T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4086 Mobile Device Programming

**Title of the Project**

# *Online Complaint Managements System*

**Group Members**

150116039 – Ahmet Faruk Çolak

150116016 – Berk Köylü

150116012 – Rahim Gün

## Contents

## 1. Project Description

This application is an online complaint management system. A user can login, then add a complaint with detailed information. He/she also can supply location information and a photograph of the complaint.

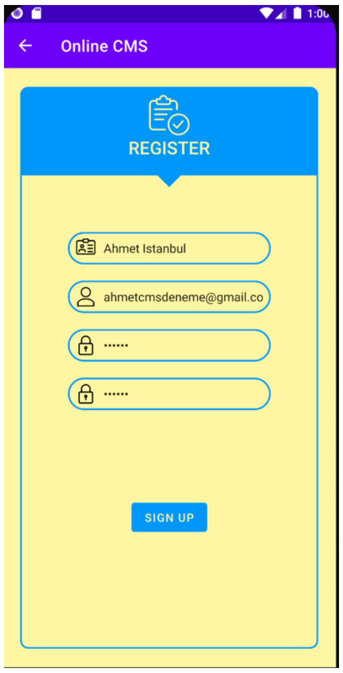
From the admin side, admin users can see all the complaints and their status. If a complaint was processed and done, he/she can change the status of the complaint.

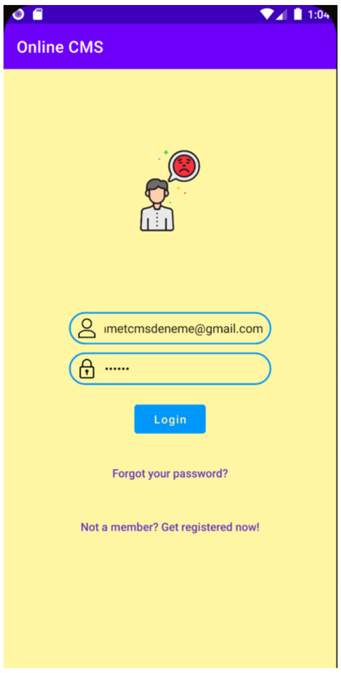
This is not a general application, but it can be used by local municipalities. For instance, İstanbul Municipality can use this for citizen’s complaints.

We implemented this project on Android Studio by using Java and XML.

## 2. Detailed Features

In this section, we are going to elaborate functionalities of the application and explain how to use the application properly. When we first open the application, the very first screen is a login activity. If the user is not a member, he/she should need to register to use the application beforehand. User can open the register screen by clicking the "Not a member?" button. Then, you can fill out the form to complete registration process. If the user writes any inappropriate input to the form, the application warns the user to check her/his input again for correction.





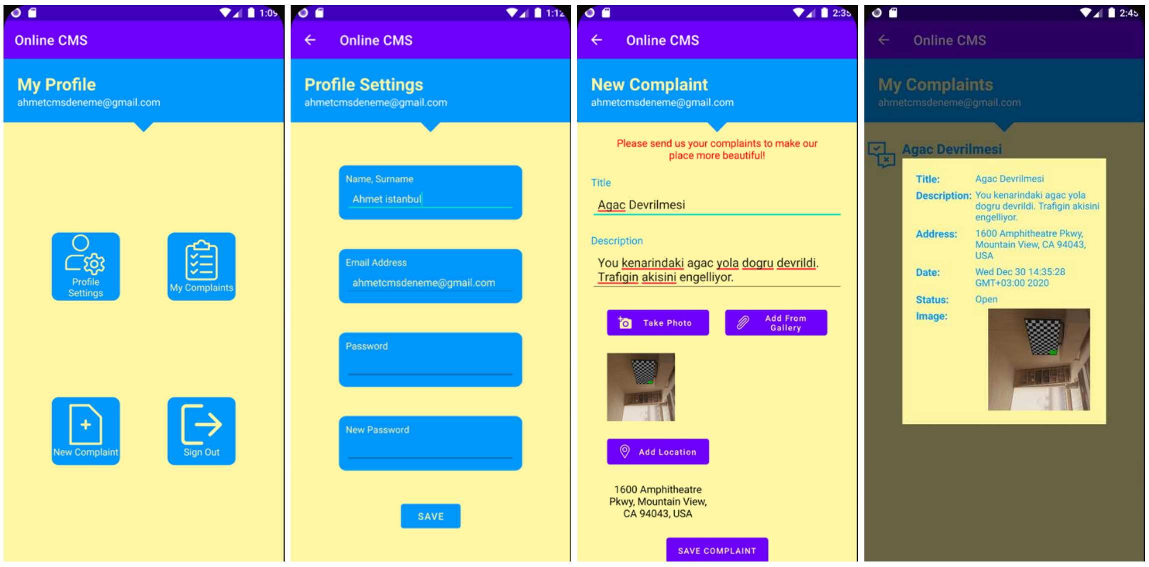
After registered, application redirects user to login screen. User can login to the application by writing email address and password. Also, users can change their password if they forget it from the “Forgot your password” section by writing their email address. And then, a password change request email will be sent.

User profile screen includes three buttons to forward use to profile settings, my complaints, new complaints and one button to log out the user from the application.

Inside the profile settings screen, users can change her/his name, email address and password easily by writing new ones then clicking the save button right below.

New complaint screen supplies fields so that the user can create his/her complaint by providing information about the title, description of the situation, photo of the situation, and location of the complaint.

My complaints screen shows the complaints of the user as a list and user can press one of the list elements to look at details of the complaint.



ahmetcmsdeneme@gmail.com

Name, Surname: Ahmet istanbul
Email Address: ahmetcmsdeneme@gmail.com
Password
New Password

Please send us your complaints to make our place more beautiful!
Title: Agac Devrilmesi
Description: You kenarindaki agac yola dogru devrildi. Trafigin akisini engelliyor.
1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA

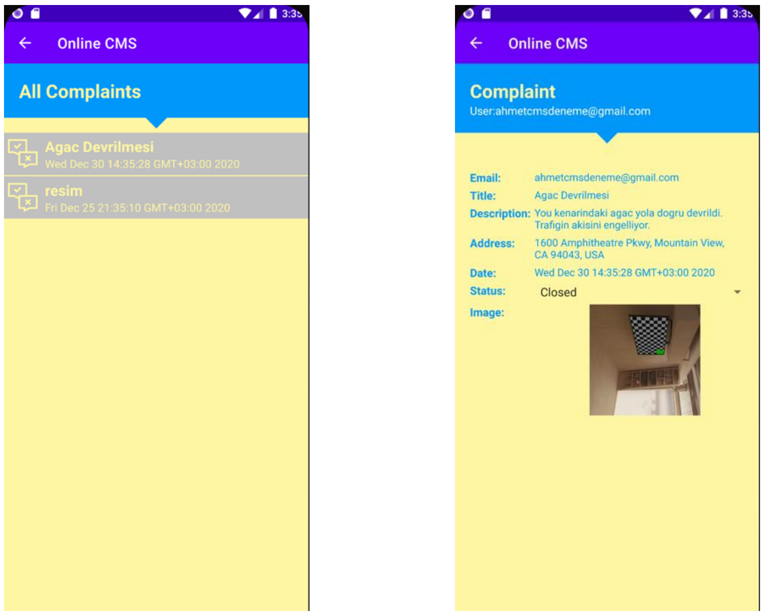
Agac Devrilmesi
Title: Agac Devrilmesi
Description: You kenarindaki agac yola dogru devrildi. Trafigin akisini engelliyor.
Address: 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
Date: Wed Dec 30 14:35:28 GMT+03:00 2020
Status: Open
Image:

*Figure 1: a) (leftmost) Profile Screen, b) (mid-left) Profile Setting Screen, c) (mid-right) New Complaint Screen, d) (rightmost) My Complaints Screen and Detailed Dialog*

Admins of the application can login the system using the login screen with their predefined mail address. Admins directly are forwarded to admin dashboard. Admin dashboard includes buttons to pass other screens that are profile settings button, all complaints button, add admin button and sign out button.

Profile settings screen shows the name of the admin, mail address of the admin, and password of the admin. Admin can change these fields by rewriting them and pressing the save button below.

All complaints screen shows the complaints of the users. Admins can see the complaints as a list and click each individual element to see the details of the complaint. If one complaint is resolved, the admin can change the status of the complaint as closed and confirmation mail of the complaint can be sent after that.


Online CMS

All Complaints

Agac Devrilmesi
Wed Dec 30 14:35:28 GMT+03:00 2020

resim
Fri Dec 25 21:35:10 GMT+03:00 2020

Online CMS

Complaint
User:ahmetcmsdeneme@gmail.com

| | |
|---|---|
| Email: | ahmetcmsdeneme@gmail.com |
| Title: | Agac Devrilmesi |
| Description: | You kenarindaki agac yola dogru devrildi. Trafigin akisini engelliyor. |
| Address: | 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA |
| Date: | Wed Dec 30 14:35:28 GMT+03:00 2020 |
| Status: | Closed |
| Image: | |

*Figure 2: a) (left) All Complaints Screen, silver background means complaint is closed. b) (right) Detail of the Complaint*

Admins can also register admins to the system at the Add Admin screen. They can write the name of the new admin, the defined email address of the new admin, and the password of the new admin. After supplying information, the admin can click the save button to finish the new admin registration process.

# 3. Technologies

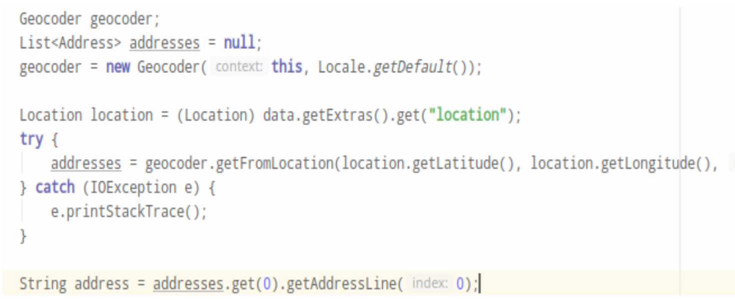
## 3.1 Main Technologies

### 3.1.1 Camera

We provide two different options to user for adding a photo about complaint. One of them is upload image from gallery and other is taking photo. We store images as Uri for both. We used "android.intent.ACTION_PICK" intent to add image gallery. This intent also asks user to select a media storage application to picking image. We used "MediaStore.ACTION_IMAGE_CAPTURE" intent for taking new photo. It launches camera application of the device. [1]

### 3.1.2 Maps / Location

We get user' location with GoogleMap fragment. When user click "Add Location" button, GoogleMap fragment is automatically created from gms.maps.SupportMapFragment. If user location is enabled, clicking the marker that is located right top of the screen shows current location. Then user click select button to confirm current location. This action saves information in a Location object. We used longitude and longitude attributes of this location object to getting address of current location with using Geocoder. This address contains city, state, postal code. [2]



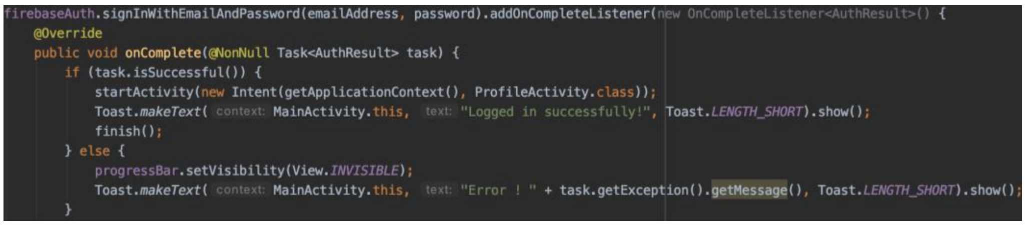
```
Geocoder geocoder;
List<Address> addresses = null;
geocoder = new Geocoder( context: this, Locale.getDefault());

Location location = (Location) data.getExtras().get("location");
try {
    addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(),  maxResults: 1);
} catch (IOException e) {
    e.printStackTrace();
}

String address = addresses.get(0).getAddressLine( index: 0);
```

### 3.1.3 Firebase (Authentication / Realtime Database / Storage)

App needs to identify users to allow an app to safely register the user data inside the server. Firebase Authentication provides backend services, easy-to-use SDKs to authenticate users to your app. It supports authentication using passwords, email addresses. Also, Firebase provides various types of authentication properties, but we used email and password-based authentication. Firebase Authentication SDK provides methods to create and manage users that use their email addresses and passwords to sign in. Firebase Authentication also handles sending password reset emails to users. [3]

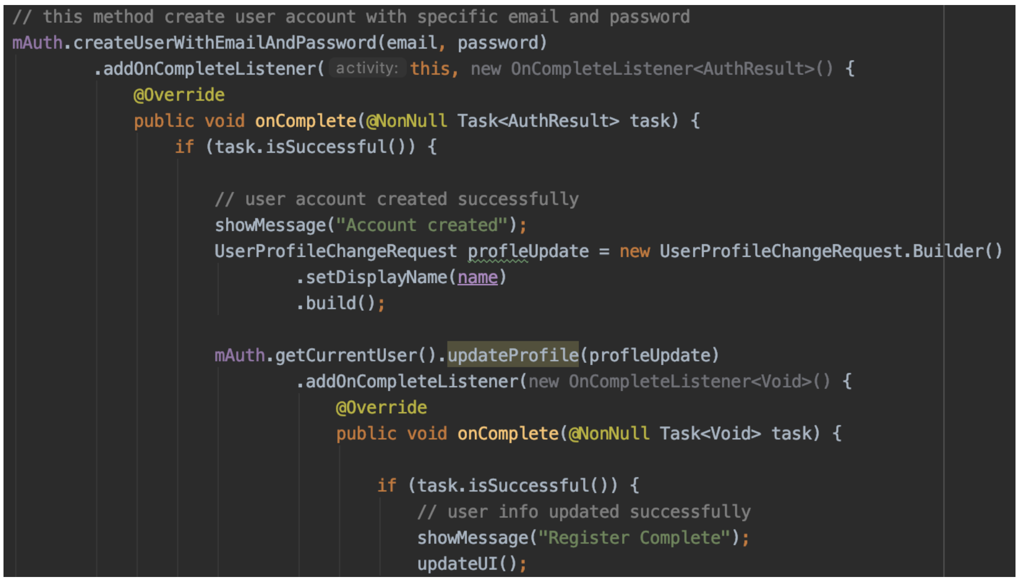
To sign a user into your app, you first need to get authentication credentials from the user. These credentials are the user's email address and password in our application. Then, you pass these credentials to the Firebase Authentication SDK. Our backend services will then verify those credentials and return a response to the client. After a successful sign in, you can access the user's basic profile information, and you can control the user's access to data stored in Firebase Storage. [4]

The code block below shows the user sign in process. “firebaseAuth” object contains the instance of FirebaseAuth class that gives us a facility that user can login the application using email address and password and if the login process is successfully completed, we can start the next activity.



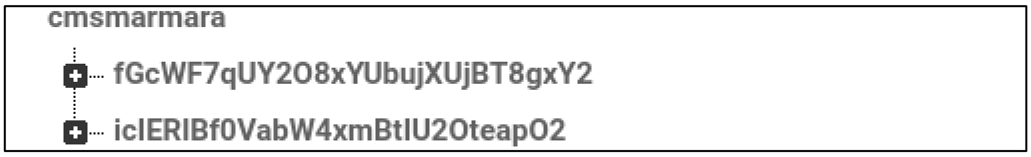
```
firebaseAuth.signInWithEmailAndPassword(emailAddress, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            startActivity(new Intent(getApplicationContext(), ProfileActivity.class));
            Toast.makeText( context: MainActivity.this,  text: "Logged in successfully!", Toast.LENGTH_SHORT).show();
            finish();
        } else {
            progressBar.setVisibility(View.INVISIBLE);
            Toast.makeText( context: MainActivity.this,  text: "Error ! " + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
```

Also, registering user process is similarly easy. Same as above, we get the instance of the FirebaseAuth class. And then, we can create the user account using "createUserWithEmailAndPassword" method and supplying password and email of the user.

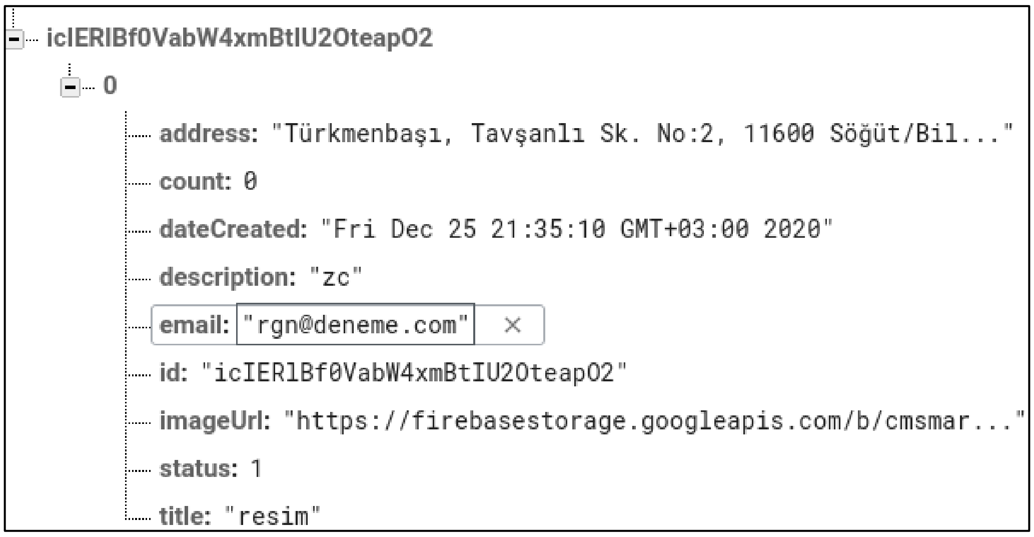


```
// this method create user account with specific email and password
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {

                    // user account created successfully
                    showMessage("Account created");
                    UserProfileChangeRequest profleUpdate = new UserProfileChangeRequest.Builder()
                            .setDisplayName(name)
                            .build();

                    mAuth.getCurrentUser().updateProfile(profleUpdate)
                            .addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {

                                    if (task.isSuccessful()) {
                                        // user info updated successfully
                                        showMessage("Register Complete");
                                        updateUI();
```

We used Firebase real-time database for saving complaints. We have a root node that storing user ID's which has submitted some complaints. In each ID, we have complaints that are owned by that user. Each complaint has title, description, address, date, status, URL of image and count field. We used a database reference to get root node of the database. After getting this reference, we expand a node with current user ID with complaint information.



```
cmsmarmara
+ fGcWF7qUY2O8xYUbujXUjBT8gxY2
+ icIERlBf0VabW4xmBtIU2OteapO2
```



```
- icIERlBf0VabW4xmBtIU2OteapO2
  - 0
      address: "Türkmenbaşı, Tavşanlı Sk. No:2, 11600 Söğüt/Bil..."
      count: 0
      dateCreated: "Fri Dec 25 21:35:10 GMT+03:00 2020"
      description: "zc"
      email: "rgn@deneme.com"
      id: "icIERlBf0VabW4xmBtIU2Oteap02"
      imageUrl: "https://firebasestorage.googleapis.com/b/cmsmar..."
      status: 1
      title: "resim"
```

We also used Firebase Storage for storing images. We named image file combination of id and count number. This file URL stored in complaint. When user want to display information about a complaint, Glide download image with this URL to imageView.

## 3.2 Other Technologies

### 3.2.1 RecyclerView / Adapter

RecyclerView makes it easy to efficiently display large sets of data. You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they are needed. [10]

We used these technologies to display complaints as a list in both the user and admin side of the application. We generated a xml design for the model of each item in the list named as “complaint_card_model” and it includes an ImageView, and two TextViews -title and the date of the complaint-.

### 3.2.2 Toast

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout. [8]

We used Toast messages too many times to inform the user about the operations and their results such as “Image Uploading…”, “Logged in Successfully” etc.

### 3.2.3 Dialog

A dialog is a small window that prompts the user to decide or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed. [9]

We used Dialogs and its subclasses several times.

For example, we use an AlertDialog when a user clicks the “Sign Out” button. It gives a warning and says, “Are you sure you want to sign out?”.

We also use a Dialog in the “My Complaints” screen. When a user clicks a complaint from the list (RecyclerView), a Dialog pops up, and displays the details of that complaint.

### 3.2.4 Spinner

Spinners provide a quick way to select one value from a set. In the default state, a spinner shows its currently selected value. Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one. [7]

We used Spinner in the detail of each “All Complaints” screen at the admin side. Every complaint has a status and admins can change it by the help of spinners. In the default state, it shows the current state of the complaint.

To use this spinner, we created an ArrayAdapter using the string array (status) and a default spinner layout and specified the layout to use when the list of choices appears and applied the adapter to the spinner.

### 3.2.5 ScrollView

A view group that allows the view hierarchy placed within it to be scrolled. Scroll view may have only one direct child placed within it. Scroll view supports vertical scrolling only. [6]

We used ScrollView in the “New Complaint” screen to have a vertically scrollable view since we have many components in that screen.

```
<ScrollView
    android:id="@+id/new_complaint_scroll_view"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginTop="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@id/new_complaint_upper_constraint">
```

To add multiple views within the scroll view, we added a ConstrainLayout, and placed additional views within that ConstrainLayout.

### 3.2.6 Glide

Glide is a fast and efficient open-source media management and image loading framework for Android that wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface. [5]

We used Glide for loading the images from the database and displaying it in the “My Complaints” and “All Complaints” screen.

```
if (!complaint.getImageUrl().equals("")){
    GlideApp.with(holder.cardLayout.getContext())
            .load(firebaseStorage.getReferenceFromUrl(complaint.getImageUrl()))
            .into(imageView);
    imageView.setVisibility(View.VISIBLE);
}
```

## 4. References

1. https://developer.android.com/training/camera/photobasics
2. https://developers.google.com/maps/documentation/android-sdk/start
3. https://firebase.google.com/docs/database/android/read-and-write
4. https://firebase.google.com/docs/storage/android/start
5. https://github.com/bumptech/glide
6. https://developer.android.com/reference/android/widget/ScrollView
7. https://developer.android.com/guide/topics/ui/controls/spinner
8. https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=tr
9. https://developer.android.com/guide/topics/ui/dialogs
10. https://developer.android.com/guide/topics/ui/layout/recyclerview