**package** com.multiPingPong.pong;  
  
**import** android.Manifest;  
**import** android.app.Activity;  
**import** android.content.ContentResolver;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.graphics.Bitmap;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.os.Environment;  
**import** android.provider.MediaStore;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.annotation.Nullable;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** androidx.core.app.ActivityCompat;  
**import** androidx.core.content.ContextCompat;  
**import** androidx.core.content.FileProvider;  
  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.webkit.MimeTypeMap;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.tasks.OnFailureListener;  
**import** com.google.android.gms.tasks.OnSuccessListener;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.storage.FirebaseStorage;  
**import** com.google.firebase.storage.StorageReference;  
**import** com.google.firebase.storage.UploadTask;  
**import** com.squareup.picasso.Picasso;  
  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
  
**public class** Profile **extends** AppCompatActivity {  
 ImageView **imageView**;*//it is made public so that any function can access it* Button **click**,**gal**;  
 EditText **name**;  
 TextView **email**;  
 Uri **imagepath**;  
 String **currentPhotoPath**;  
 **private** FirebaseAuth **firebaseAuth**;  
 **private** FirebaseDatabase **firebaseDatabase**;  
 **private** DatabaseReference **databaseReference**;  
 FirebaseStorage **firebaseStorage**;  
 **private** StorageReference **storageReference**;*/\*Since our app is connected to "Firebase Storage" but still we need to get the "Storage  
 Reference"\*/* **public static final int *CAMERA\_REQUEST\_CODE***=102;  
 **public static final int *CAMERA\_PERM\_CODE***=101;  
 **public static final int *GALLERY\_REQUEST\_CODE***=105;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 Intent intent=getIntent();  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***profile***);  
  
 **name**=(EditText) findViewById(R.id.***one***);  
 **email**=(TextView) findViewById(R.id.***two***);  
 **imageView** = (ImageView) findViewById(R.id.***three***);  
 Button click = (Button) findViewById(R.id.***four***);  
 Button gal = (Button) findViewById(R.id.***five***);  
  
 **firebaseAuth** = FirebaseAuth.*getInstance*();  
 **firebaseStorage** = FirebaseStorage.*getInstance*();  
 **storageReference** = **firebaseStorage**.getReference();  
  
 *// storageReference= FirebaseStorage.getInstance().getReference();* click.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 askCameraPermissions();  
 }  
 });  
 gal.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent gallery=**new** Intent(Intent.***ACTION\_PICK***,MediaStore.Images.Media.***EXTERNAL\_CONTENT\_URI***);*/\*Here we have defined an  
Intent to perform action\_pick which means picking up images using mediastore class from the URI specified\*/* startActivityForResult(gallery,***GALLERY\_REQUEST\_CODE***);  
 }  
 });  
 }  
  
 **public void** askCameraPermissions()  
 {  
 **if**(ContextCompat.*checkSelfPermission*(**this**, Manifest.permission.***CAMERA***)!= PackageManager.***PERMISSION\_GRANTED***)*/\*  
 checkSelfPermission() is used to check whether permission has already been granted or not if not then we are requesting the  
 permissions using "requestPermissions()" \*/* {  
 ActivityCompat.*requestPermissions*(**this**,**new** String[]{Manifest.permission.***CAMERA***},***CAMERA\_PERM\_CODE***);  
 }  
 **else** dispatchTakePictureIntent();  
 }  
  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {*/\*  
When the user responds to your app's permission request, the system invokes your app's onRequestPermissionsResult() method, passing  
 it the user response. Your app has to override that method to find out whether the permission was granted. The callback is passed  
 the same request code you passed to requestPermissions(). \*/* **if**(requestCode==***CAMERA\_PERM\_CODE***)  
 **if**(grantResults.**length**>0 && grantResults[0] == PackageManager.***PERMISSION\_GRANTED***)  
 {  
 dispatchTakePictureIntent();  
 }  
 **else** {  
 Toast.*makeText*(**this**,**"Camera Permission is required to use camera"**,Toast.***LENGTH\_LONG***).show();  
 }  
 }  
 */\*So we will start the in-built camera of our app by creating the camera Intent\*/  
 /\* public void openCamera()  
 {  
 Intent camera= new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 startActivityForResult(camera,CAMERA\_REQUEST\_CODE);/\*Starting another activity, whether one within your app or from another  
 app, doesn't need to be a one-way operation. You can also start another activity and receive a result back. For example, your app  
 can start a camera app and receive the captured photo as a result. Or, you might start the Contacts app in order for the user to  
select a contact and you'll receive the contact details as a result.And you retrieve the result by overriding the "onActivityResult()"\*/  
 /\* }\*/* @Override  
 */\*The Android Camera application encodes the photo in the return Intent delivered to onActivityResult() as a small Bitmap in the  
extras, under the key "data". The following code retrieves this image and displays it in an ImageView.\*/* **protected void** onActivityResult(**int** requestCode, **int** resultCode, @Nullable Intent data) {  
 */\*Here "data" is the image data which we have recieved and we are trying to retireve by overriding this function\*/* **if** (requestCode == ***CAMERA\_REQUEST\_CODE***) {*/\*The startActivityForResult method takes an intent and a request code. In the example, the intent points explicitly to the  
activity being started. The request code is any int value. The request code identifies the return result when the result arrives.  
(You can call startActivityForResult more than once before you get any results. When results arrive, you use the request code to  
 distinguish one result from another.)\*//\*The call to setResult sends a result code (RESULT\_OK, RESULT\_CANCELED, or any positive  
 int value that’s meaningful to the receiver) along with the intent full of useful information.\*/  
/\*In short the meaning of above explanation is "request\_code is the calling function identity, from where it is requested,  
result\_code is the called function identifier, also it specifies status of the called message as Intent.ACTIVITY\_OK and so on."\*/  
 /\*If u still don't understand what is said above refer to <https://www.dummies.com/web-design-development/mobile-apps/how-to-get-results-back-from-an-activity-in-your-android-app/>\*/* **if** (resultCode == Activity.***RESULT\_OK*** && data.getData() != **null**) {  
   
 }  
 */\*Bundle bundle=data.getExtras();/\*"data" is the object of "intent" and "getExtras" return type is "Bundle" which and we  
 are using it to recieve data \*/  
 /\*Bitmap bitmap=(Bitmap) bundle.get("data");/\*so here we are trying to get the "bitmap" format of the image. Therefore  
 typecasting it\*/  
 /\* imageView.setImageBitmap(bitmap);//now just setting the image recieved on to the "imageview" of the "mainAcivity"\*/* }  
  
  
 **if** (requestCode == ***GALLERY\_REQUEST\_CODE***) {  
 **if** (resultCode == Activity.***RESULT\_OK***) {  
 Uri contentUri = data.getData();  
 *//we are creating contentUrl from the "data"  
 /\* Intent intent = new Intent();  
 intent.setType("image/\*");  
 intent.setAction(intent.ACTION\_GET\_CONTENT);  
 startActivityForResult(Intent.createChooser(intent, "Select an Image"), PICK\_Image);\*  
  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 if (requestCode == PICK\_Image && resultCode == RESULT\_OK && data.getData() != null) {  
 imagepath = data.getData();  
 try {  
 Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), imagepath);  
 imageView.setImageBitmap(bitmap);  
 senduserimage();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 super.onActivityResult(requestCode, resultCode, data);  
 }/  
/\*In the next 2 lines we are creating FILE , we are again creating this file because now here this file is going to be inside  
the "gallery" so there has to be a new file earlier when we created the file that was inside the "Internal Storage of Mobile"\*/* String timeStamp = **new** SimpleDateFormat(**"yyyyMMdd\_HHmmss"**).format(**new** Date());  
 String imageFileName = **"JPEG\_"** + timeStamp + **"."** + getFileExt(contentUri);*/\*here we are also specify the type of  
file whether it is .JPEG or .png or any other . So for that we are using getFileExt(contentUri) and this method is written just  
below this function \*/* Log.*d*(**"tag"**, **"OnActivityResult:Gallery Image URI"** + imageFileName);  
  
 *//imageView.setImageURI(contentUri);* updateImageToFirebase(imageFileName,contentUri);  
  
 }  
 }  
 **super**.onActivityResult(requestCode, resultCode, data);  
 }  
 **private void** updateImageToFirebase(String name,Uri contentUri)  
 {  
 **final** StorageReference image = **storageReference**.child(**"pictures/"** + name);*/\*It will create a new directory in the firebase called "pictures" and inside it the image will be saved\*/  
/\*So in order to upload the images into the firebase we need to use the same reference "image" created above and pass the "contentUri" of that file and so we also have added  
 a listener named "OnSuccessListener" which would be called only when the image is uploaded successfully in the firebase else "OnFailureListener()" will be called\*/* image.putFile(contentUri).addOnSuccessListener(**new** OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
*/\*now we want the image uri in case of successful uploading of the image therefore we have used the same "storageReference" image for that and by use of getDownloadUrl()  
we are going to get the uri and after that we have also added the OnSuccessListener() on it\*/* image.getDownloadUrl().addOnSuccessListener(**new** OnSuccessListener<Uri>() {  
 @Override  
 **public void** onSuccess(Uri uri) {  
 Log.*d*(**"tag"**, **"onSuccess: Uploaded Image URl is "** + uri.toString());  
 Picasso.*get*().load(uri).into(**imageView**);*/\*This will load the image into imageView Whose variable name is  
 imageView. \*/* }  
  
 });  
 Toast.*makeText*(Profile.**this**, **"Image Is Uploaded."**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }).addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception e) {  
 Toast.*makeText*(Profile.**this**, **"Upload Failled."**, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
 **private** String getFileExt(Uri contentUri)  
 {*/\*Content Resolver resolves a URI to a specific Content provider.Content Provider provides an interface to query content.For  
more info :<androiddesignpatterns.com/2012/06/content-resolvers-and-content-providers.html>\*/* ContentResolver c= getContentResolver();  
 MimeTypeMap mimeTypeMap=MimeTypeMap.*getSingleton*();*/\*By use of "getSingleton()" of "MimeTypeMap" we are getting the extens-  
ion of that image file\*/* **return** mimeTypeMap.getExtensionFromMimeType(c.getType(contentUri));  
 }  
  
 **private** File createImageFile() **throws** IOException {*/\*here "createImageFile()" can throw IOException hence below also we have  
included it inside try catch block\*/  
 // Create an image file name* String timeStamp = **new** SimpleDateFormat(**"yyyyMMdd\_HHmmss"**).format(**new** Date());  
 String imageFileName = **"JPEG\_"** + timeStamp + **"\_"**;  
 */\*The below line is commented because getExternalFilesDir() will make the images private to our application and we wont be allowed  
 to display the image in Gallery.Therefore we made use of "Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES)".  
 But, if you don't want to store images in gallery then you can use getExternalFilesDir() and no need to use  
 "Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES)"\*/  
  
 /\*File storageDir = getExternalFilesDir(Environment.DIRECTORY\_PICTURES);/\*"getExternalFilesDir()" helps u to get absolute path  
of directory in the external storage where your image is to be stored\*/* File storageDir = Environment.*getExternalStoragePublicDirectory*(Environment.*DIRECTORY\_PICTURES*);*/\*So inside  
Environment.DIRECTORY\_PICTURES,I am going to save my images\*/* File image = File.*createTempFile*(  
 imageFileName, */\* prefix \*/* **".jpg"**, */\* suffix \*/* storageDir */\* directory \*/* );  
  
 *// Save a file: path for use with ACTION\_VIEW intents* **currentPhotoPath** = image.getAbsolutePath();  
 **return** image;*/\*This image can be displayed in the ImageView or uploaded in the server\*/* }  
  
*/\*The following function will open the camera and start taking picture \*/* **private void** dispatchTakePictureIntent() {  
 */\*So we will start the in-built camera of our app by creating the camera Intent\*/* Intent takePictureIntent = **new** Intent(MediaStore.***ACTION\_IMAGE\_CAPTURE***);  
 *// Ensure that there's a camera activity to handle the intent* **if** (takePictureIntent.resolveActivity(getPackageManager()) != **null**) {*/\*the statement inside if will check whether camera  
 is present in the device or not and if the device is going to give us the camera resources or not\*/  
 // Create the File where the photo should go* File photoFile = **null**;  
 **try** {  
 photoFile = createImageFile();  
 } **catch** (IOException ex) {  
 *// Error occurred while creating the File* }  
 *// Continue only if the File was successfully created  
/\*--------------------------------What is a Content Provider?-------------------------------------------------\*/  
/\*Ans:Content providers can help an application manage access to data stored by itself, stored by other apps, and provide a way to  
share data with other apps. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the  
standard interface that connects data in one process with code running in another process. Implementing a content provider has many  
advantages. Most importantly you can configure a content provider to allow other applications to securely access and modify your app  
 data as illustrated in figure 1.<https://developer.android.com/guide/topics/providers/content-providers.html>\*/* **if** (photoFile != **null**) {  
 Uri photoURI = FileProvider.*getUriForFile*(**this**,  
 **"com.multiPingPong.pong.android.fileprovider"**,  
 photoFile);*/\*default functionality of FileProvider includes content URI generation for files, you don't need  
to define a subclass in code.You need to generate url because To share a file with another app using a content URI, your app has to  
 generate the content URI.content URI is the file location where our image is saved\*/* takePictureIntent.putExtra(MediaStore.***EXTRA\_OUTPUT***, photoURI);*/\*The name of the Intent-extra used to indicate a  
 content resolver Uri to be used to store the requested image or video.\*/* startActivityForResult(takePictureIntent,***CAMERA\_REQUEST\_CODE***);*/\*Starting another activity, whether one within your  
app or from another app, doesn't need to be a one-way operation. You can also start another activity and receive a result back. For  
 example, your app can start a camera app and receive the captured photo as a result. Or, you might start the Contacts app in order  
 for the user to select a contact and you'll receive the contact details as a result.And you retrieve the result by overriding  
 the "onActivityResult()"\*/* }  
 }  
 }