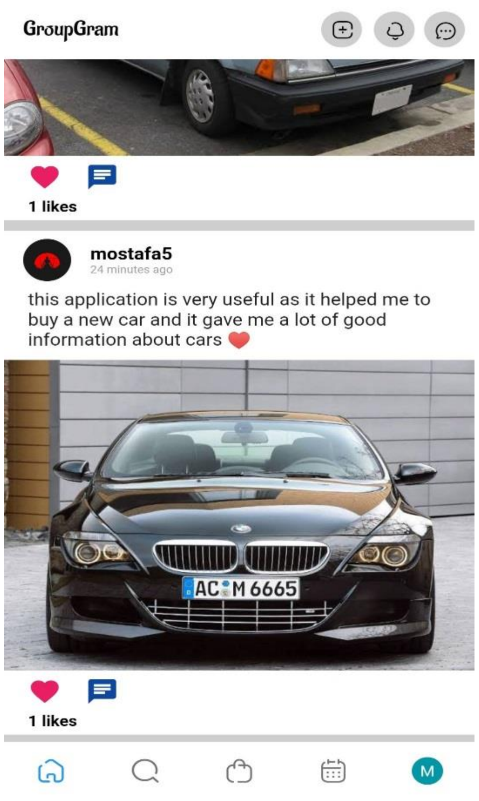


2- Database schema (Mapping):

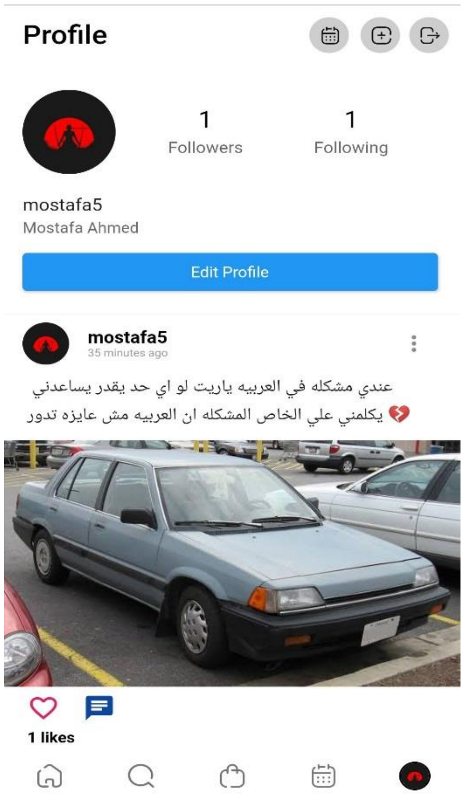


User interface design:

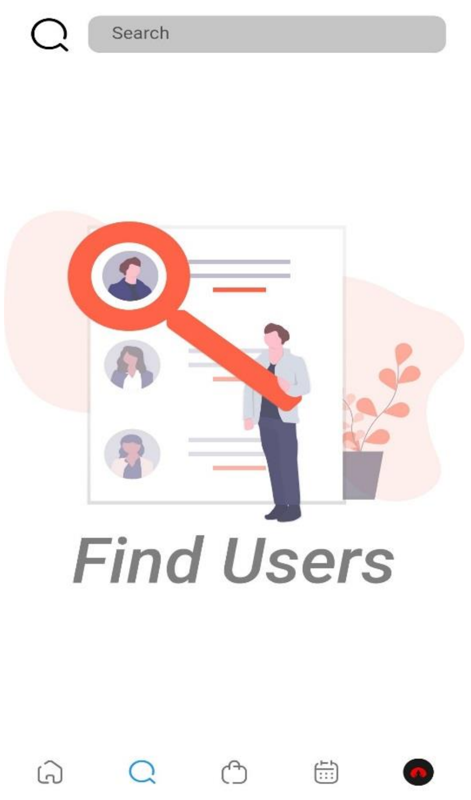


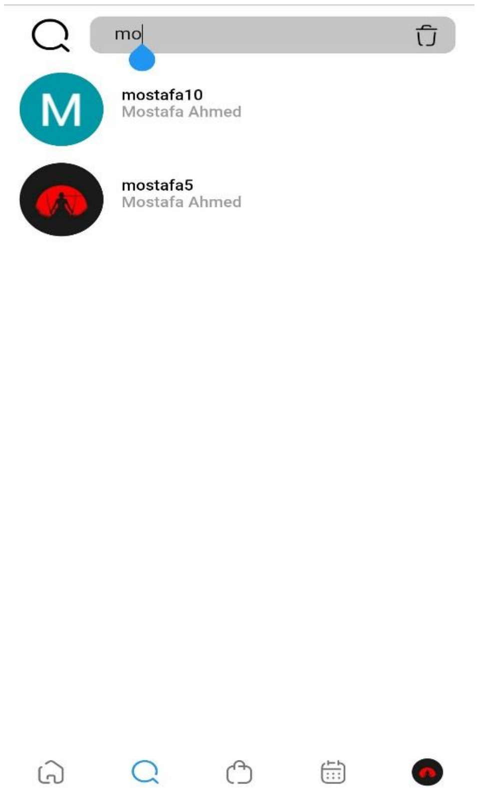


mostafa5
24 minutes ago
this application is very useful as it helped me to buy a new car and it gave me a lot of good information about cars ❤️



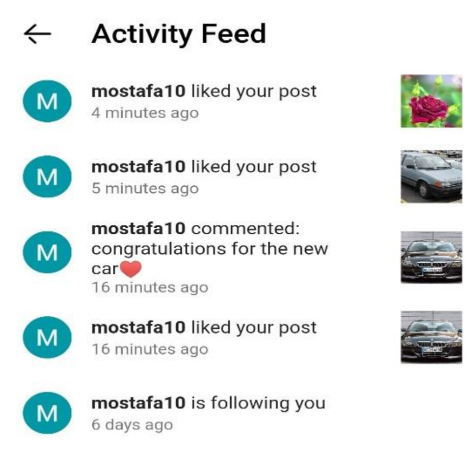
mostafa5
35 minutes ago
عندي مشكله في العربيه ياريت لو اي حد يقدر يساعدني يكلمني علي الخاص المشكله ان العربيه مش عايزه تدور 💔





mostafa10
Mostafa Ahmed

mostafa5
Mostafa Ahmed



Activity Feed

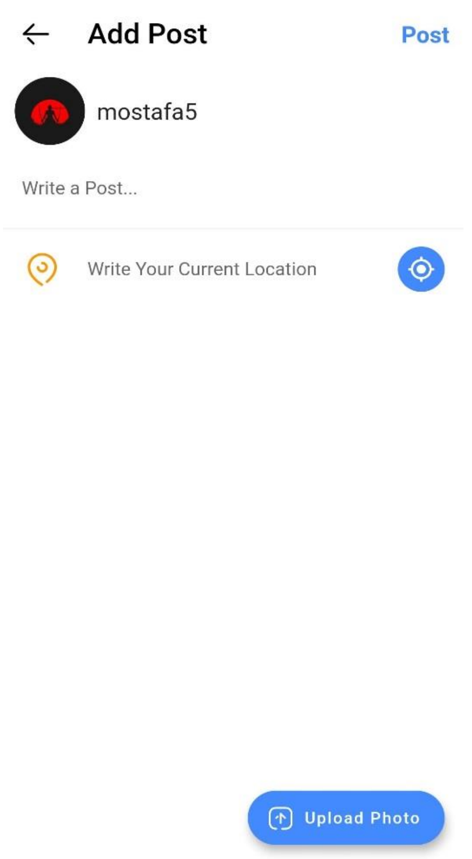
mostafa10 liked your post
4 minutes ago

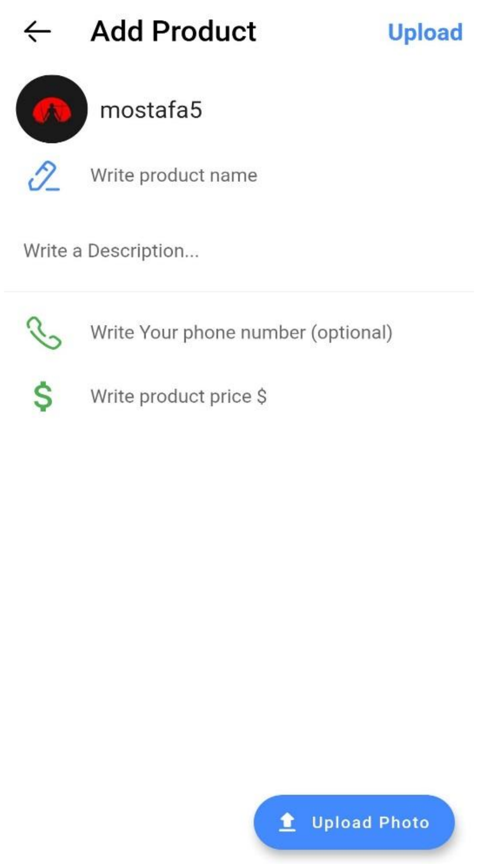
mostafa10 liked your post
5 minutes ago

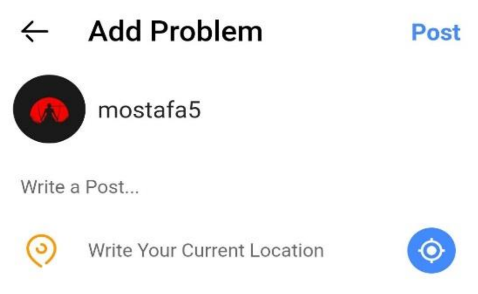
mostafa10 commented: congratulations for the new car❤️
16 minutes ago

mostafa10 liked your post
16 minutes ago

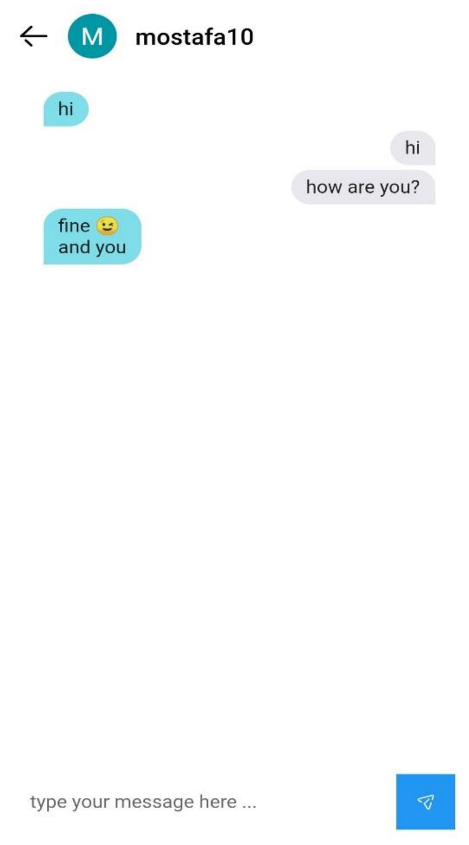
mostafa10 is following you
6 days ago







Chat
mostafa10
Start Chat!

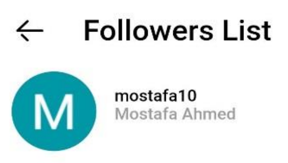


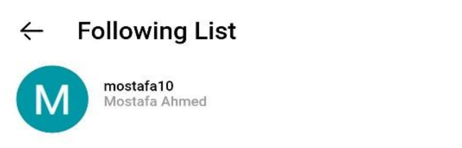
hi

hi

how are you?

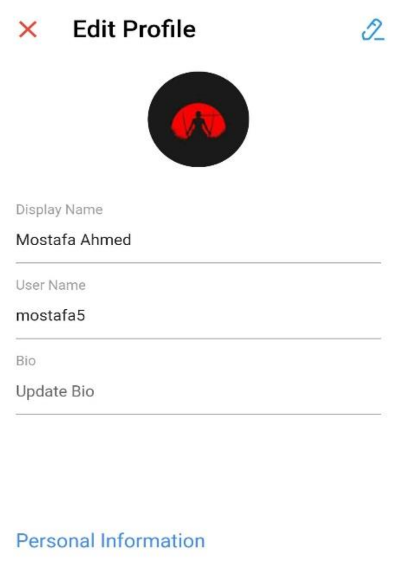
fine 😉
and you





Following List

mostafa10
Mostafa Ahmed



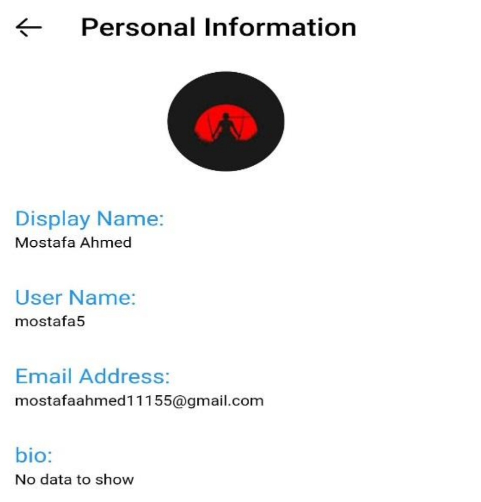
Edit Profile

Display Name
Mostafa Ahmed

User Name
mostafa5

Bio
Update Bio

Personal Information



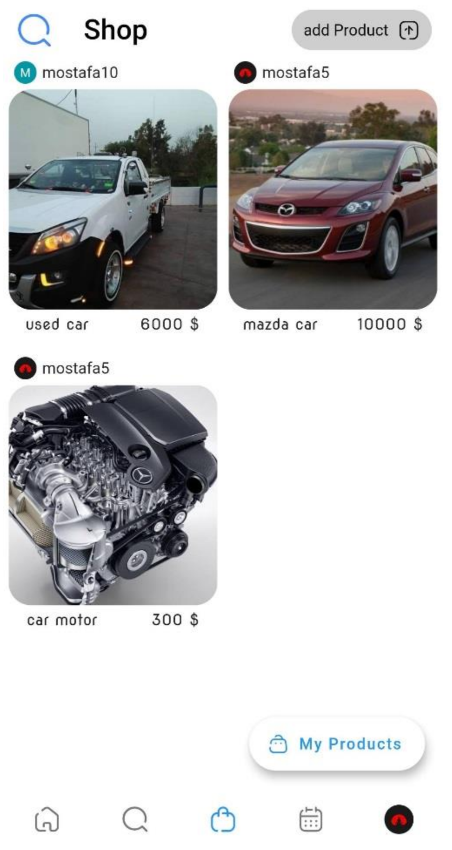
Personal Information

Display Name:
Mostafa Ahmed

User Name:
mostafa5

Email Address:
mostafaahmed11155@gmail.com

bio:
No data to show

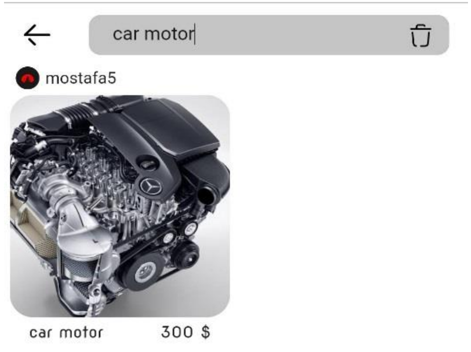


Shop

add Product

mostafa10	mostafa5
used car 6000 $	mazda car 10000 $
mostafa5	
car motor 300 $	

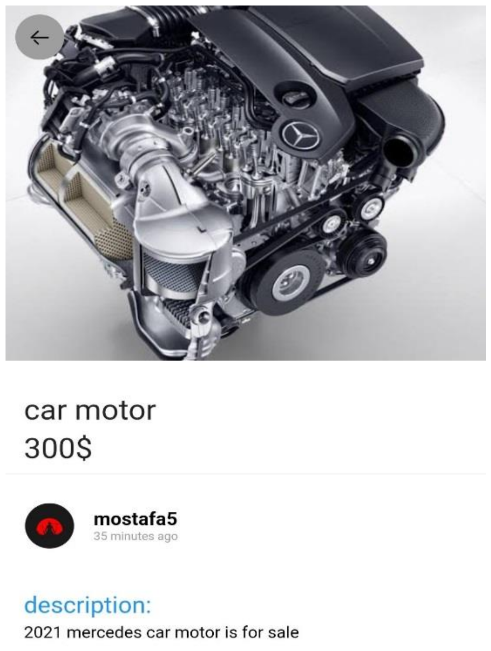
My Products



car motor

mostafa5

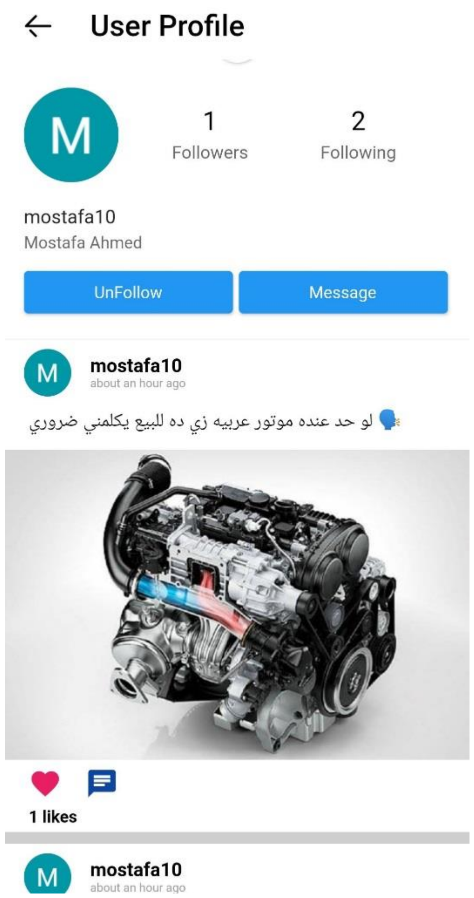
car motor 300 $



car motor
300$

mostafa5
35 minutes ago

description:
2021 mercedes car motor is for sale



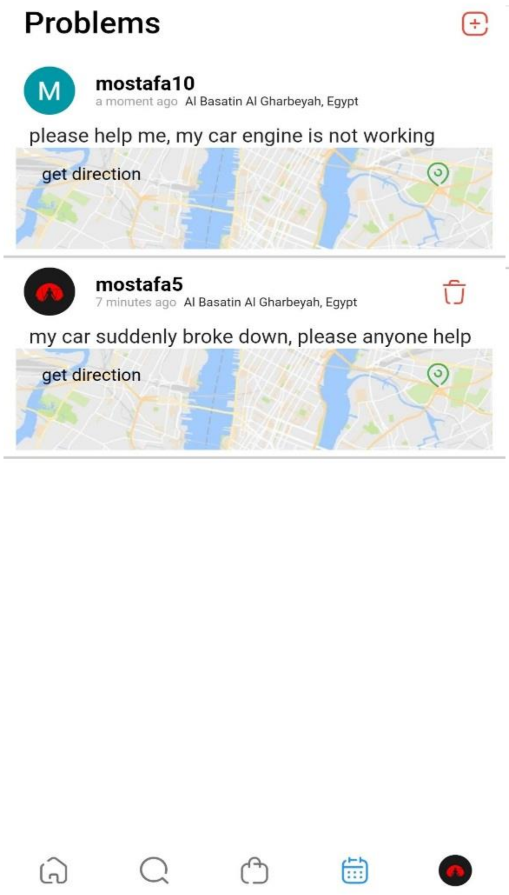
User Profile

mostafa10
Mostafa Ahmed

mostafa10
about an hour ago
لو حد عنده موتور عربيه زي ده للبيع يكلمني ضروري

1 likes

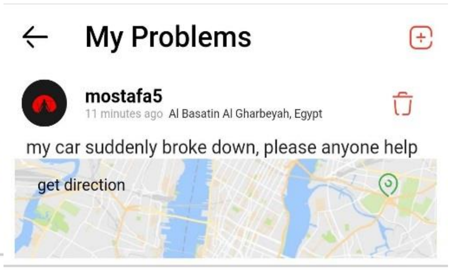
mostafa10
about an hour ago



Problems

mostafa10
a moment ago Al Basatin Al Gharbeyah, Egypt
please help me, my car engine is not working

mostafa5
7 minutes ago Al Basatin Al Gharbeyah, Egypt
my car suddenly broke down, please anyone help



My Problems

mostafa5
11 minutes ago Al Basatin Al Gharbeyah, Egypt
my car suddenly broke down, please anyone help

5. Implementation

In this chapter, we will discuss application's implementation, and present its code, the UI, its flowchart and the algorithms and tools used to build it

Description of implementation:

This application is concerned with a very important issue which is making the process of communication between people easier by creating an environment where all people can make posts about anything, comment on posts, like posts, add product for sale in the shop, make a problem post, and edit profile

We used many packages to divide the cores of every package of code such as package for user dashboard, package for profile dashboard, package for shop dashboard, and package for home dashboard. We used the provider package to manage the state of some functions in the application, and we also created collections in firebase such as feed, posts, issues, comments, users, following, followers, and shop posts. And for every collection we made the crud operations from the real firebase firestore package in pubdev

We managed the state of fetch using stream builder for randomly changing stuff and future builder for only refreshable stuff like posts, also we have forms of login and profile edit and management for every user as long as he logged into his account which is already verified using google sign in

For getting the hardware solutions we had to bring a very strong PC to bear the software IDE that we used which was the android studio, so we need 16 GB RAM and 3.5 GHZ Processor and large memory space, and for running the application we needed a virtual space for the android emulator and a real device to see how the application is running on a real device

We also used built-in pubdev libraries and packages for helping us in the process of implementation

For doing all these processes and developing the application we studied flutter and dart for developing the UI, and firebase for making the database, so we studied a course named (Flutter and dart the complete guide)

Programming language and technology:

- We used flutter framework technology to build the structure of the application and to implement the code

- Firebase fire store and firebase storage as a remote database

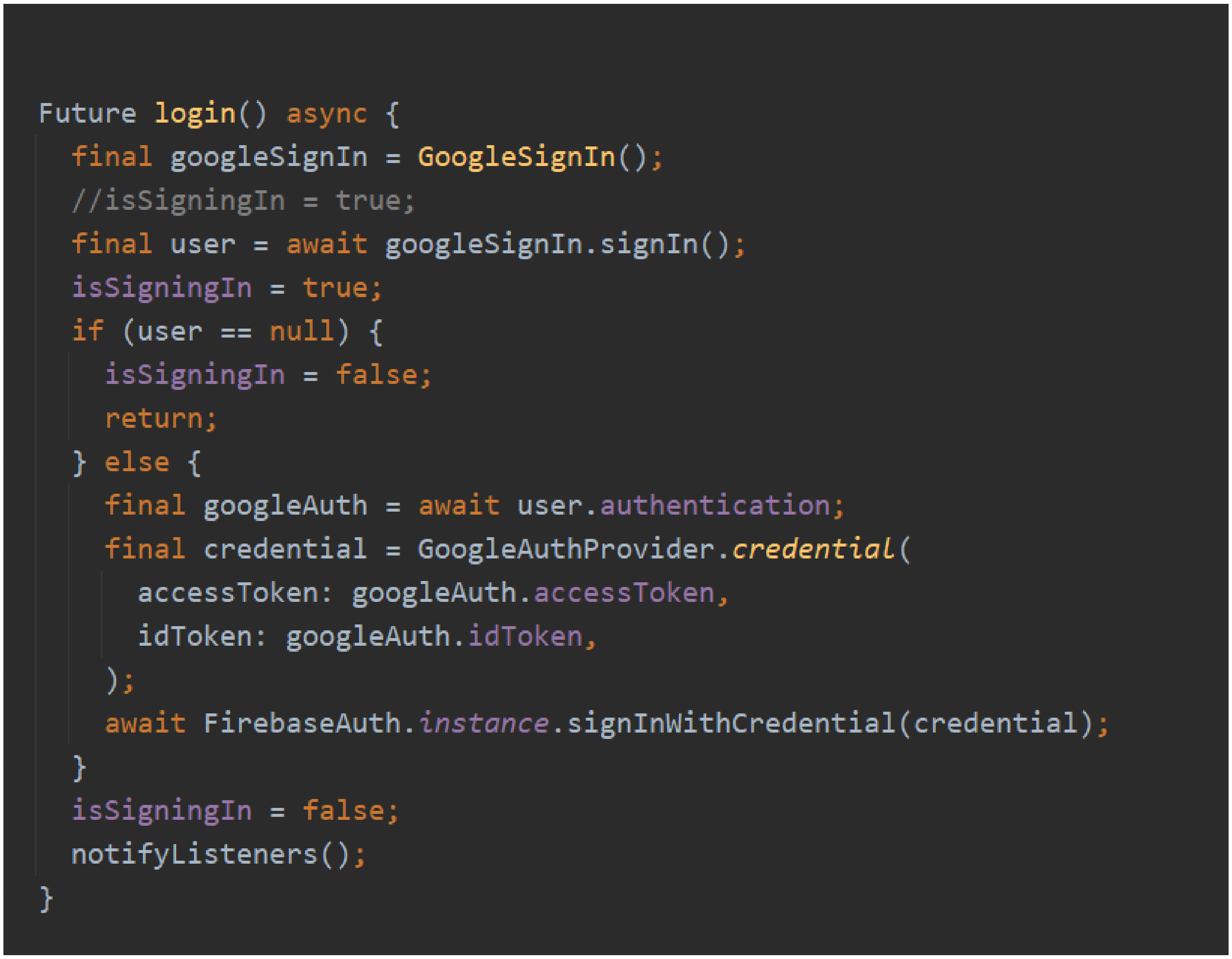
- Google map's application and the geolocator package to detect the user location and open it for navigation in google maps

- We used firebase authentication and google sign in to authenticate users

- We used providers for the state management of our application

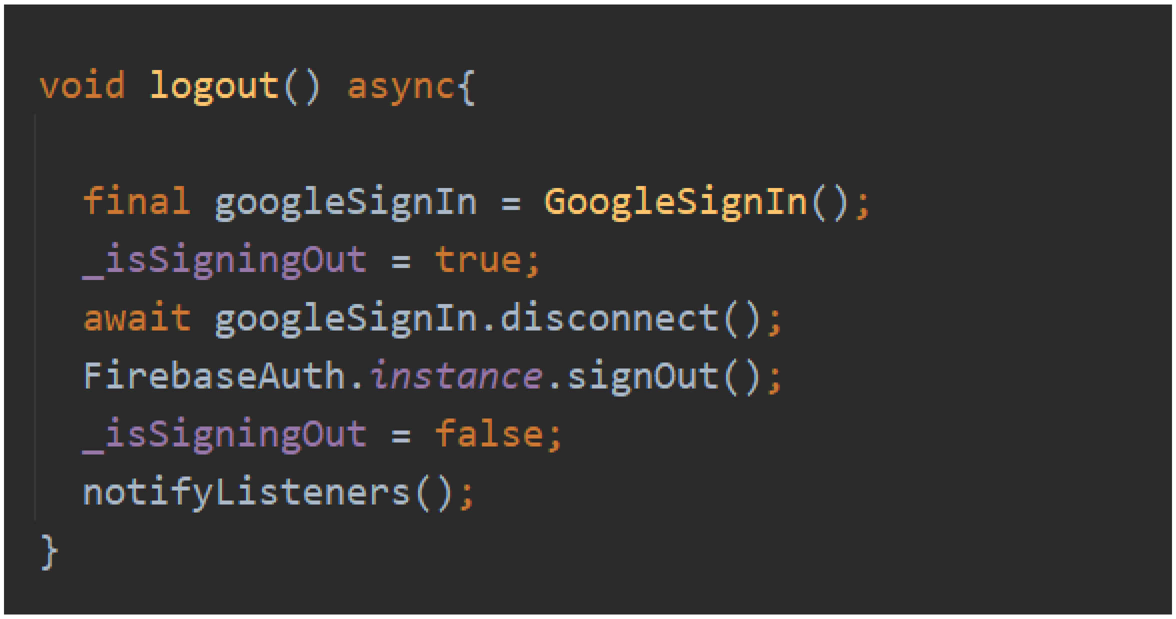
Part of implementation:

Login:



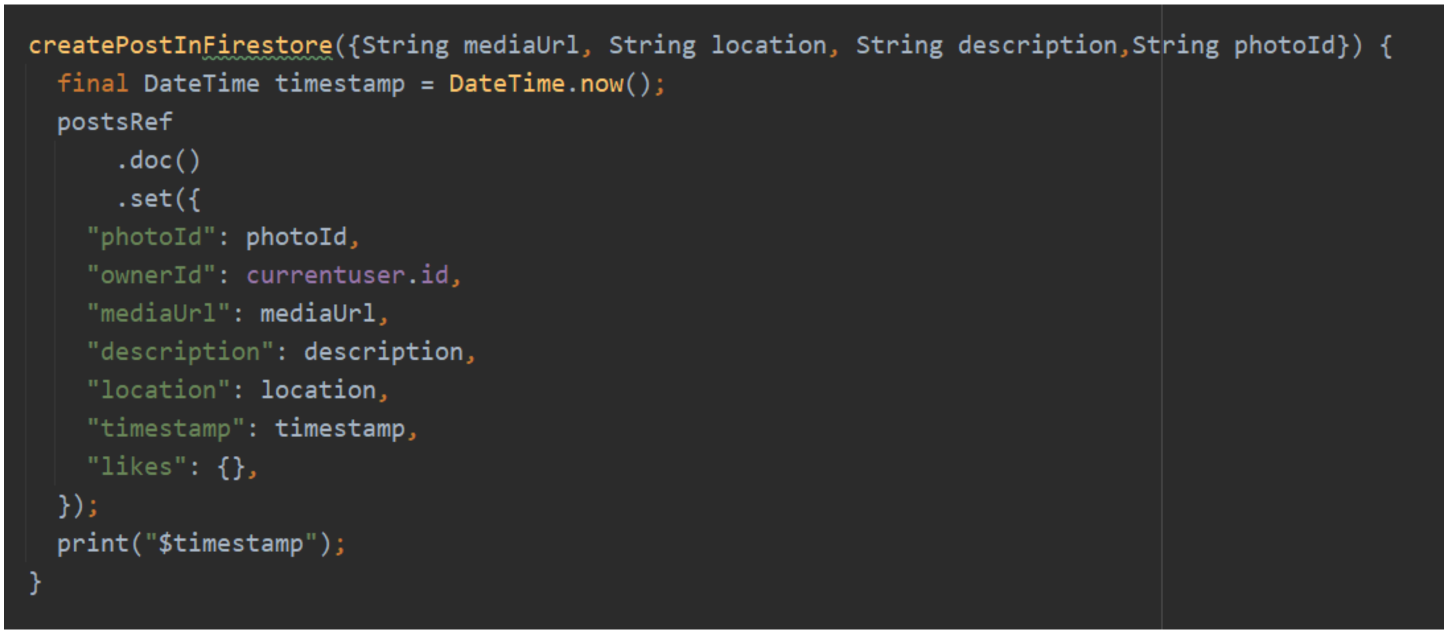
```
Future login() async {
  final googleSignIn = GoogleSignIn();
  //isSigningIn = true;
  final user = await googleSignIn.signIn();
  isSigningIn = true;
  if (user == null) {
    isSigningIn = false;
    return;
  } else {
    final googleAuth = await user.authentication;
    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );
    await FirebaseAuth.instance.signInWithCredential(credential);
  }
  isSigningIn = false;
  notifyListeners();
}
```

Logout:



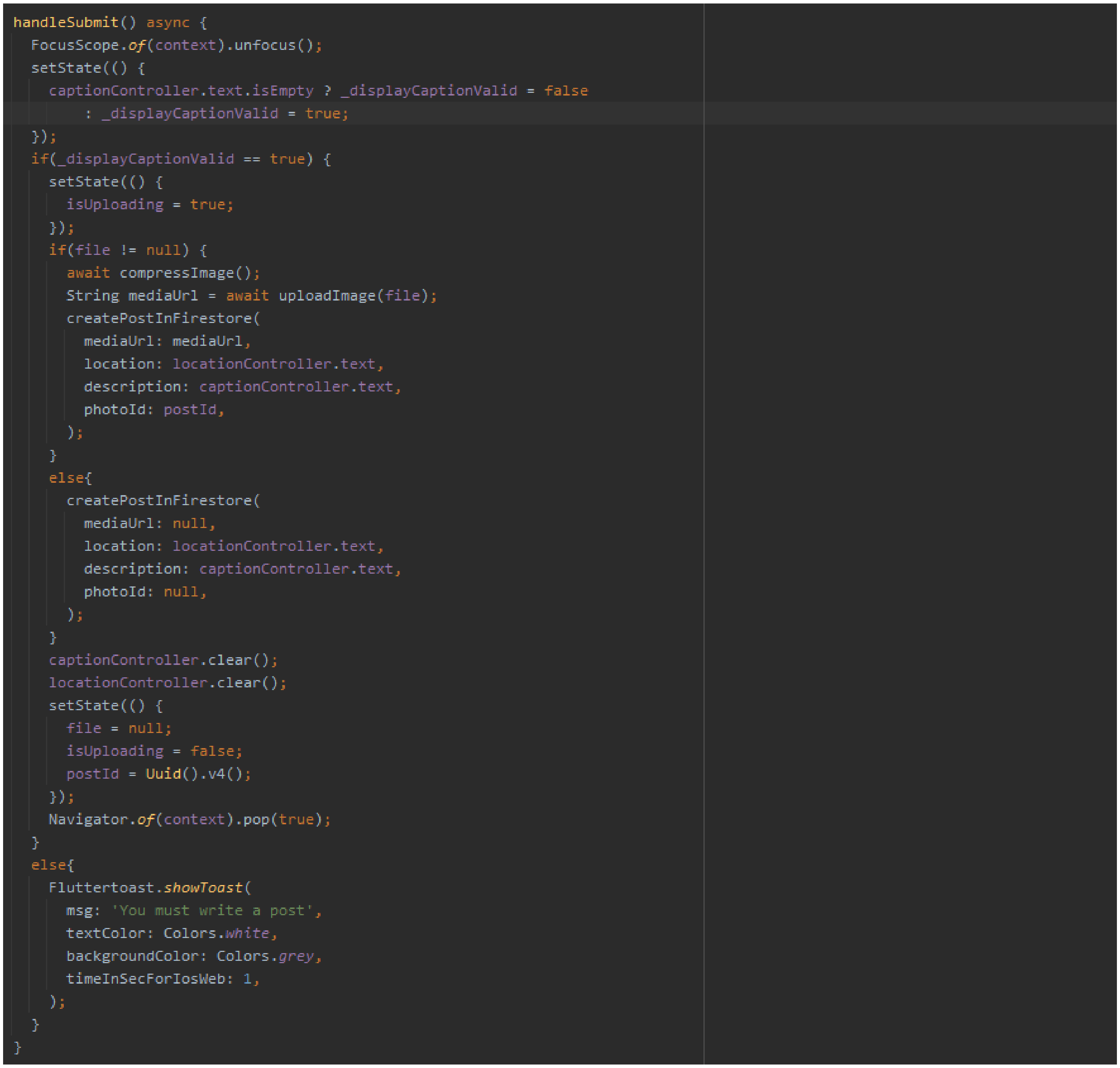
```
void logout() async{

  final googleSignIn = GoogleSignIn();
  _isSigningOut = true;
  await googleSignIn.disconnect();
  FirebaseAuth.instance.signOut();
  _isSigningOut = false;
  notifyListeners();
}
```

Add post:



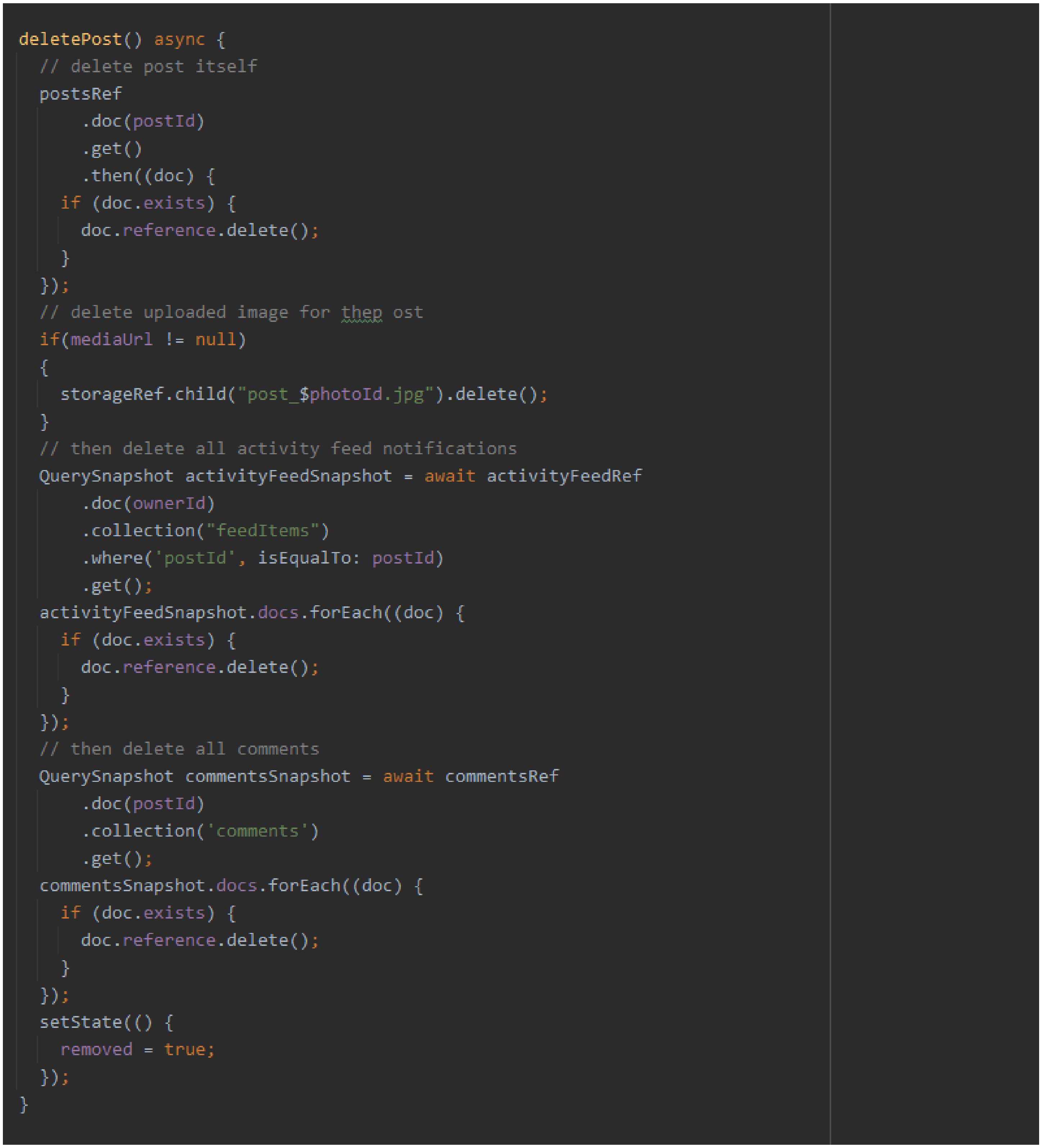
```
createPostInFirestore({String mediaUrl, String location, String description,String photoId}) {
  final DateTime timestamp = DateTime.now();
  postsRef
      .doc()
      .set({
    "photoId": photoId,
    "ownerId": currentuser.id,
    "mediaUrl": mediaUrl,
    "description": description,
    "location": location,
    "timestamp": timestamp,
    "likes": {},
  });
  print("$timestamp");
}
```

Handle post submit:



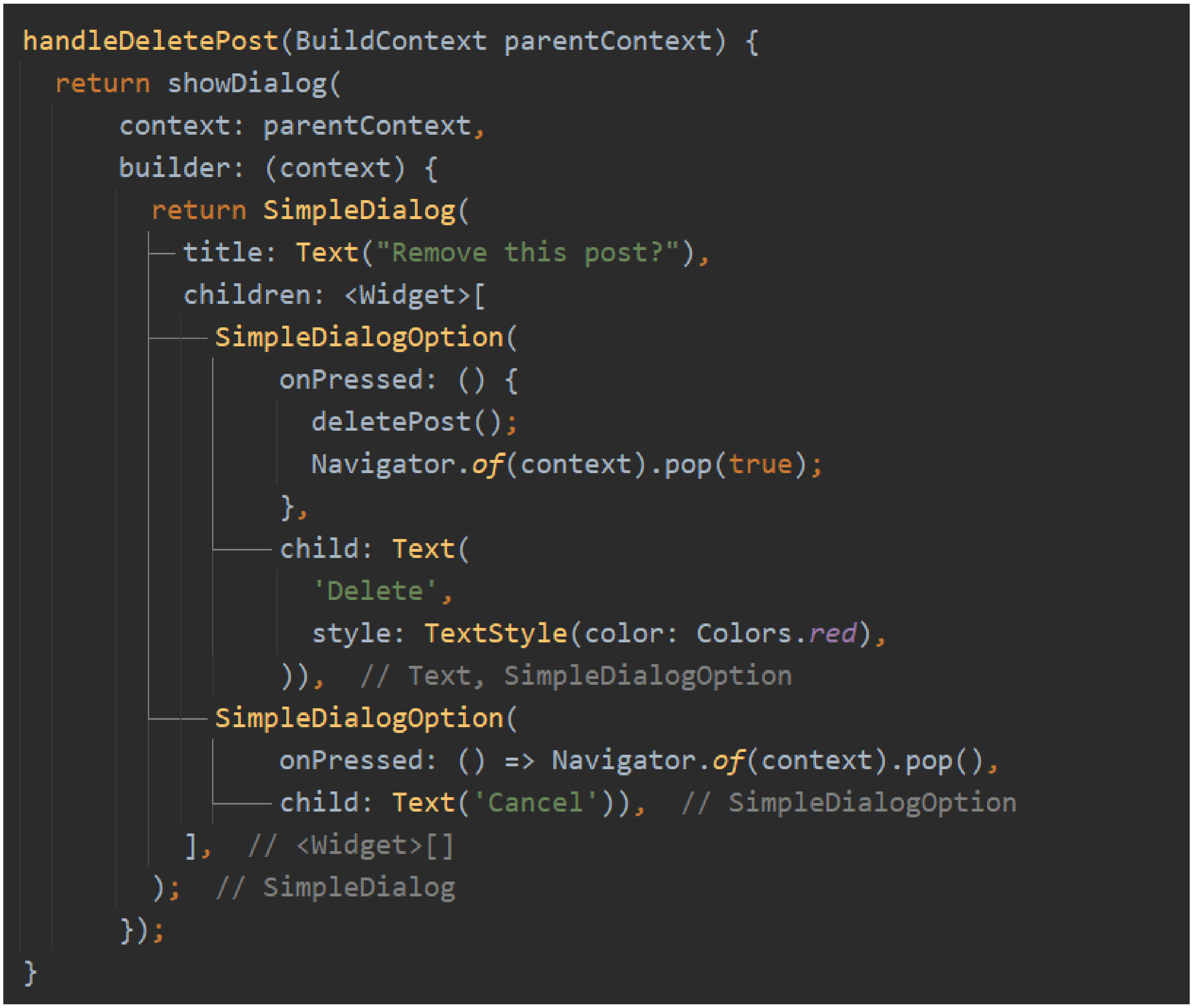
```
handleSubmit() async {
  FocusScope.of(context).unfocus();
  setState(() {
    captionController.text.isEmpty ? _displayCaptionValid = false
        : _displayCaptionValid = true;
  });
  if(_displayCaptionValid == true) {
    setState(() {
      isUploading = true;
    });
    if(file != null) {
      await compressImage();
      String mediaUrl = await uploadImage(file);
      createPostInFirestore(
        mediaUrl: mediaUrl,
        location: locationController.text,
        description: captionController.text,
        photoId: postId,
      );
    }
    else{
      createPostInFirestore(
        mediaUrl: null,
        location: locationController.text,
        description: captionController.text,
        photoId: null,
      );
    }
    captionController.clear();
    locationController.clear();
    setState(() {
      file = null;
      isUploading = false;
      postId = Uuid().v4();
    });
    Navigator.of(context).pop(true);
  }
  else{
    Fluttertoast.showToast(
      msg: 'You must write a post',
      textColor: Colors.white,
      backgroundColor: Colors.grey,
      timeInSecForIosWeb: 1,
    );
  }
}
```

Delete post:



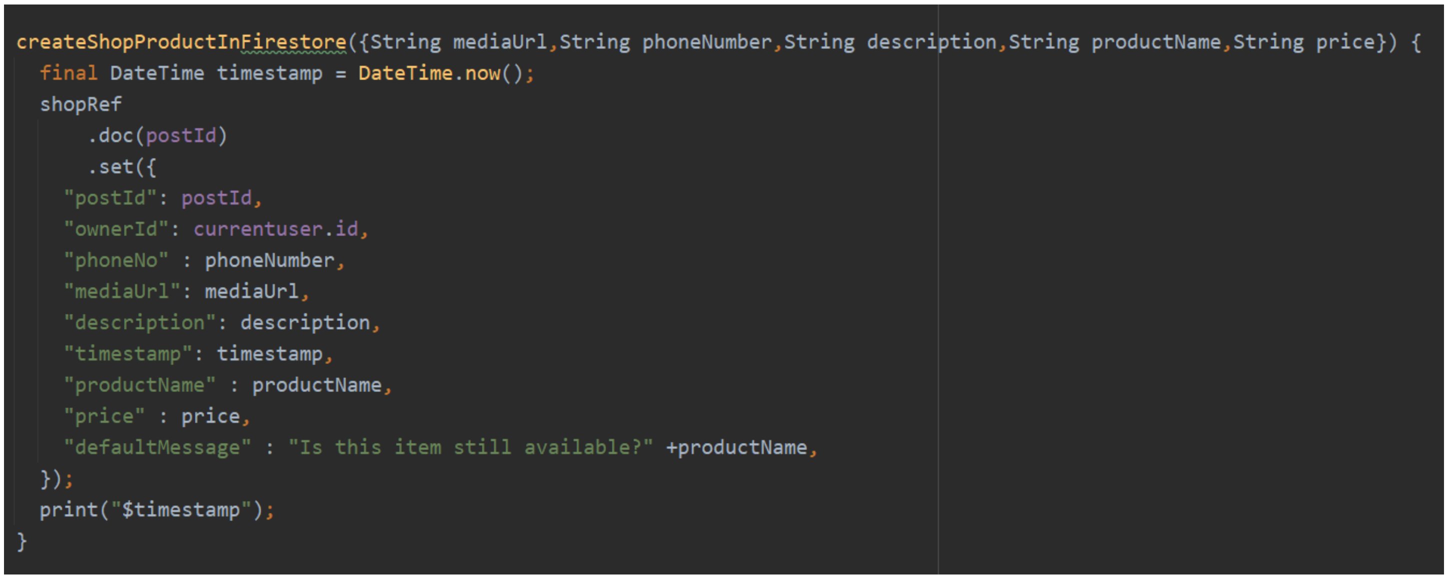
```
deletePost() async {
  // delete post itself
  postsRef
      .doc(postId)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  // delete uploaded image for thep ost
  if(mediaUrl != null)
  {
    storageRef.child("post_$photoId.jpg").delete();
  }
  // then delete all activity feed notifications
  QuerySnapshot activityFeedSnapshot = await activityFeedRef
      .doc(ownerId)
      .collection("feedItems")
      .where('postId', isEqualTo: postId)
      .get();
  activityFeedSnapshot.docs.forEach((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  // then delete all comments
  QuerySnapshot commentsSnapshot = await commentsRef
      .doc(postId)
      .collection('comments')
      .get();
  commentsSnapshot.docs.forEach((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  setState(() {
    removed = true;
  });
}
```

Handle delete post:



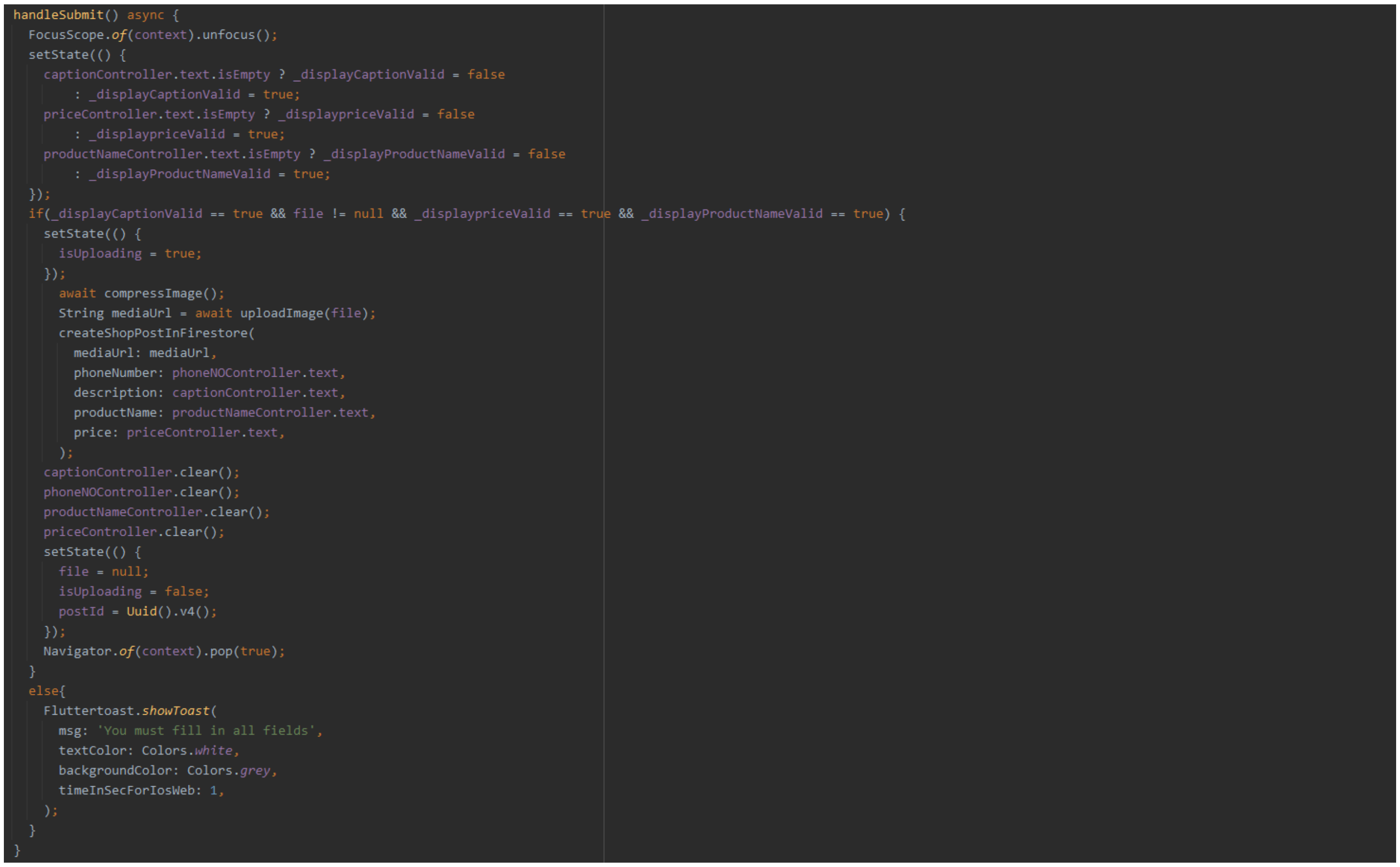
```
handleDeletePost(BuildContext parentContext) {
  return showDialog(
      context: parentContext,
      builder: (context) {
        return SimpleDialog(
          title: Text("Remove this post?"),
          children: <Widget>[
            SimpleDialogOption(
                onPressed: () {
                  deletePost();
                  Navigator.of(context).pop(true);
                },
                child: Text(
                  'Delete',
                  style: TextStyle(color: Colors.red),
                )),  // Text, SimpleDialogOption
            SimpleDialogOption(
                onPressed: () => Navigator.of(context).pop(),
                child: Text('Cancel')),  // SimpleDialogOption
          ],  // <Widget>[]
        );  // SimpleDialog
      });
}
```

Add product:



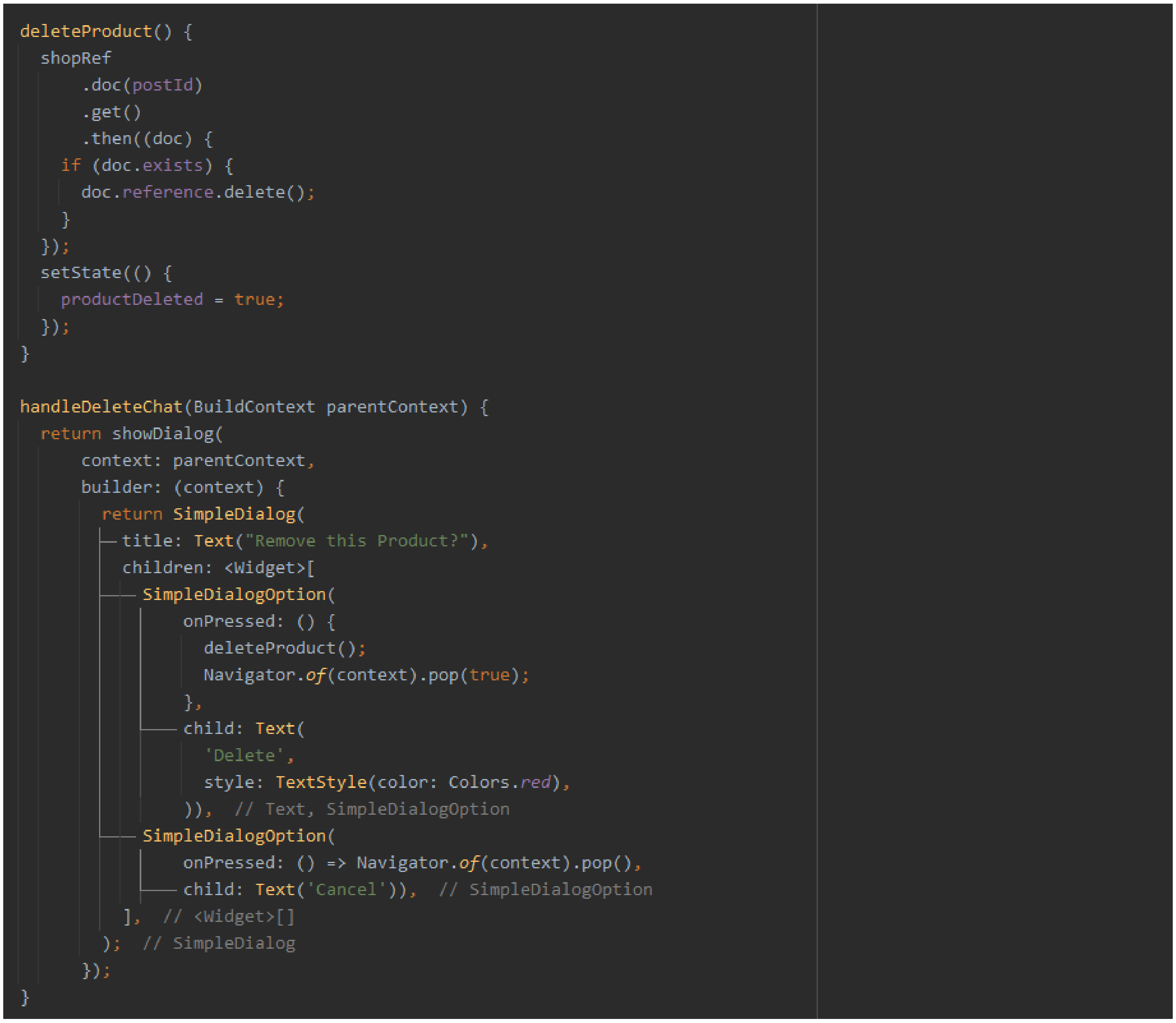
```
createShopProductInFirestore({String mediaUrl,String phoneNumber,String description,String productName,String price}) {
  final DateTime timestamp = DateTime.now();
  shopRef
      .doc(postId)
      .set({
    "postId": postId,
    "ownerId": currentuser.id,
    "phoneNo" : phoneNumber,
    "mediaUrl": mediaUrl,
    "description": description,
    "timestamp": timestamp,
    "productName" : productName,
    "price" : price,
    "defaultMessage" : "Is this item still available?" +productName,
  });
  print("$timestamp");
}
```

Handle product submit:



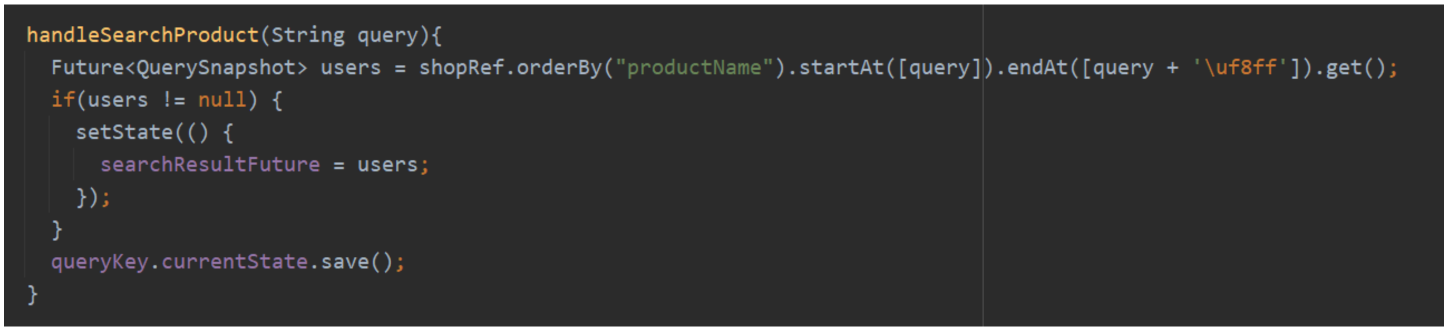
```
handleSubmit() async {
  FocusScope.of(context).unfocus();
  setState(() {
    captionController.text.isEmpty ? _displayCaptionValid = false
        : _displayCaptionValid = true;
    priceController.text.isEmpty ? _displaypriceValid = false
        : _displaypriceValid = true;
    productNameController.text.isEmpty ? _displayProductNameValid = false
        : _displayProductNameValid = true;
  });
  if(_displayCaptionValid == true && file != null && _displaypriceValid == true && _displayProductNameValid == true) {
    setState(() {
      isUploading = true;
    });
      await compressImage();
      String mediaUrl = await uploadImage(file);
      createShopPostInFirestore(
        mediaUrl: mediaUrl,
        phoneNumber: phoneNOController.text,
        description: captionController.text,
        productName: productNameController.text,
        price: priceController.text,
      );
    captionController.clear();
    phoneNOController.clear();
    productNameController.clear();
    priceController.clear();
    setState(() {
      file = null;
      isUploading = false;
      postId = Uuid().v4();
    });
    Navigator.of(context).pop(true);
  }
  else{
    Fluttertoast.showToast(
      msg: 'You must fill in all fields',
      textColor: Colors.white,
      backgroundColor: Colors.grey,
      timeInSecForIosWeb: 1,
    );
  }
}
```

Delete product:



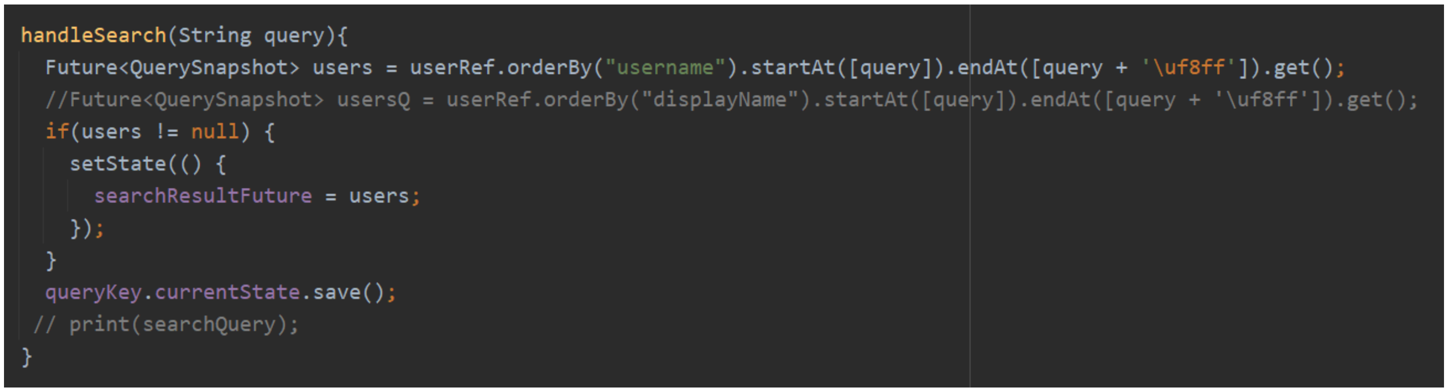
```
deleteProduct() {
  shopRef
      .doc(postId)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  setState(() {
    productDeleted = true;
  });
}

handleDeleteChat(BuildContext parentContext) {
  return showDialog(
      context: parentContext,
      builder: (context) {
        return SimpleDialog(
          title: Text("Remove this Product?"),
          children: <Widget>[
            SimpleDialogOption(
                onPressed: () {
                  deleteProduct();
                  Navigator.of(context).pop(true);
                },
                child: Text(
                  'Delete',
                  style: TextStyle(color: Colors.red),
                )),  // Text, SimpleDialogOption
            SimpleDialogOption(
                onPressed: () => Navigator.of(context).pop(),
                child: Text('Cancel')),  // SimpleDialogOption
          ],  // <Widget>[]
        );  // SimpleDialog
      });
}
```

Handle search product:



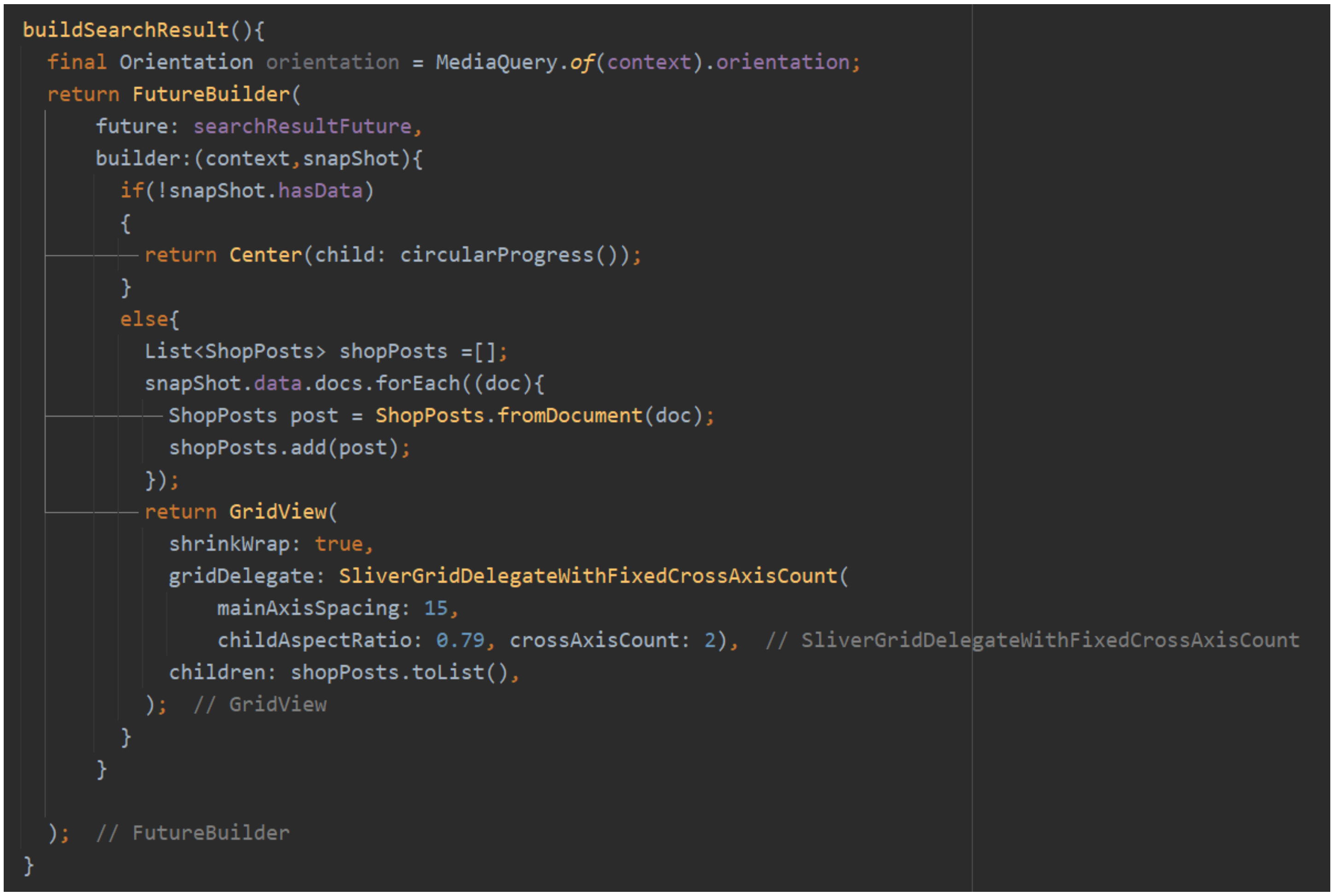
```
handleSearchProduct(String query){
  Future<QuerySnapshot> users = shopRef.orderBy("productName").startAt([query]).endAt([query + '\uf8ff']).get();
  if(users != null) {
    setState(() {
      searchResultFuture = users;
    });
  }
  queryKey.currentState.save();
}
```

Handle search:



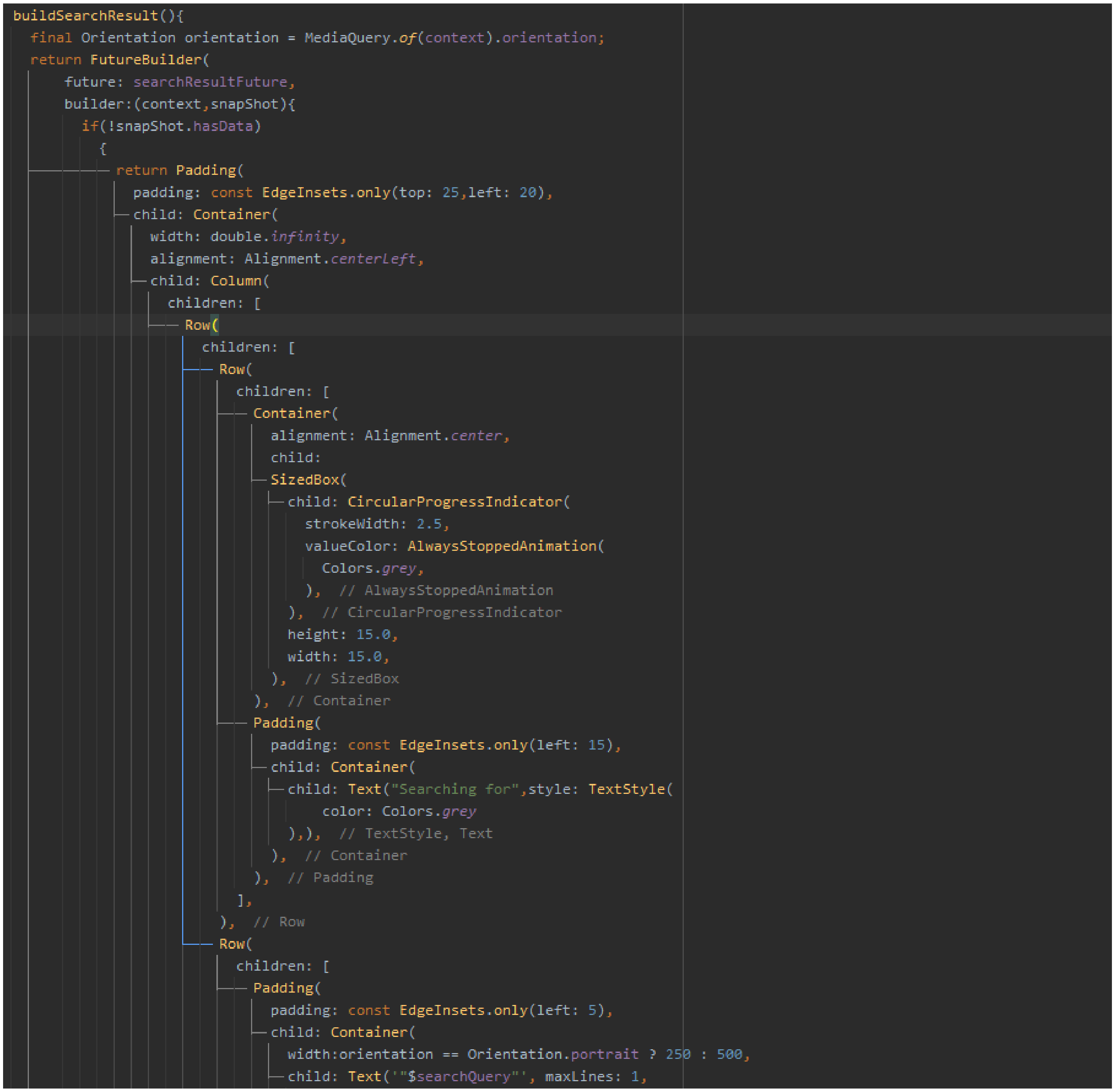
```
handleSearch(String query){
  Future<QuerySnapshot> users = userRef.orderBy("username").startAt([query]).endAt([query + '\uf8ff']).get();
  //Future<QuerySnapshot> usersQ = userRef.orderBy("displayName").startAt([query]).endAt([query + '\uf8ff']).get();
  if(users != null) {
    setState(() {
      searchResultFuture = users;
    });
  }
  queryKey.currentState.save();
 // print(searchQuery);
}
```

Build search product result:

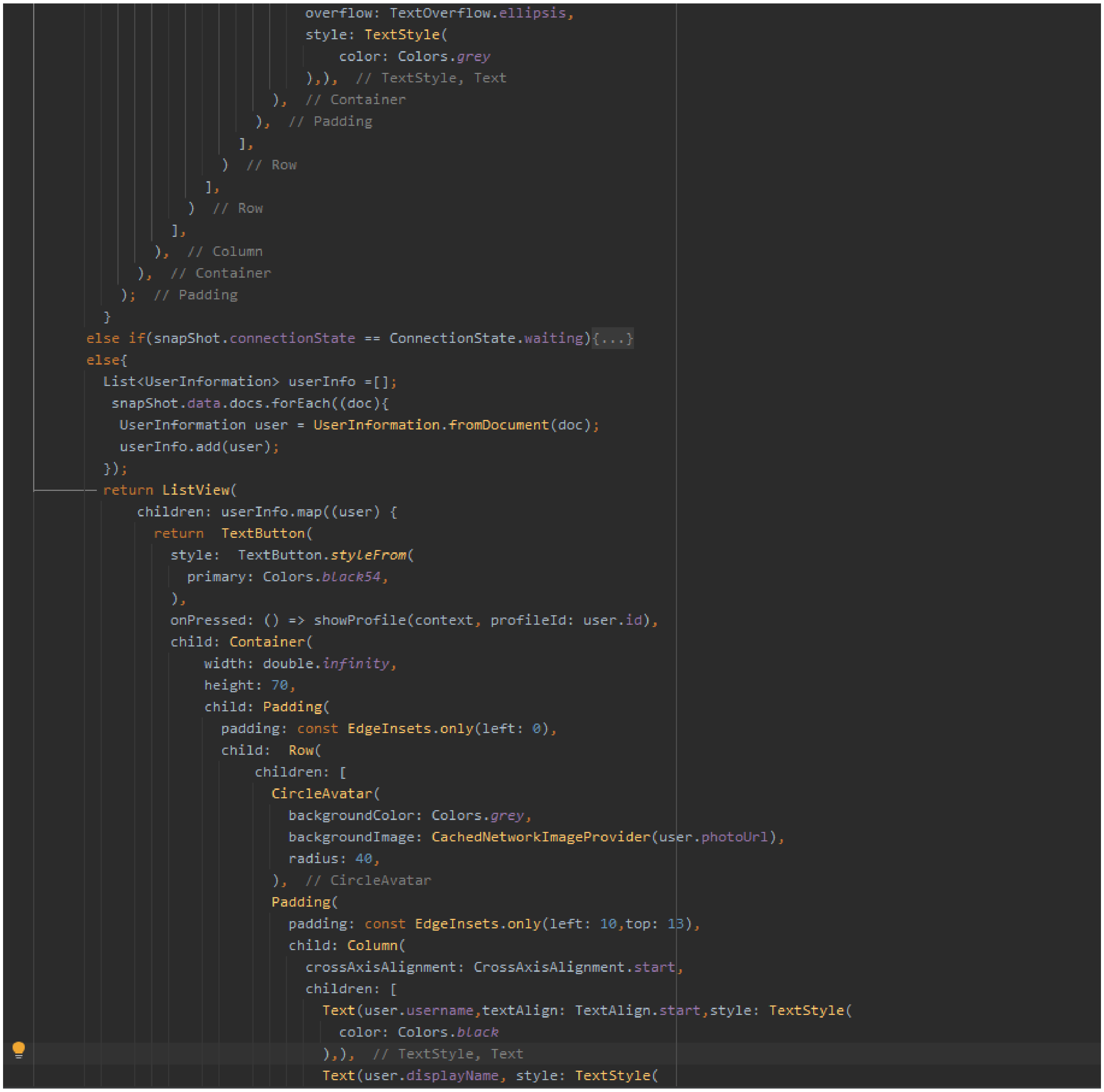


```
buildSearchResult(){
  final Orientation orientation = MediaQuery.of(context).orientation;
  return FutureBuilder(
      future: searchResultFuture,
      builder:(context,snapShot){
        if(!snapShot.hasData)
        {
          return Center(child: circularProgress());
        }
        else{
          List<ShopPosts> shopPosts =[];
          snapShot.data.docs.forEach((doc){
            ShopPosts post = ShopPosts.fromDocument(doc);
            shopPosts.add(post);
          });
          return GridView(
            shrinkWrap: true,
            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                mainAxisSpacing: 15,
                childAspectRatio: 0.79, crossAxisCount: 2),  // SliverGridDelegateWithFixedCrossAxisCount
            children: shopPosts.toList(),
          );  // GridView
        }
      }

  );  // FutureBuilder
}
```

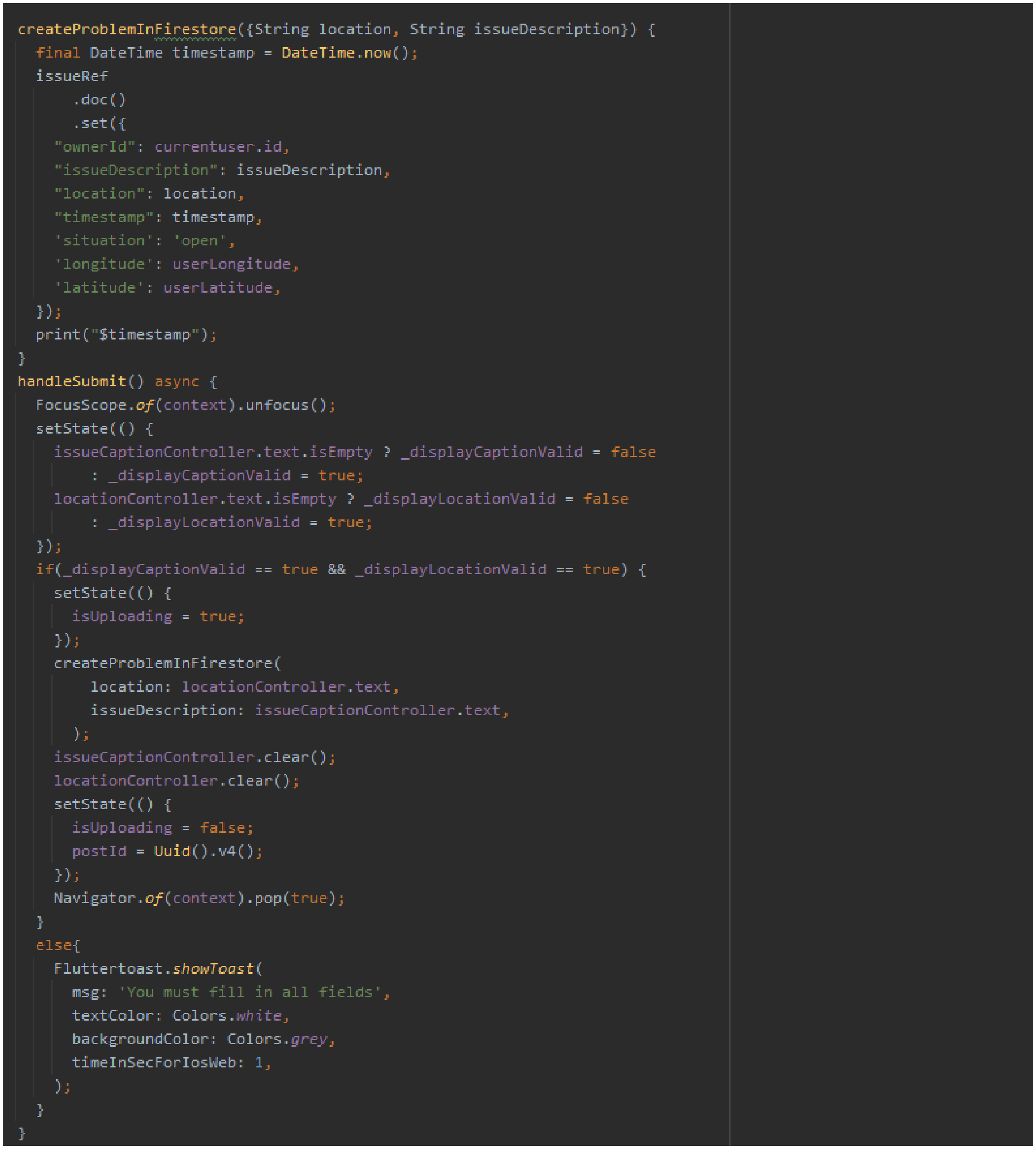
Build search result:



```
buildSearchResult(){
  final Orientation orientation = MediaQuery.of(context).orientation;
  return FutureBuilder(
      future: searchResultFuture,
      builder:(context,snapShot){
        if(!snapShot.hasData)
          {
            return Padding(
              padding: const EdgeInsets.only(top: 25,left: 20),
              child: Container(
                width: double.infinity,
                alignment: Alignment.centerLeft,
                child: Column(
                  children: [
                    Row(
                      children: [
                        Row(
                          children: [
                            Container(
                              alignment: Alignment.center,
                              child:
                              SizedBox(
                                child: CircularProgressIndicator(
                                  strokeWidth: 2.5,
                                  valueColor: AlwaysStoppedAnimation(
                                    Colors.grey,
                                  ),  // AlwaysStoppedAnimation
                                ),  // CircularProgressIndicator
                                height: 15.0,
                                width: 15.0,
                              ),  // SizedBox
                            ),  // Container
                            Padding(
                              padding: const EdgeInsets.only(left: 15),
                              child: Container(
                                child: Text("Searching for",style: TextStyle(
                                    color: Colors.grey
                                ),),  // TextStyle, Text
                              ),  // Container
                            ),  // Padding
                          ],
                        ),  // Row
                        Row(
                          children: [
                            Padding(
                              padding: const EdgeInsets.only(left: 5),
                              child: Container(
                                width:orientation == Orientation.portrait ? 250 : 500,
                                child: Text('"$searchQuery"', maxLines: 1,
```

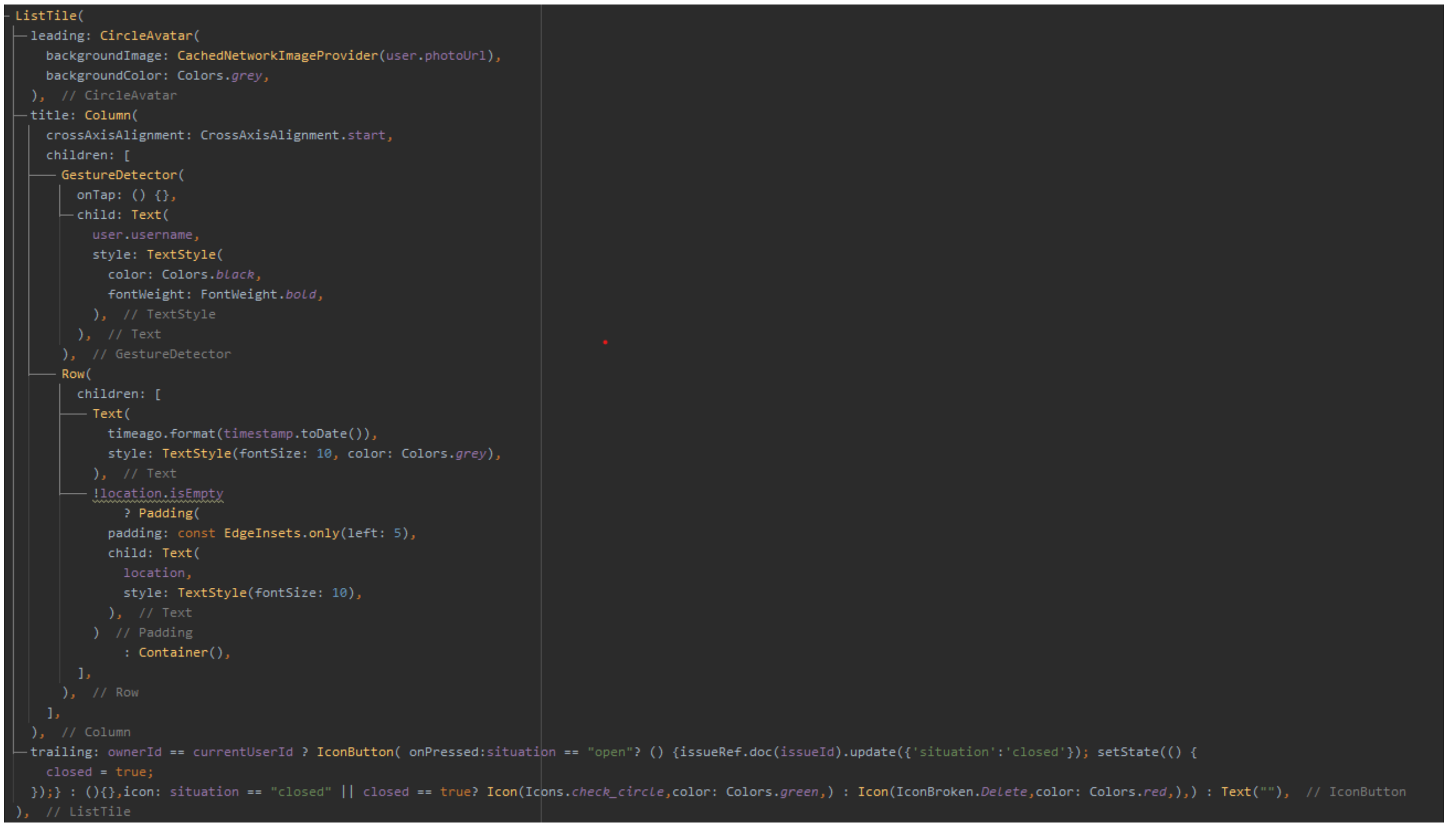

```
                          overflow: TextOverflow.ellipsis,
                          style: TextStyle(
                              color: Colors.grey
                          ),),  // TextStyle, Text
                      ),  // Container
                    ),  // Padding
                  ],
                )  // Row
              ],
            )  // Row
          ],
        ),  // Column
      ),  // Container
    );  // Padding
  }
else if(snapShot.connectionState == ConnectionState.waiting){...}
else{
  List<UserInformation> userInfo =[];
   snapShot.data.docs.forEach((doc){
    UserInformation user = UserInformation.fromDocument(doc);
    userInfo.add(user);
  });
  return ListView(
      children: userInfo.map((user) {
        return  TextButton(
          style:  TextButton.styleFrom(
            primary: Colors.black54,
          ),
          onPressed: () => showProfile(context, profileId: user.id),
          child: Container(
              width: double.infinity,
              height: 70,
              child: Padding(
                padding: const EdgeInsets.only(left: 0),
                child:  Row(
                    children: [
                      CircleAvatar(
                        backgroundColor: Colors.grey,
                        backgroundImage: CachedNetworkImageProvider(user.photoUrl),
                        radius: 40,
                      ),  // CircleAvatar
                      Padding(
                        padding: const EdgeInsets.only(left: 10,top: 13),
                        child: Column(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: [
                            Text(user.username,textAlign: TextAlign.start,style: TextStyle(
                              color: Colors.black
                            ),),  // TextStyle, Text
                            Text(user.displayName, style: TextStyle(
```

Add problem:



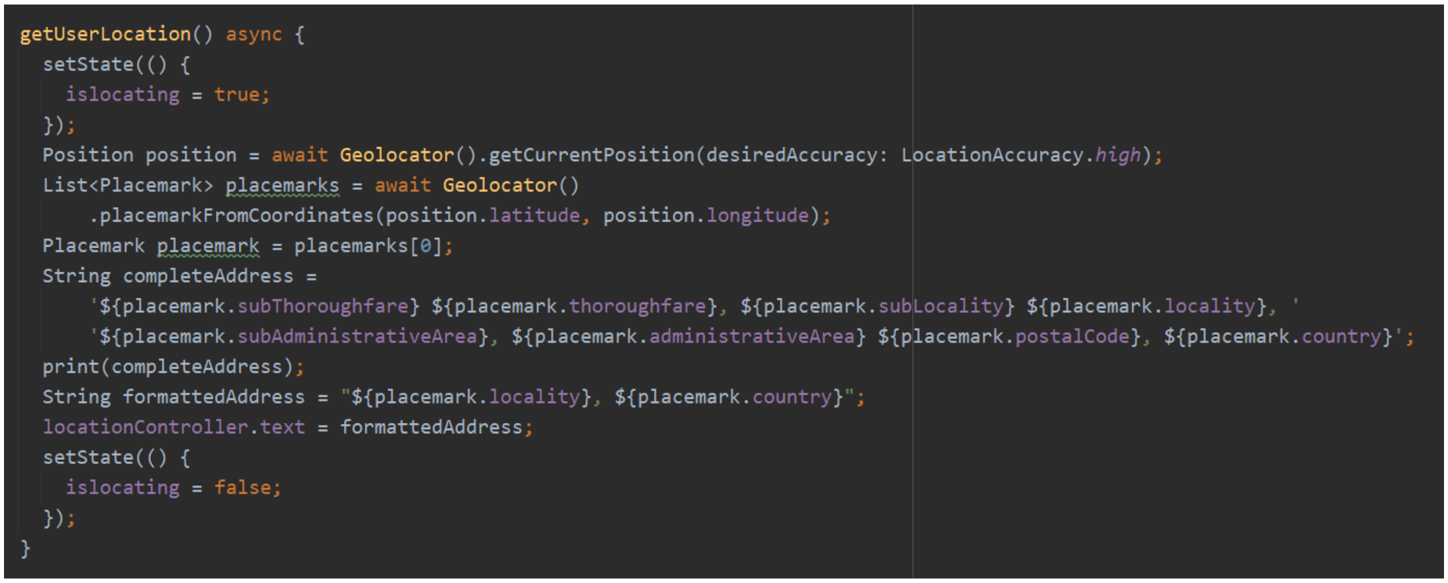
```
createProblemInFirestore({String location, String issueDescription}) {
  final DateTime timestamp = DateTime.now();
  issueRef
      .doc()
      .set({
    "ownerId": currentuser.id,
    "issueDescription": issueDescription,
    "location": location,
    "timestamp": timestamp,
    'situation': 'open',
    'longitude': userLongitude,
    'latitude': userLatitude,
  });
  print("$timestamp");
}
handleSubmit() async {
  FocusScope.of(context).unfocus();
  setState(() {
    issueCaptionController.text.isEmpty ? _displayCaptionValid = false
        : _displayCaptionValid = true;
    locationController.text.isEmpty ? _displayLocationValid = false
        : _displayLocationValid = true;
  });
  if(_displayCaptionValid == true && _displayLocationValid == true) {
    setState(() {
      isUploading = true;
    });
    createProblemInFirestore(
        location: locationController.text,
        issueDescription: issueCaptionController.text,
      );
    issueCaptionController.clear();
    locationController.clear();
    setState(() {
      isUploading = false;
      postId = Uuid().v4();
    });
    Navigator.of(context).pop(true);
  }
  else{
    Fluttertoast.showToast(
      msg: 'You must fill in all fields',
      textColor: Colors.white,
      backgroundColor: Colors.grey,
      timeInSecForIosWeb: 1,
    );
  }
}
```

Delete problem:



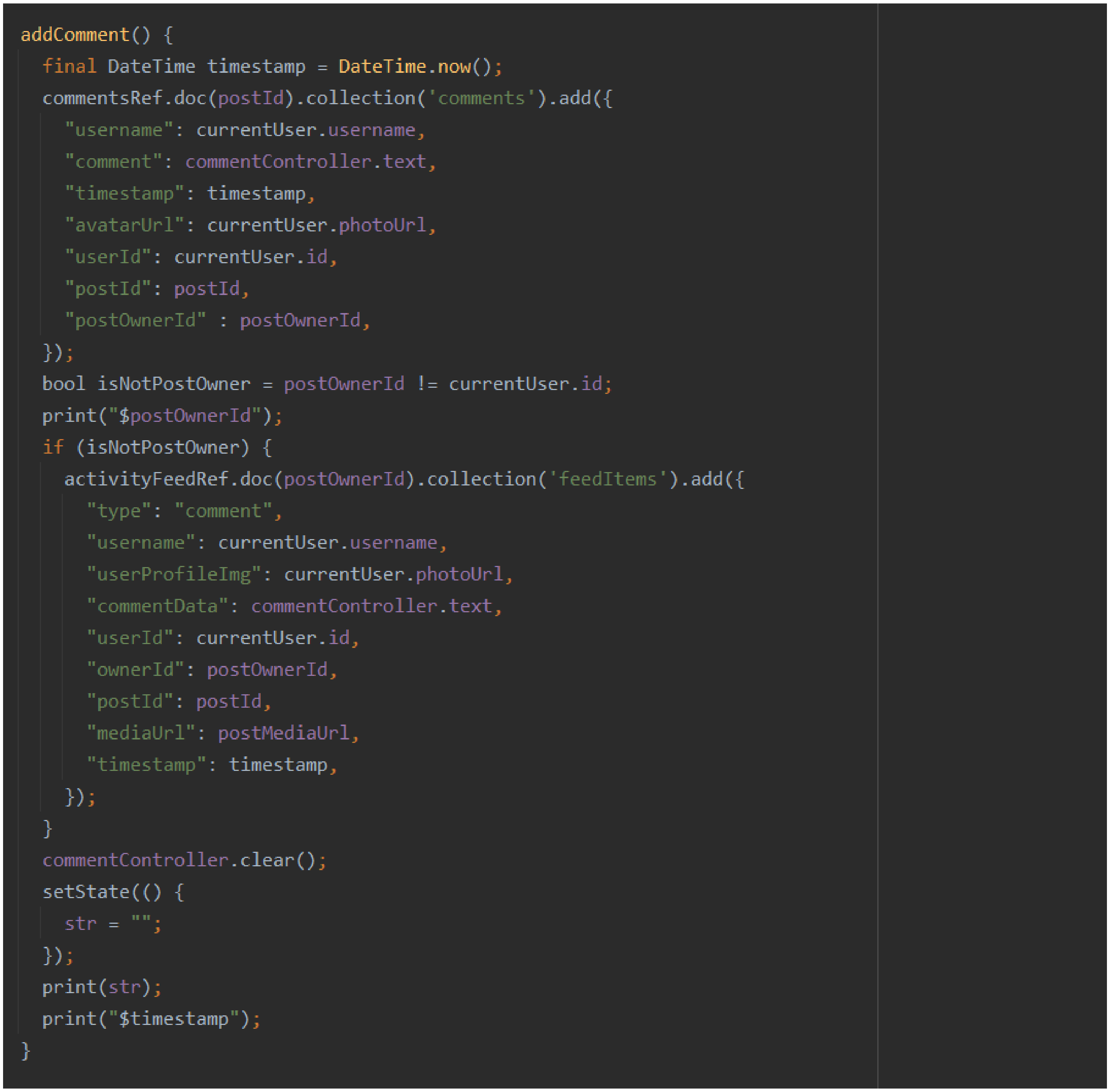
```
ListTile(
  leading: CircleAvatar(
    backgroundImage: CachedNetworkImageProvider(user.photoUrl),
    backgroundColor: Colors.grey,
  ),  // CircleAvatar
  title: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      GestureDetector(
        onTap: () {},
        child: Text(
          user.username,
          style: TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.bold,
          ),  // TextStyle
        ),  // Text
      ),  // GestureDetector
      Row(
        children: [
          Text(
            timeago.format(timestamp.toDate()),
            style: TextStyle(fontSize: 10, color: Colors.grey),
          ),  // Text
          !location.isEmpty
              ? Padding(
            padding: const EdgeInsets.only(left: 5),
            child: Text(
              location,
              style: TextStyle(fontSize: 10),
            ),  // Text
          )  // Padding
              : Container(),
        ],
      ),  // Row
    ],
  ),  // Column
  trailing: ownerId == currentUserId ? IconButton( onPressed:situation == "open"? () {issueRef.doc(issueId).update({'situation':'closed'}); setState(() {
    closed = true;
  });} : (){},icon: situation == "closed" || closed == true? Icon(Icons.check_circle,color: Colors.green,) : Icon(IconBroken.Delete,color: Colors.red,),) : Text(""),  // IconButton
),  // ListTile
```

Get user location:



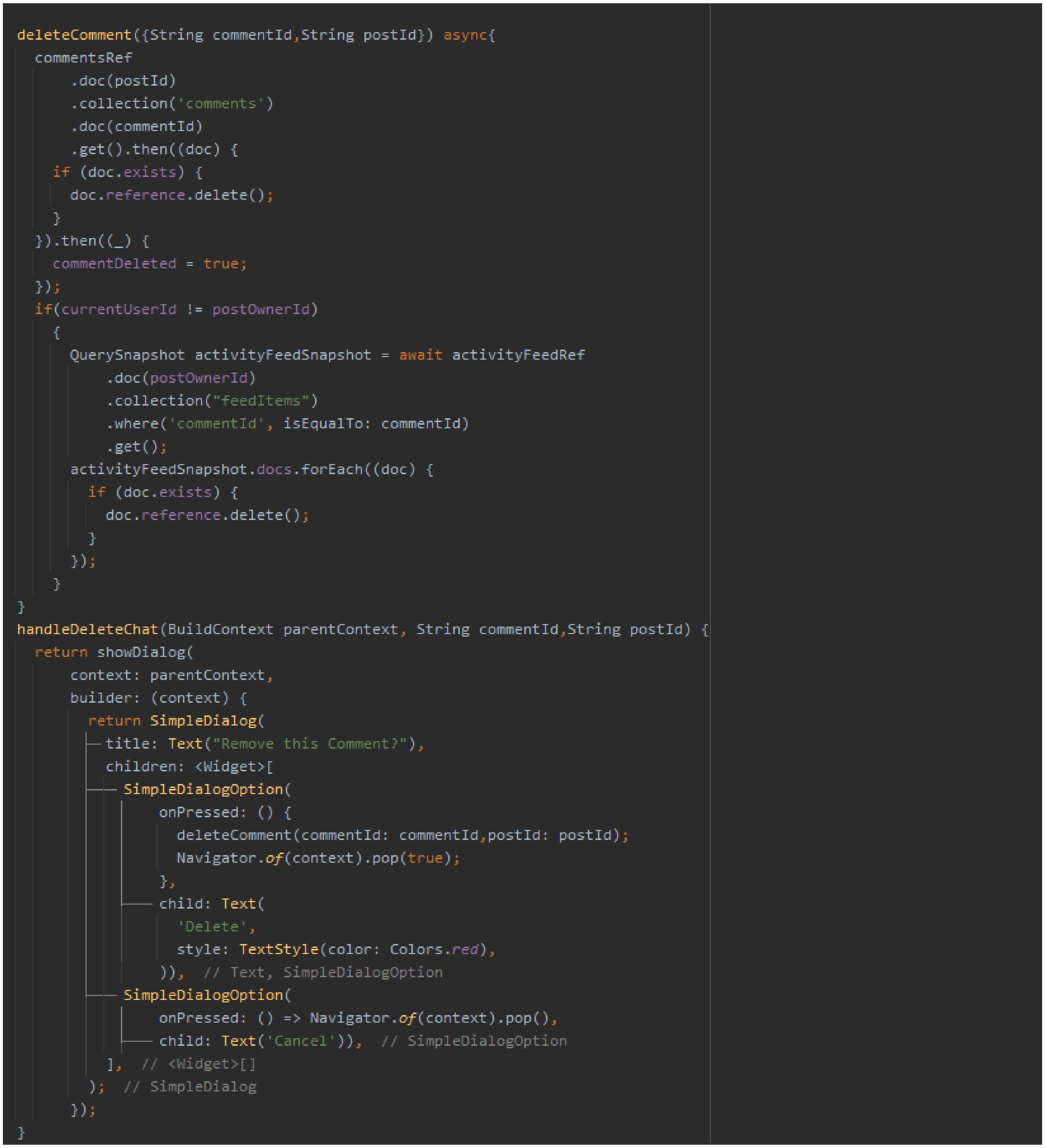
```
getUserLocation() async {
  setState(() {
    islocating = true;
  });
  Position position = await Geolocator().getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
  List<Placemark> placemarks = await Geolocator()
      .placemarkFromCoordinates(position.latitude, position.longitude);
  Placemark placemark = placemarks[0];
  String completeAddress =
      '${placemark.subThoroughfare} ${placemark.thoroughfare}, ${placemark.subLocality} ${placemark.locality}, '
      '${placemark.subAdministrativeArea}, ${placemark.administrativeArea} ${placemark.postalCode}, ${placemark.country}';
  print(completeAddress);
  String formattedAddress = "${placemark.locality}, ${placemark.country}";
  locationController.text = formattedAddress;
  setState(() {
    islocating = false;
  });
}
```

Add comment:



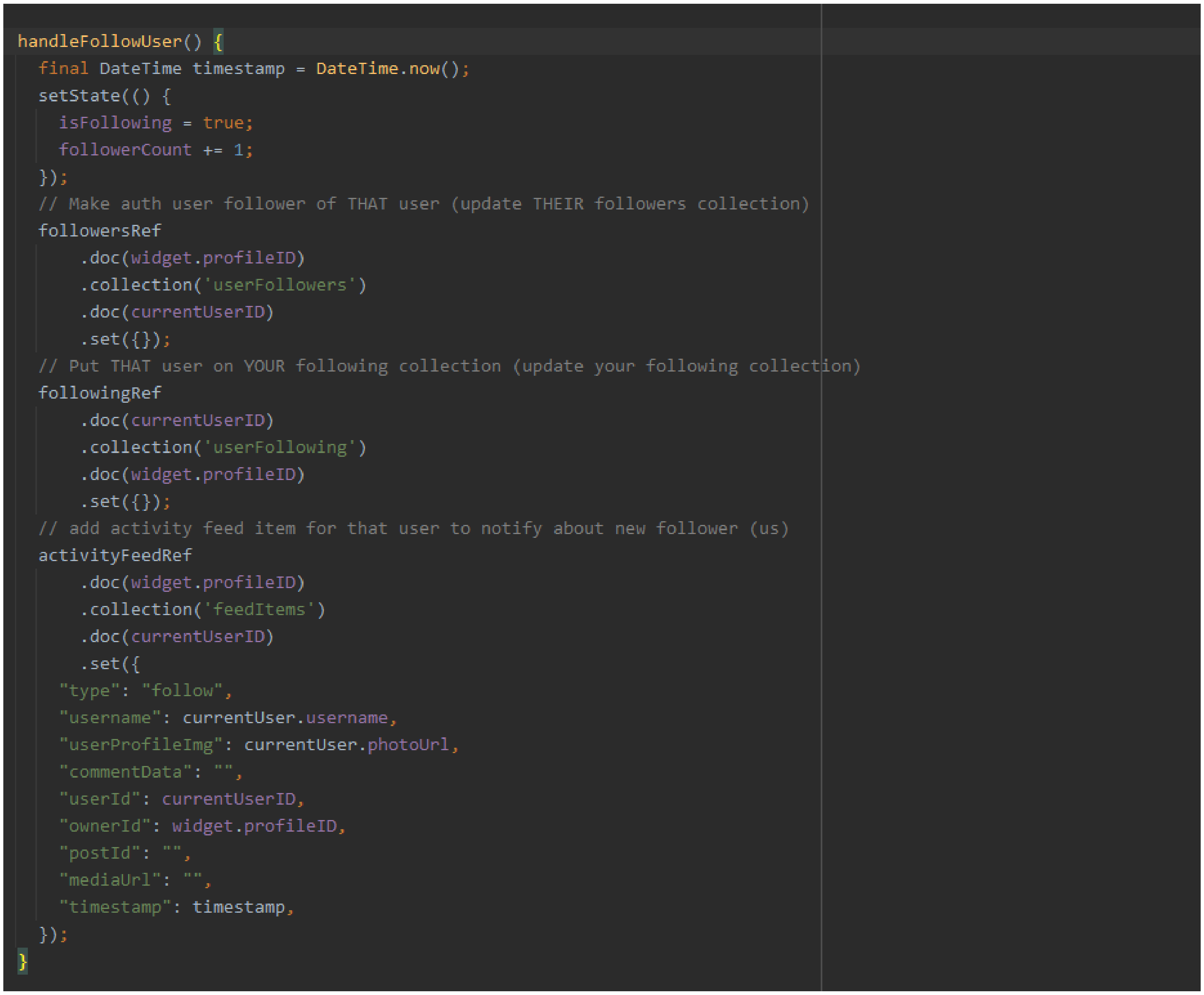
```
addComment() {
  final DateTime timestamp = DateTime.now();
  commentsRef.doc(postId).collection('comments').add({
    "username": currentUser.username,
    "comment": commentController.text,
    "timestamp": timestamp,
    "avatarUrl": currentUser.photoUrl,
    "userId": currentUser.id,
    "postId": postId,
    "postOwnerId" : postOwnerId,
  });
  bool isNotPostOwner = postOwnerId != currentUser.id;
  print("$postOwnerId");
  if (isNotPostOwner) {
    activityFeedRef.doc(postOwnerId).collection('feedItems').add({
      "type": "comment",
      "username": currentUser.username,
      "userProfileImg": currentUser.photoUrl,
      "commentData": commentController.text,
      "userId": currentUser.id,
      "ownerId": postOwnerId,
      "postId": postId,
      "mediaUrl": postMediaUrl,
      "timestamp": timestamp,
    });
  }
  commentController.clear();
  setState(() {
    str = "";
  });
  print(str);
  print("$timestamp");
}
```

Delete comment:



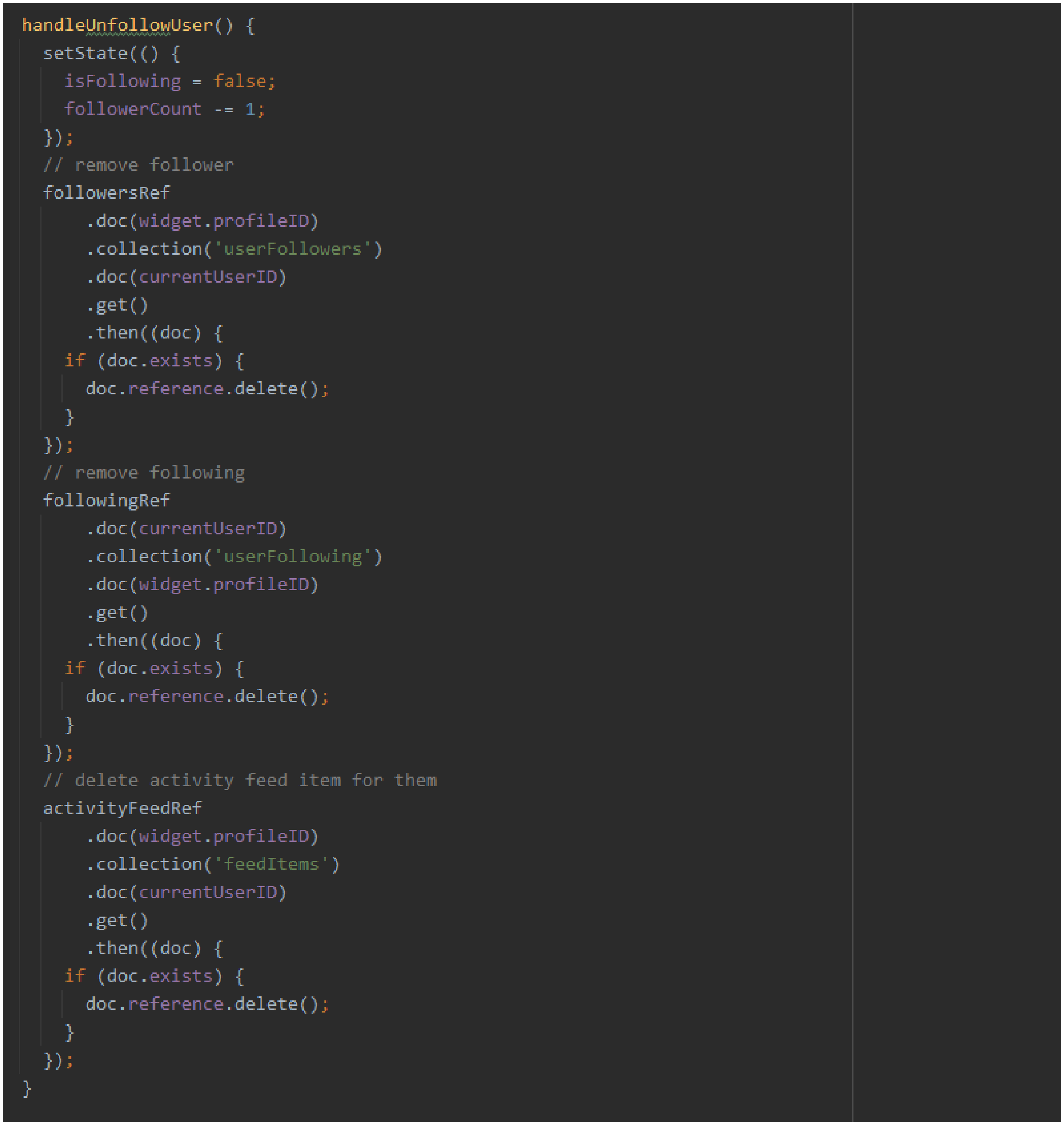
```dart
deleteComment({String commentId,String postId}) async{
  commentsRef
      .doc(postId)
      .collection('comments')
      .doc(commentId)
      .get().then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  }).then((_) {
    commentDeleted = true;
  });
  if(currentUserId != postOwnerId)
    {
      QuerySnapshot activityFeedSnapshot = await activityFeedRef
          .doc(postOwnerId)
          .collection("feedItems")
          .where('commentId', isEqualTo: commentId)
          .get();
      activityFeedSnapshot.docs.forEach((doc) {
        if (doc.exists) {
          doc.reference.delete();
        }
      });
    }
}
handleDeleteChat(BuildContext parentContext, String commentId,String postId) {
  return showDialog(
      context: parentContext,
      builder: (context) {
        return SimpleDialog(
          title: Text("Remove this Comment?"),
          children: <Widget>[
            SimpleDialogOption(
                onPressed: () {
                  deleteComment(commentId: commentId,postId: postId);
                  Navigator.of(context).pop(true);
                },
                child: Text(
                  'Delete',
                  style: TextStyle(color: Colors.red),
                )),  // Text, SimpleDialogOption
            SimpleDialogOption(
                onPressed: () => Navigator.of(context).pop(),
                child: Text('Cancel')),  // SimpleDialogOption
          ],  // <Widget>[]
        );  // SimpleDialog
      });
}
```

Follow:



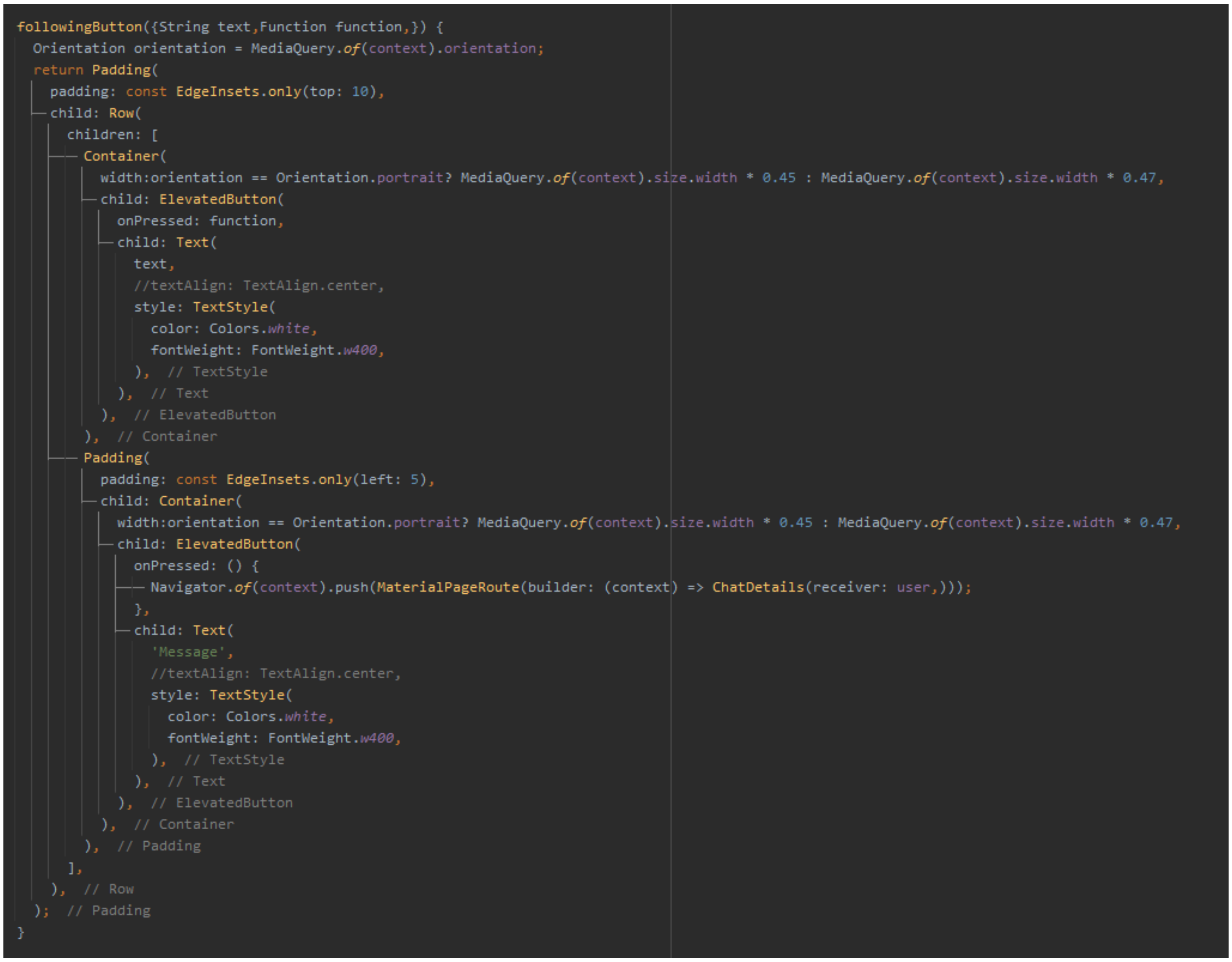
```
handleFollowUser() {
  final DateTime timestamp = DateTime.now();
  setState(() {
    isFollowing = true;
    followerCount += 1;
  });
  // Make auth user follower of THAT user (update THEIR followers collection)
  followersRef
      .doc(widget.profileID)
      .collection('userFollowers')
      .doc(currentUserID)
      .set({});
  // Put THAT user on YOUR following collection (update your following collection)
  followingRef
      .doc(currentUserID)
      .collection('userFollowing')
      .doc(widget.profileID)
      .set({});
  // add activity feed item for that user to notify about new follower (us)
  activityFeedRef
      .doc(widget.profileID)
      .collection('feedItems')
      .doc(currentUserID)
      .set({
    "type": "follow",
    "username": currentUser.username,
    "userProfileImg": currentUser.photoUrl,
    "commentData": "",
    "userId": currentUserID,
    "ownerId": widget.profileID,
    "postId": "",
    "mediaUrl": "",
    "timestamp": timestamp,
  });
}
```

Unfollow:



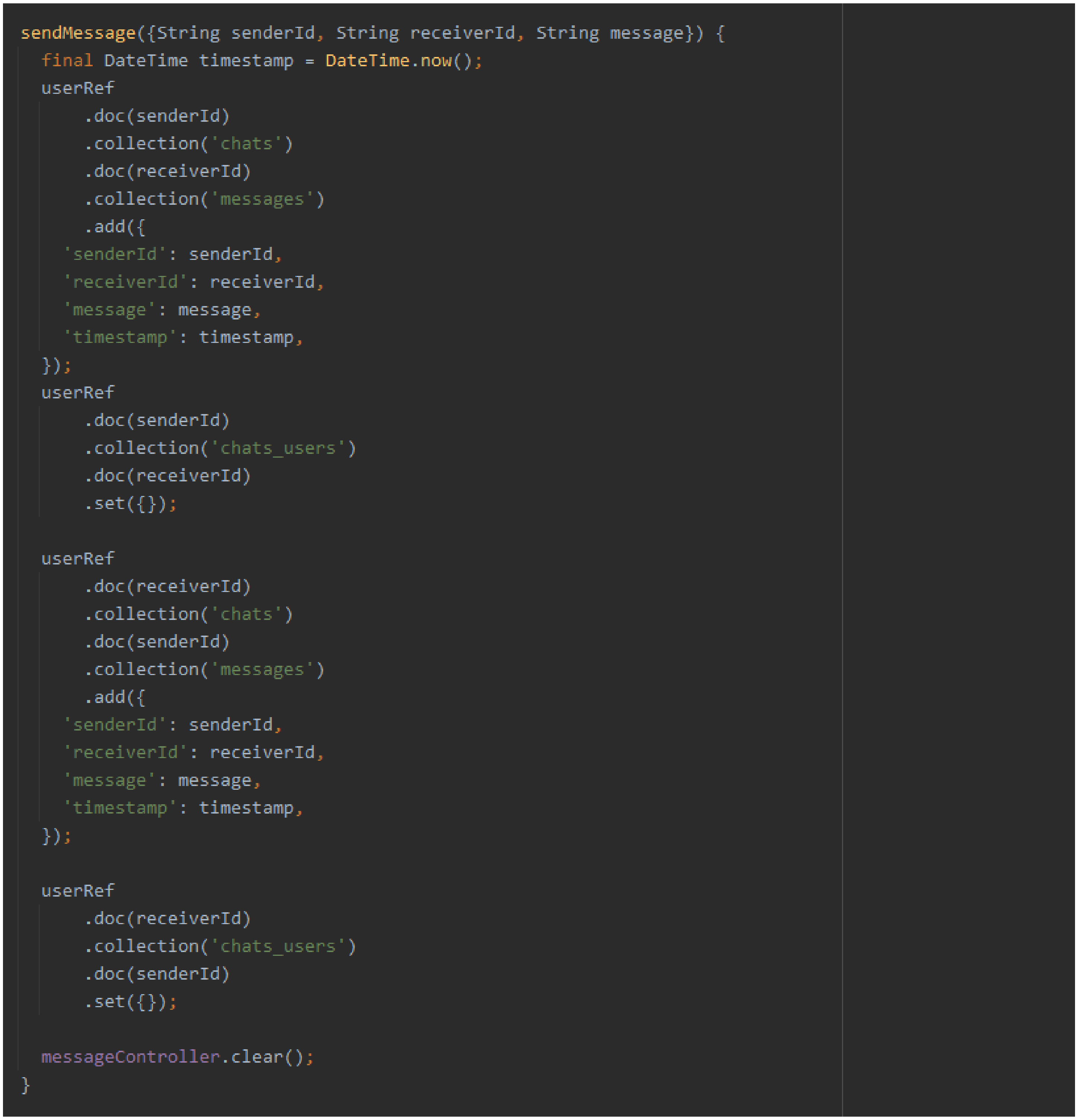
```
handleUnfollowUser() {
  setState(() {
    isFollowing = false;
    followerCount -= 1;
  });
  // remove follower
  followersRef
      .doc(widget.profileID)
      .collection('userFollowers')
      .doc(currentUserID)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  // remove following
  followingRef
      .doc(currentUserID)
      .collection('userFollowing')
      .doc(widget.profileID)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  // delete activity feed item for them
  activityFeedRef
      .doc(widget.profileID)
      .collection('feedItems')
      .doc(currentUserID)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
}
```

Follow button:



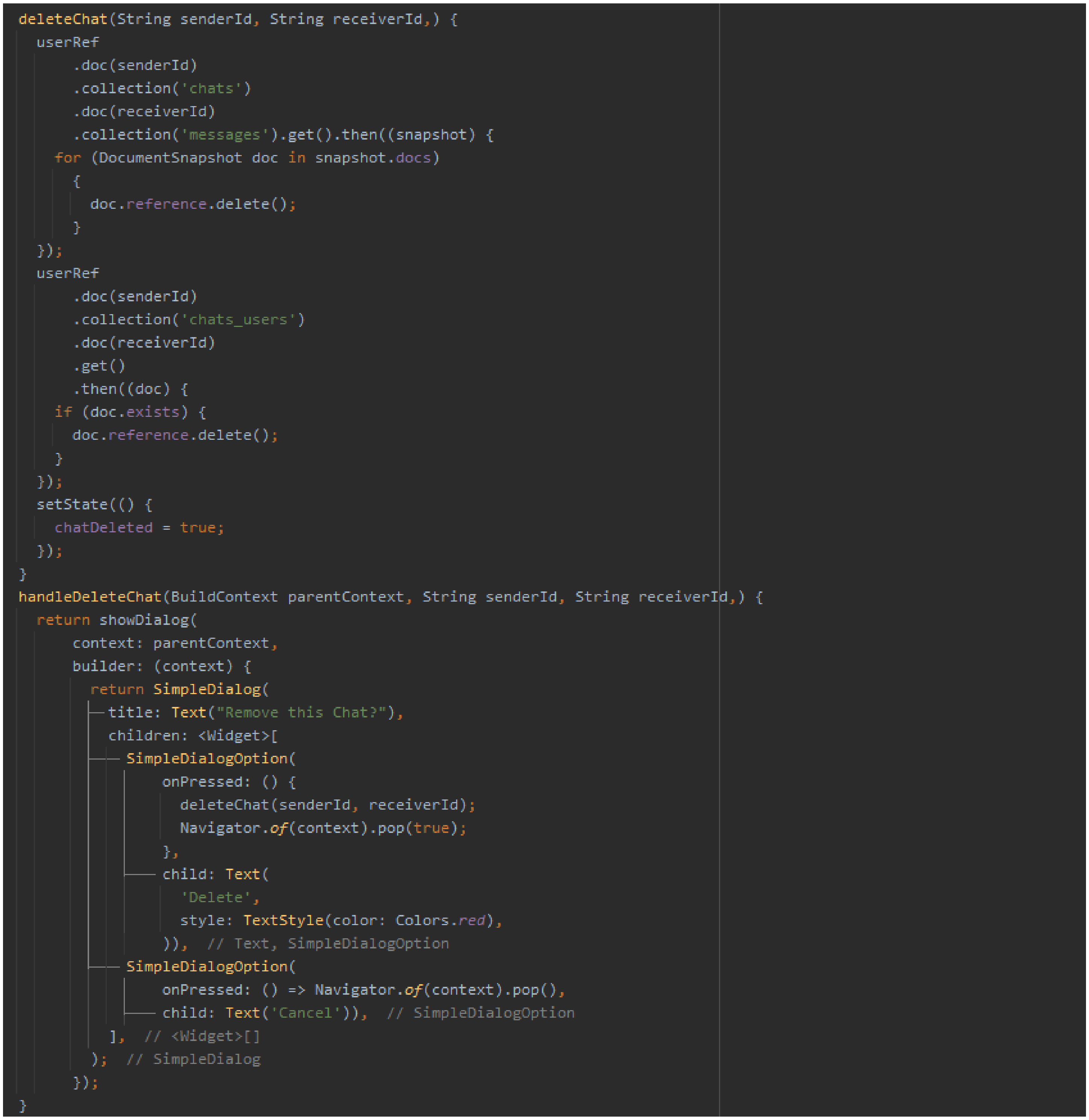
```
followingButton({String text,Function function,}) {
  Orientation orientation = MediaQuery.of(context).orientation;
  return Padding(
    padding: const EdgeInsets.only(top: 10),
    child: Row(
      children: [
        Container(
          width:orientation == Orientation.portrait? MediaQuery.of(context).size.width * 0.45 : MediaQuery.of(context).size.width * 0.47,
          child: ElevatedButton(
            onPressed: function,
            child: Text(
              text,
              //textAlign: TextAlign.center,
              style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.w400,
              ),  // TextStyle
            ),  // Text
          ),  // ElevatedButton
        ),  // Container
        Padding(
          padding: const EdgeInsets.only(left: 5),
          child: Container(
            width:orientation == Orientation.portrait? MediaQuery.of(context).size.width * 0.45 : MediaQuery.of(context).size.width * 0.47,
            child: ElevatedButton(
              onPressed: () {
                Navigator.of(context).push(MaterialPageRoute(builder: (context) => ChatDetails(receiver: user,)));
              },
              child: Text(
                'Message',
                //textAlign: TextAlign.center,
                style: TextStyle(
                  color: Colors.white,
                  fontWeight: FontWeight.w400,
                ),  // TextStyle
              ),  // Text
            ),  // ElevatedButton
          ),  // Container
        ),  // Padding
      ],
    ),  // Row
  );  // Padding
}
```

Send message:



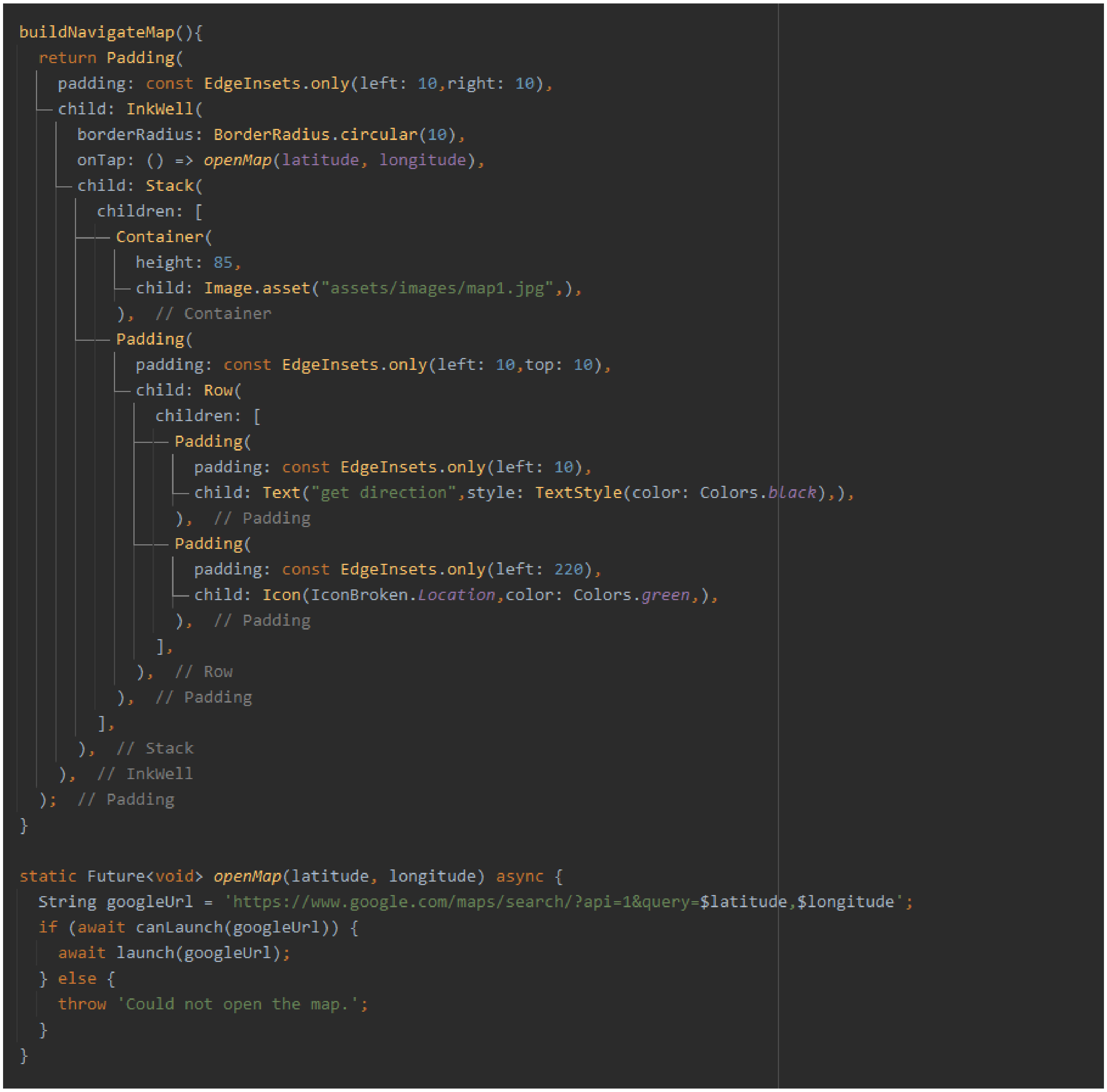
```
sendMessage({String senderId, String receiverId, String message}) {
  final DateTime timestamp = DateTime.now();
  userRef
      .doc(senderId)
      .collection('chats')
      .doc(receiverId)
      .collection('messages')
      .add({
    'senderId': senderId,
    'receiverId': receiverId,
    'message': message,
    'timestamp': timestamp,
  });
  userRef
      .doc(senderId)
      .collection('chats_users')
      .doc(receiverId)
      .set({});

  userRef
      .doc(receiverId)
      .collection('chats')
      .doc(senderId)
      .collection('messages')
      .add({
    'senderId': senderId,
    'receiverId': receiverId,
    'message': message,
    'timestamp': timestamp,
  });

  userRef
      .doc(receiverId)
      .collection('chats_users')
      .doc(senderId)
      .set({});

  messageController.clear();
}
```

Delete chat:



```
deleteChat(String senderId, String receiverId,) {
  userRef
      .doc(senderId)
      .collection('chats')
      .doc(receiverId)
      .collection('messages').get().then((snapshot) {
    for (DocumentSnapshot doc in snapshot.docs)
      {
        doc.reference.delete();
      }
  });
  userRef
      .doc(senderId)
      .collection('chats_users')
      .doc(receiverId)
      .get()
      .then((doc) {
    if (doc.exists) {
      doc.reference.delete();
    }
  });
  setState(() {
    chatDeleted = true;
  });
}
handleDeleteChat(BuildContext parentContext, String senderId, String receiverId,) {
  return showDialog(
      context: parentContext,
      builder: (context) {
        return SimpleDialog(
          title: Text("Remove this Chat?"),
          children: <Widget>[
            SimpleDialogOption(
                onPressed: () {
                  deleteChat(senderId, receiverId);
                  Navigator.of(context).pop(true);
                },
                child: Text(
                  'Delete',
                  style: TextStyle(color: Colors.red),
                )),  // Text, SimpleDialogOption
            SimpleDialogOption(
                onPressed: () => Navigator.of(context).pop(),
                child: Text('Cancel')),  // SimpleDialogOption
          ],  // <Widget>[]
        );  // SimpleDialog
      });
}
```

Open map:



```
buildNavigateMap(){
  return Padding(
    padding: const EdgeInsets.only(left: 10,right: 10),
    child: InkWell(
      borderRadius: BorderRadius.circular(10),
      onTap: () => openMap(latitude, longitude),
      child: Stack(
        children: [
          Container(
            height: 85,
            child: Image.asset("assets/images/map1.jpg",),
          ), // Container
          Padding(
            padding: const EdgeInsets.only(left: 10,top: 10),
            child: Row(
              children: [
                Padding(
                  padding: const EdgeInsets.only(left: 10),
                  child: Text("get direction",style: TextStyle(color: Colors.black),),
                ), // Padding
                Padding(
                  padding: const EdgeInsets.only(left: 220),
                  child: Icon(IconBroken.Location,color: Colors.green,),
                ), // Padding
              ],
            ), // Row
          ), // Padding
        ],
      ), // Stack
    ), // InkWell
  ); // Padding
}

static Future<void> openMap(latitude, longitude) async {
  String googleUrl = 'https://www.google.com/maps/search/?api=1&query=$latitude,$longitude';
  if (await canLaunch(googleUrl)) {
    await launch(googleUrl);
  } else {
    throw 'Could not open the map.';
  }
}
```

6. Testing plan

In this chapter, we will discuss types of testing used and test cases we examined through our application to detect bugs of the system and solve it

Functional testing:

Unit testing

Testing of individual items (e.g. modules, programs, objects, classes, etc.) as part of the coding phase, in isolation from other development items and the system as a whole

Integration testing

Testing the interfaces between major (e.g. systems level application modules) and minor (e.g. individual programs or components) items within the application

System testing

Testing a system behavior after it is fully integrated, as a whole when development is finished, hence, the system can be tested as complete entity

Regression testing

To check older functionality after integrating new functionality

Acceptance testing

Testing to evaluate the application compliance to ensure that it is ready to be deployed into the business, operational or production environment

Non-functional testing:

<u>Performance testing:</u>

Accomplish a designated function regarding processing time and throughput rate

<u>Load testing:</u>

Measuring the behavior of increasing load that can be handled by the system

<u>Stress testing:</u>

Evaluate a system or component at or beyond the limits of its specified requirements

<u>Security testing:</u>

Testing how well the system is protected against unauthorized internal or external access

Black-box testing techniques:

- Boundary value analysis - (Used)
- Equivalence partitioning - (Used)
- Decision table testing
- State transition testing
- Error guessing
- Graph based testing methods
- Comparison testing

White-box testing techniques:

- Statement coverage
- Branch coverage
- Condition coverage
- Multiple condition coverage
- Path coverage
- Function coverage

- User login test case:

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
01	Check user login with valid data	1. Open application 2. Click on sign in with google 3. Enter email 4. Enter password	Email = user@gmail.com Password = 123	User will be proceeded to the home page	As expected	Pass
02	Check user login with invalid data	1. Open application 2. Click on sign in with google 3. Enter invalid email 4. Enter password	Email = userr@gmail.com Password = 123	User cannot be proceeded to the home page	This email is not correct, please try again	Pass

- **Add post test case:**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
03	Check add post with valid data	1. Open app 2. Click on add post 3. Write post 4. Attach a photo 5. Click on add post	Post = 'hello all' Photo = 1.jpg	Post will be added and appear in the home page	As expected	Pass
04	Check add post without attaching a photo	1. Open app 2. Click on add post 3. Write post 4. Don't attach a photo 5. Click on add post	Post = 'hello all' Photo = null	Post will be added and appear in the home page	As expected	Pass
05	Check add post with an empty text field	1. Open app 2. Click on add post 3. Leave the text field of the post empty 4. Attach a photo 5. Click on add post	Post = ' ' Photo = 1.jpg	Post will be added and appear in the home page	As expected	Pass

06	Check add post with an empty text field and not attaching a photo	1. Open app 2. Click on add post 3. Leave the text field of the post empty 4. Don't attach a photo 5. Click on add post	Post = ' ' Photo = ' '	Post will not be added and an error message will appear to the user	This post cannot be uploaded	Pass

- **Search user test case:**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
07	Check search user with valid data	1. Open app 2. Click on search icon 3. Write a valid username 4. Click on search	Username = 'mostafaahmed5'	This user appears in the search page and I can go to his profile	As expected	Pass
08	Check search user with invalid data	1. Open app 2. Click on search icon 3. Write an invalid username 4. Click on search	Username = 'mostafaahmed20'	No user appear in the search page	There is no user with this username	Pass

- **<u>Send message test case:</u>**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
09	Check send message with valid data	1. Open app 2. Open profile of any user 3. Click on send message 4. Write a valid message 5. Click send	Message = 'hello'	This message will be sent and it will appear in the chat for the other user	As expected	Pass
10	Check send message with invalid data	1. Open app 2. Open profile of any user 3. Click on send message 4. Don't write anything 5. Click send	Message = ' '	This message will not be sent and nothing will appear in the chat for the other user	Message is not sent	Pass

- **Add product test case:**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
11	Check add product with valid data	1. Open app 2. Open shop 3. Click on add product button 4. Write valid product details 5. Attach product photo 6. Click on add	Product details = 'car motor' Photo = 1.jpg	This product will be added in the shop and will appear to all users	As expected	Pass
12	Check add product with invalid data	1. Open app 2. Open shop 3. Click on add product button 4. Write invalid product details 5. Attach product photo 6. Click on add	Product details = 'null' Photo = 1.jpg	This product will not be added in the shop and will not appear to all users	Product is not added in the shop	Pass

13	Check add product with invalid data	1. Open app 2. Open shop 3. Click on add product button 4. Write valid product details 5. Don't attach product photo 6. Click on add	Product details = 'car motor' Photo = null	This product will not be added in the shop and will not appear to all users	Product is not added in the shop	Pass
14	Check add product with invalid data	1. Open app 2. Open shop 3. Click on add product button 4. Write invalid product details 5. Don’t attach product photo 6. Click on add	Product details = 'null' Photo = null	No product will be added in the shop and no new product will appear to all users	Product is not added in the shop	Pass

- **Search product test case:**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
15	Check search product with valid data	1. Open app 2. Open shop 3. Click on search icon 4. Write a valid product name 5. Click on search	Search query = 'car motor'	All products with this name will appear in the search page and I can see each product details and chat with the seller	As expected	Pass
16	Check search product with invalid data	1. Open app 2. Open shop 3. Click on search icon 4. Write an invalid product name 5. Click on search	Search query = null	No products appear in the search page	There is no product with this name	Pass

- **Add problem test case:**

Test Case ID	Test scenario	Test Steps	Test data	Expected result	Actual results	Pass Or fail
17	Check add problem with valid data	6. Open app 7. Open problems section 8. Click on add problem 9. Write your problem 10. Provide your location 11. Click on add problem	Problem details = 'my car broke down suddenly' Location = current location	This problem is added and will appear to other users	As expected	Pass
18	Check add problem with invalid data	6. Open app 7. Open problems section 8. Click on add problem 9. Write your problem 10. Don't provide your location 11. Click on add problem	Problem details = 'my car broke down suddenly' Location = null	This problem is not added in the problems section	Problem cannot be added without location	Pass

7. Conclusion and results

In this chapter, we will give a brief summary of the study including the problems found and the proposed solution, also we will show the benefits of the project

In this project we tried to make the best we can to help people to easily communicate with each other and help other people if their cars broke down suddenly in the street by creating an environment which is very suitable for that purpose

Consider it as a way of expressing our duty towards our society as we felt responsible to participate in making the process of communication between people easier than before using the technology

In summary, our society would be more convenient if we made the process of communication between people easier and simpler than before, and surely using the technology and all the advances that happened in this field in the last years is very useful and it also reduce the cost of communication very much

We should continue to try to help people communicate and help each other in solving any kind of problems

We tried to do that with our limited fund and resources, and we are sure that we are going to expand this application and enhance its performance and fame if we get the right resources and fund

Finally, realizing the fact that "a few clicks" can do a lot to a society will definitely make us more aware of the technology role and become better versions of ourselves

GroupGram
social media

THANK YOU