TECHNI SOCIAL MEDIA FOR ARTIST

Student Statement:
I, Zakaria Benfeddoul, assert that I have applied ethics to the design process and in the selection of the final proposed design. I also confirm that I have held the safety of the public to be paramount and has addressed this in the presented design wherever may be applicable.

______________________Zakaria Benfeddoul______________________

Approved by the Supervisor

______________________Dr. Nasser Assem______________________
ACKNOWLEDGEMENTS

I would like to take this opportunity to thank Dr. Assem Nasser for accepting to be my supervisor and also for all the time he spent helping me overcoming all the difficulties and obstacles encountered while working on my capstone project. All of his guidelines and advises where very useful, it also helped me improve the quality of my work, be more inspired and have more ideas to develop.

More importantly, I would like to thank my parents for all the sacrifices they made for me. I am blessed to have parents that do everything they can for the sake of their children. I am very grateful for all the financial and moral support throughout all this years. All of this means everything, without our parents we are nothing, because as we all know, nobody wants you to be better than him expect your parents.

I would also like to thank my fiancé Lamyae, my sister Hajar, my friend Hamza, my cousin Ali, who supported be during difficult times and answered present when I needed them.

Let us not forget to thank Al Akhawayn University for all the knowledge and opportunities offered during my curriculum, the hard working staff of all Schools combined, and all those working hard to offer the most suitable and convenient living environment for us student here in AUI.
Figures

Figure 1 Rapid Application Model .................................................. 15
Figure 2 Firebase Architecture .................................................... 16
Figure 3 Firebase Data Synchronization ......................................... 17
Figure 4 Use case diagram .............................................................. 19
Figure 5: Activity diagram .............................................................. 21
Figure 6 ERD ................................................................. 22
Figure 7 Mobile application Login activity ....................................... 24
Figure 8 Mobile application Register activity .................................... 25
Figure 9 Mobile application Setup activity ....................................... 25
Figure 10 Mobile application Crop feature ........................................ 26
Figure 11 Mobile application Main activity ....................................... 27
Figure 12 Add post layout .............................................................. 28
Figure 13 Edit post layout .............................................................. 28
Figure 14 Settings layout .............................................................. 29
Figure 15 Profile layout ............................................................... 30
Figure 16 Online user layout .......................................................... 30
Figure 17 Message layout ............................................................. 31
Figure 18: Find friends layout ......................................................... 32
I. ABSTRACT

Being someone who spends most of his free time drawing, I always dreamed about a way to regroup all artists and be able to share this common passion with others. This idea of having a software that will bring this concept to life came to my mind while thinking about a project for my Capstone, and what is best to combine the useful with the pleasant. Knowing that almost everybody spend most of his time in his phone, I thought that having it as a mobile application will be the best and most practical way of use.

As I already implemented a mobile application as a Software engineering class project using Android Studio, I decided to follow the same strategy and use all the knowledge acquired in addition to Dr. Nasser Assem’s help to develop a good mobile application.

The idea of “Techi” started with having Social Media mobile application that will offer the ability to share and regroup different artists. In addition of that, this mobile application need to be converted to a web application to be of more use.
II. INTRODUCTION

This project is related to the Capstone Design course (EGR 4402) that is required to be taken by all student during their last regular semester of enrollment. It is meant to let the student make use of all the knowledge and information gathered in all courses and demonstrate his mastery of different concepts related to his major. Concerning this project, it will show the uses of different concepts and skills related to Computer Science. It covers the development of a Mobile Application, a Web Application and the use of Software Engineering.

The project is about creating a social media application (both web and mobile application) that will have for objective regrouping different kind of artists that will share different art pieces of many kind.

The first thing to do was to identify the community that I wanted to target. This is something that is necessary before starting the development of any software. The need to understand what might be interesting for this community is mandatory in order to define the features and functions.

The community wants and needs cannot be defined if we do not have enough knowledge about the domain related to them. As someone who spends most of his time drawing and sharing his art with others of the same interest, I find it easy to see what might please this targeted community.

The following thing to do is to choose the right technology that will suit the needs. As this is a software that I am developing by myself, friendly and quick use software are needed. Activity stream is the core of every social media software, the ability to interact between artists is one the main features that should be included in this application. Status updated will also be included in order to have a continuity in the flow and use of the software. Ratings and the ability to create a competition between users is also mandatory. It will offer the possibility to look for the desired art depending on the type or the name of the artist or any key word related to what the user is looking for.
III. STEEPELE ANALYSIS

Technology:
The need to keep track of the technology changes is mandatory, a research should be made to see if we are using the latest technology available in the market related to our subject to produce the best outcome possible with all new possible features. It is important for any company to consider this point seriously as it concerns the improvement of customers’ needs and experience. By tracking those changes, new projects and opportunities can see the day. This way Enovation can be maintained as it is one the main characteristics a company should have.

Economic:
It includes looking at the changes in the economic situation. This mobile application is a social media in principal, we can conclude that profits can be made from publicities, advertising and other utilities like sponsored accounts. This means that we need to track changes concerning the economy related Internet advertising.

Environmental:
This is something to take into consideration if our project will have an impact on the environment. We may use, in the future this exact application to be able to enhance digital art and make everything (interactions, exchange of art…) be virtual in order to save Mother Nature by using less paper and concreate material. The majority of the work done using physical material can nowadays be done using electronical devices instead. Political:
Government policies are things to take seriously when making a project in our case arts that might affect our application due to some restrictions should be banned. We can add a feature that will allow the user to report any content then check if this report is valid or not. Since we are accessing all data that is shared and can make all possible modification, it is in our obligation to make sure that the data is clean in order to maintain the good use of this mobile application. For it being a social media, it is easy to found ourselves in trouble as information in those kind of software spread very quickly, that is why we need to demonstrate that we are concerned when it comes to shared posts that goes against political laws.
**Ethical:**
Ethical factors are set of social principles that might affect how firms behave and we do need to be behave ethically. We shall respect the user personal information that are to be private. Accounts need to be protected and user usage only. If the user decides to Sponsor and advertise his content, offers need to be respected and only true information shall be stated. The user will have the possibility to see what information other users can check.

**Social:**
We need to examine social and cultural changes to make sure that the content of our software follows the flow in order to suite all users. Now days, our society life style did change a lot. The majority, not to say everybody, is using a smartphone and we need to follow those changes to suite best the user’s needs. The way this mobile application is developed goes with idea that many changes will be done in the future and that those changes need to respect the user’s expectations.
IV. REQUIREMENTS SPECIFICATIONS:

3.1 Functional Requirements:

Introduction:
This Application is a Social media project focused mainly on the mobile application format. It will allow users to share their art and talk about their passion. The application makes it easier for people to find with the same interest.
In this section, we will state the mobile application Functional Requirements.

Create new account:
The user shall be able to create an account before being able to use the application. After choosing to create a new account by clicking on the button “Create a new account” available in the Login activity.
The user will have to provide his E-mail, his password and will need to confirm his password. The user will then provide his username, his full name, his country and add a profile picture for his profile.

Login:
If the user already has an account, he shall be able to provide his email and password and then login.
In the login activity, the user can choose to use his Facebook, Twitter or Google account in order to Login.

Home Page:
When a user Login successfully, the application must send the user to the home page where all uploaded posts are shared.
The home page shall provide a button that allows the user to choose a picture of his art and share it adding to it a description and another button that shows the main layout.
**Main layout:**
The application shall provide a screen with many features to the user. The user shall be able to choose between:
- Add a new post.
- Access to his profile.
- Access to home page.
- Get friends list.
- Find friends.
- Check messages.
- Access settings.
- Logout.

**Add new post:**
The user shall be able to choose a picture from his gallery, add a description to it and upload it.

**Access to Profile:**
The user shall be able to access his profile and modify his information.

**Access to Home Page:**
If the user is on another screen, he shall be able to go back to the home page.

**Get Friends List:**
The user shall be able to display his friend list.

**Messages:**
The user shall be able to access his message box and communicate with other users.

**Logout:**
The user shall be able to logout from his account.
About:
The application shall provide a screen with the company’s information.

3.2 Non-Functional Requirements:

Accessibility requirement:
The system shall be accessible by all people with smartphones and have interest in art.
- An estimated amount of 5% of people with drawing art interest.

Availability requirement:
The system shall always be available unless there a maintenance.
The system shall notify the user of a potential maintenance in advance.

Confidentiality requirement:
The system shall protect all user’s information. User’s information can only be shared with other friend users.

Efficiency requirement:
The system shall be able to Login the user in no more than 5 seconds.
The system shall be able to upload an image in less than 10 seconds.

Integrity requirement:
The system shall provide uploaded images with the same quality.
The system shall update the profile picture of the user according to the exact output from the cropping function.

Reliability requirement:
The system shall rollback all updates if a failure occurs when changing the user information.
Scalability requirement:
The system shall be able to handle the big amount of users that will be increasing throughout the years. The firebase database will allow storing high amount of data.

Usability requirement:
The final product shall be easy to use for all ages. The system language will be in English in order to target a large number of users.
The system functionalities shall be easy to recognize.

Performance requirement:
The system performance shall be optimal. The execution speed shall be high and with a low response time. Android Studio will guarantee all this requirements.

Maintainability requirement:
The system shall be maintained in order to overcome system failures.
The system shall not exceed more than one maintenance in 24 hours.

Extensibility requirement:
The system shall allow the addition of new features in the future.

V. FEASIBILITY STUDY:

Feasibility study is an important step in all projects whatever the domain. It is the main thing to do before starting implementing any project. It allows us to see if the project is feasible to start working on it or not. With we can check for example the budget, do some research to see if it is going to be profitable or not and decide on the modification that can lead to a better result, in theory, before investing in anything. This way we can predict and avoid possible losses and irreversible mistakes or constraints that, in some cases, cannot be fix and may oblige the person working on the project to redo all the work (this means huge losses in the
Product and Service Feasibility:
The marketplace of social media is one of the thriving domains nowadays. As the majority of people, not to say 90% of earth’s population, use smartphones. To combine both, having a mobile application of a social media regrouping users with the same interest is this project main goal.

Technical Feasibility:
The development if this application is technically possible using free open software. We will be using Android studio (featuring JAVA and XML) to develop our mobile application. As it is a social media application, we need to store the user information and all the data. For this we will be using Firebase database.

Marketing Strategy Feasibility:
This application marketing strategy revolves around web advertising. As it is a social media application that regroups artists, it will mainly be about advertisements related to art (for example art supplies and new art tools). It is important to note that the community shall be well targeted.

The mobile application main goal, for it being a social media, should be to suit best the user. After some research, I was able to identify what users (people interest in art) would like to have in a social media dedicated to art. I decided then to implement this project for it being feasible and practical. This conclude that I can start giving life to the project for all it will bring to the targeted society.
VI. METHODOLOGY

As we are talking about a mobile application that needs to be developed in a short period of time, my choice was to go with the Rapid Application Development Model, the agile software development methodology that will suit best my work. This methodology goes mainly with the fact that you need to regularly get feedback and try to fix what needs to be fixed with the addition of a lot of testing. It is very useful in my case because I do need to grasp in every detail what will be best for my application in order to satisfy the user as much as possible. For it being a social media, I did ask some of my friends to test the application each time I added a new feature in order to stay updated.

The methodology was very helpful in order to meet all diaries deadlines as we need each time to update and add new features. We can say that this Rapid Application Development is more of a less talk more action methodology.

The main advantages of this RAD methodology are as follows:

- “We can break a project into small, more achievable pieces.”[1]
- “We get a working product more quickly”[1]
- “We get direct user feedback constantly”[1]

![Rapid Application Model](image)

*Figure 1: Rapid Application Model*
VII. SOFTWARE ARCHITECTURE

For this section, I will be presenting the system architecture that I worked with. The Three-tier architecture is the one used in this project. It represents the most used multilayered architecture. The three layers presented in the Three-tier architecture are as follows:

- "Presentation tier: This first tier represents the top level. It has all the information about all services offered in the application.
- Application tier: Can be called the logic tier, it represents the part that controls the application functionalities.
- Data tier: Represents the part where the information is stored (database servers)" [2] With this architecture, we can respect our non-functional requirements in a more practical way. It is easier to maintain the system with three independent tiers. For tiers to be independent makes it easier for us to make change in order to meet the user new requirements that can be added or modified easily.

In our case, the presentation tier contains the Mobile Application where the user connects and use the application features. The Application (or Logic) tier is all information added by the users that is stored in the Firebase that can be modified or delete depending on the needs. The data tier in here is the Firebase database.

Java servlets are what work as listeners with Firebase. Those servlets are running in the Application Engine flexible environment.

![Firebase Architecture](image.png)

*Figure 2: Firebase Architecture*
Transactions are used in order to ensure one servlet for each user event.
The communication between the application and the servlet stages are as follows:

- When a new user logs in, the app requests a logging servlet for that user by adding an entry under `/requestLogger/` in the Firebase Realtime Database.
- One of the servlets accepts the assignment by updating value of the entry to its servlet identifier. The servlet uses a Firebase transaction to guarantee that it is the only servlet can update the value. Once the value is updated, all other servlets ignore the request.

When the user logs in, logs out, or changes to a new channel, it logs the action in `/inbox/[SERVLET_ID]/[USER_ID]/`, where `[SERVLET_ID]` is the identifier of the servlet instance and `[USER_ID]` is a hash value representing the user.

The servlet watches the inbox for new entries and collects the log data.
Data is synchronized automatically across devices by the Firebase Real time Database each time the application stores in it. [4]

*Figure3: Firebase Data Synchronization*
VIII. DESIGN

7.1 Use Case Diagram

Figure 4: Use case diagram
This use case diagram shows the user interaction with the mobile application and all features offered to him. Depending on the user, if it is a new one he needs first to create an account and provide personal information. The actors here are the user and the system. When the user already has an account, he can login and choose to either modify his personal information, add or modify a post, like existing posts, look for artists or add new friends. On the other hand the system need to propose new artists and check the modified information before updating it.

7.2 Activity Diagram
The activity diagram in here allows us to state all possible action that a user can make using the application and their alternatives. At the start of the application, the user has two choices, either to login or to create a new account. If creating an account, as we can see in the figure the system should check the viability of the email address, the password and the confirm password before accepting those information entered by the user. After that, the system should save those information and check if the user chose a profile picture, a valid username, a full name and a country. Those information should also be saved after validation. The user is then in the main layout where he can add new posts, look for friends, access profile, or logout. Adding new post implies adding an image with its description.
Figure 5: Activity diagram
7.3 Entity Relationship Diagram

Each user can have many posts. Each post was uploaded only by one user.

Each message use has many users interacting with each other and many text messages.

Each post has only one post image, and each user has only one profile image.

Figure 6: ERD
IX. IMPLEMENTATION

8.1 Login Activity:
This following implementation represents the Login Activity. First layout encountered by the user in the application. Presented below, we can see the variables as UserEmail and UserPassword set as EditText for it being the information entered by the user if he already has an account, the LoginButton is set on click listener as it is selected in order to for the user to eventually login and access the mobile application main activity, this is done using the method SendUsertoMainActivity. If the user does not have an account, he is needs to be sent to the register activity, this is why we use the NeednewAccount link and the method SendUsertoRegisterActivity.

The following code does handle the cases where the user enter false email and password in order to login. This is done using Firebase feature. The information entered in the register activity is stored in the Firebase database, this way we can verify the validity of the information entered by the user. The Loading bar is included to inform the user about what is the system currently working on, if the information entered is wrong, an error message appear to the user. The user can choose to connect using the google authentication for example making it easier for him to access directly this mobile application features. This case too takes into account valid information.
8.2 Register and Setup Activity:
The user can choose between many options, he can either choose to create an account by providing his email and a password that needs to be confirmed (the password needs to be more than 6 digits), or he can choose the option to use his Facebook, Twitter, or Google account to register easily.
When the user account is created successfully the user is send to the setup activity where he is asked to choose his profile picture, enter a user name, a full name, and his country.

As mentioned before, the information entered by the user (Account and profile information) are both saved in the firebase database where they can be accessed. The code does handle the cases where the user either forgets to fill one of the columns. A message pops up in case he needs to provide his email, password or confirm his password. If the password and the confirm password do not much, the user is informed about it and is asked to reconfirm it. Same thing goes for the setup activity.
The Crop method allows the user to crop the picture he choose for his profile image

Figure 10: Mobile application Crop feature

After this, the method SendUserToMainActivity above will send the user to the main activity.
8.3 Main Activity:

In the main activity, the user is offered many features. He can add new posts that are shared between all users. When willing to do that, the user can click on the button Add New Post either from the button offered in the main menu bar or the in the navigation header. The user is able to edit or delete his posts. He can also choose to access the account settings to modify what can be modified. He can also access the user profile or look for friends (those options are available in the navigation header).
When it comes to editing a post, the only user able to do that should be the one who uploaded this exact post, this is why those two buttons are set to visible only if the ID of the current user matches the database ID related to the selected picture.

The user can choose settings option in order to modify his information.
In here, the user can modify all his account information. He can modify his profile picture, his user name, his full name, the country, the gender, and the type of art he is specialized in. The information is then updated and saved in the Firebase database.

In addition to that, the user can choose to view his profile info by choosing the option profile and have an idea what information is displayed to other users. The next figure represent the profile layout.
When selecting find friends the user can see the users that are online, this way he can have an idea on who is active at that moment or not. By clicking on the user that shows online, the user can see the exact user location using google maps service associated with the firebase tool. Users also have the ability to communicate between themselves, using messages. As this is a social media mobile application this is a main feature for this application.
Figure 17: Message layout
In this layout, the user can look for users using the search bar.

Figure 18: Find friends layout
X. TECHNOLOGY USED

For the development of the mobile application:
Concerning all what is related to the mobile application and the working of its functionalities I used:
- Android Studio: It was used for the development of this mobile application.
- Firebase: Was used for the data handling when it comes to storing data (user information), exchange of messages, or checking online users.

Other Tools:
- Creately.com was used in order to draw the diagrams.
- Photoshop was used for all the graphical needs, used mainly to design the user interface. It was also used to design the mobile application logo derived from its name Techni.
XI. Conclusion

This social media mobile application can still be improved. Features that we can think about are endless. Of course, things cannot be added normally, we need first to make sure that they respect and answer user’s needs. In order for this mobile application to come with benefits, real marketing strategy need to be applied in order to include publicities. Those publicities are going to represent our mane income and the more users we will have the more beneficial will it be.

In addition to that, another feature that can be interesting is the ability for the user to sell his art if another user is interested. For this, we will need to add a cart feature where the user will be able to add all the interesting arts that are shared. When the user is done, he will eventually apply his order and messages will be sent to the related users which posts where chosen. Those users will then inform the other user about the availability and when the transaction is accepted, a feature dealing with the payment will allow this transaction. This feature can also represent a potential income for us if we deduct a small amount from each transaction for the service.

The other feature will represent the ability to organize events where users can meet in real life, we can use the already available feature that allows users to localize each other in order to localize events and confirm their presence.

Thanks to this capstone project, I was able to learn more about the development of mobile application and how to use Android Studio and how to combine it with the Firebase tool. I found out that Endless possibilities can be reached and achieved combining those tools. In addition to that, I was able to use the knowledge acquired during those past 4 years here in Al Akhawayne University. It was an opportunity for me to see what I need to improve and how to organize myself when it comes to working on a project. I found out that, when it comes to working on a project, many things needs to be taken into consideration, the requirements that needs to be respected, the ability to design and define all steps needed to reach the wanted goals and also to be able to analyze and thing in a more professional way the social, technological, economic, political, legal, and ethical aspects.
XII. References:

XIII. Appendix:

Login activity code:

```java
package com.example.sakaria.techni;

import ...

public class LoginActivity extends AppCompatActivity {

    private Button LoginButton;
    private ImageView googleSignInButton;
    private EditText UserEmail, UserPassword;
    private TextView NeedNewAccountLink;
    private ProgressDialog loadingBar;

    private FirebaseAuth mAuth;

    private static final int RC_SIGN_IN = 1;
    private GoogleApiClient mGoogleSignInClient;
    private static final String TAG = "LoginActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        mAuth = FirebaseAuth.getInstance();

        NeedNewAccountLink = (TextView) findViewById(R.id.register_account_link);
        UserEmail = (EditText) findViewById(R.id.login_email);
        UserPassword = (EditText) findViewById(R.id.login_password);
        LoginButton = (Button) findViewById(R.id.login_button);
        googleSignInButton = (ImageView) findViewById(R.id.google_signin_button);
        loadingBar = new ProgressDialog(this);

        NeedNewAccountLink.setOnClickListener(v -> {
            SendUserToRegisterActivity();
        });

        LoginButton.setOnClickListener(v -> {
            AllowUserToLogin();
        });
    }
```
// Configure Google Sign In

GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
  .requestEmail(true)
  .build();

mGoogleSignInClient = new GoogleApiClient.Builder(context)
  .enableAutoManage(this, ContextCompat.getMainExecutor(this))
  .addApi(GoogleSignInApi.signUp(), new GoogleSignInApi.SignInCallback()
  {
    @Override
    public void on.signInResult(GoogleSignInResult result)
    {
      if (result.isSuccess())
      {
        FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();
        firebaseAuth.signInWithGoogle(result.getAccount());
      }
      else
      {
        Toast.makeText(this, R.string.credentials_invalid, Toast.LENGTH_SHORT).show();
      }
    }
  })
  .build();

googleSignInButton.setOnClickListener(new View.OnClickListener()
  {
    @Override
    public void onClick(View view)
    {
      mGoogleSignInClient.signIn();
    }
  });

private void signIn() {
  Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleSignInClient);
  startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);

  if (requestCode == RC_SIGN_IN) {
    loadingBar.setTitle(getString(R.string.google_sign_in_title));
    loadingBar.setMessage(getString(R.string.google_sign_in_message));
    loadingBar.show();
    loadingBar.setCancelableOnTouchOutside(true);
    if (result.isSuccess())
    {
      FirebaseAccount firebaseAccount = result.getAccount();
      firebaseAccount.signInWithGoogle();
    }
    else
    {
      Toast.makeText(this, R.string.credentials_invalid, Toast.LENGTH_SHORT).show();
    }
  }
}

private void firebaseAuthWithGoogle(FirebaseAccount acct) {
  Log.d(TAG, R.string.firebaseAuthWithGoogle);
  AuthCredential credential = GoogleAuthProvider.getCredential(acct.getEmail(), acct.getPassword());
  mAuthentication.addItemToSavedLocations(acct); // Add saved account to list
  Task<AuthResult> task = mFirebaseAuthWithGoogle.signWithCredential(credential)
    .addOnCompleteListener(new OnCompleteListener<AuthResult>()
    {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task)
      {
        if (task.isSuccessful())
        {
          Log.d(TAG, R.string.firebaseAuthWithCredential_success);
          SendDjiToMainActivity();
          loadingBar.dismiss();
        }
        else
        {
          Log.w(TAG, R.string.firebaseAuthWithCredential_failure, task.getException());
          String message = task.getException().toString();
          SendDjiToMainActivity();
          Toast.makeText(LoginActivity.this, R.string.authentication_failed + message, Toast.LENGTH_SHORT).show();
          loadingBar.dismiss();
        }
      }
    });
}
```java
@Override
protected void onStart()
{
    super.onStart();

    FirebaseUser currentUser = mAuth.getCurrentUser();
    if (currentUser != null)
    {
        SendUserToMainActivity();
    }
}

private void SendUserToLoginActivity()
{
    String email = UserEmail.getText().toString();
    String password = UserPassword.getText().toString();

    if (TextUtils.isEmpty(email))
    {
        Toast.makeText(this, R.string.username_required, Toast.LENGTH_SHORT).show();
    } else if (TextUtils.isEmpty(password))
    {
        Toast.makeText(this, R.string.password_required, Toast.LENGTH_SHORT).show();
    } else {
        loadingBar.setTitle(R.string.login);
        loadingBar.setMessage(R.string.login_message);
        loadingBar.show();
        loadingBar.setCancellable(false,
        true);

        mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(
        new TaskCompletionListener(task -> {
            if (task.isSuccessful())
            {
                SendUserToMainActivity();
                Toast.makeText(LoginActivity.this, R.string.login_successfully, Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
            } else {
                String message = task.getException().getMessage();
                Toast.makeText(LoginActivity.this, R.string.error + message, Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
            }
        })
        .addOnFailureListener(taskFailure ->
        {
            loadingBar.dismiss();
        });
    }
}

private void SendUserToMainActivity()
{
    Intent mainIntent = new Intent(LoginActivity.this, MainActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(mainIntent);
    finish();
}

private void SendUserToLoginActivity()
{
    Intent mainIntent = new Intent(LoginActivity.this, LoginActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(mainIntent);
    finish();
}

private void SendUserToRegisterActivity()
{
    Intent registerIntent = new Intent(LoginActivity.this, RegisterActivity.class);
    startActivity(registerIntent);
    finish();
}
```
package com.example.zakaria.techni;

import ...

public class RegisterActivity extends AppCompatActivity {

    private EditText UserName, UserPassword, UserConfirmPassword;
    private Button CreateAccountButton;
    private ProgressDialog loadingBar;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        mAuth = FirebaseAuth.getInstance();

        UserName = (EditText) findViewById(R.id.register_email);
        UserPassword = (EditText) findViewById(R.id.register_password);
        UserConfirmPassword = (EditText) findViewById(R.id.register_confirm_password);
        CreateAccountButton = (Button) findViewById(R.id.register_create_account);
        loadingBar = new ProgressDialog(context, this);

        CreateAccountButton.setOnClickListener(v -> {
            CreateNewAccount();
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
FirebaseUser currentUser = mAuth.getCurrentUser();
if(currentUser != null) {
    SendUserToMainActivity();
}

private void SendUserToMainActivity() {
    Intent mainIntent = new Intent( packageContext, RegisterActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivityForResult(mainIntent);
    finish();
}

private void CreateNewAccount() {
    String email = UserEmail.getText().toString();
    String password = UserPassword.getText().toString();
    String confirmPassword = UserConfirmPassword.getText().toString();

    if(TextUtils.isEmpty(email)) {
        Toast.makeText(this, text: 'Please write your email...', Toast.LENGTH_SHORT).show();
    } else if (TextUtils.isEmpty(password)) {
        Toast.makeText(this, text: 'Please write your password...', Toast.LENGTH_SHORT).show();
    } else if (TextUtils.isEmpty(confirmPassword)) {
        Toast.makeText(this, text: 'Please confirm your password...', Toast.LENGTH_SHORT).show();
    } else if (!password.equals(confirmPassword)) {

else if(!password.equals(confirmPassword)) {
    Toast.makeText(RegisterActivity.this, "Your password do not match", Toast.LENGTH_SHORT).show();
}
else {
    loadingBar.setTitle("Creating New Account");
    loadingBar.setMessage("Please wait, while creating your account...");
    loadingBar.show();
    loadingBar.setCancelable(true);

    mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(task -> {

            if(task.isSuccessful()) {
                SendUserToSetupActivity();
                Toast.makeText(RegisterActivity.this, "Account created successfully", Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
            }
            else {
                String message = task.getException().getMessage();
                Toast.makeText(RegisterActivity.this, "Error: " + message, Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
            }
        });
}

private void SendUserToSetupActivity() {
    Intent setupIntent = new Intent(RegisterActivity.this, SetupActivity.class);
    setupIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(setupIntent);
    finish();
}
Setup activity code:

```java
package com.example.zakaria.techni;

import ...

public class SetupActivity extends AppCompatActivity {

    private EditText UserName, FullName, CountryName;
    private Button SaveinformationButton;
    private CircleImageView ProfileImage;
    private ProgressDialog loadingBar;

    private FirebaseAuth mAuth;
    private DatabaseReference UsersRef;
    private StorageReference UserProfileImageRef;
    String currentUserID;
    final static int Gallery_pick = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_setup);

        mAuth = FirebaseAuth.getInstance();
        currentUserID = mAuth.getCurrentUser().getUid();
        UsersRef = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserID);
        UserProfileImageRef = FirebaseStorage.getInstance().getReference().child("Profile Images");

        UserName = (EditText) findViewById(R.id.setup_username);
        FullName = (EditText) findViewById(R.id.setup_full_name);
        CountryName = (EditText) findViewById(R.id.setup_country_name);
        SaveinformationButton = (Button) findViewById(R.id.setup_information_button);
        ProfileImage = (CircleImageView) findViewById(R.id.setup_profile_image);
        loadingBar = new ProgressDialog(context, this);

        SaveinformationButton.setOnClickListener((view) -> {
            SaveAccountSetupInformation();
        });
    }
```
ProfileImage.setOnClickListener(new View.OnClickListener() {
    Intent galleryIntent = new Intent();
galleryIntent.setAction(Intent.ACTION_GET_CONTENT);
galleryIntent.setType("image/*");
startActivityForResult(galleryIntent, GalleryPick);}
);}setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    Uri uri = Uri.parse(parent.getItemAtPosition(position).toString());
    CropImage.activity();
    .setGuidelines(CropImageView.Guidelines.ON)
    .setAspectRatio(AspectRatioFrame.aspectRatioV1)
    .start(getActivity(), this);
}
@Override
public void onNothingSelected(AdapterView<?> parent) {
}
};
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == Gallery_pick && resultCode == RESULT_OK && data != null) {
        Uri imageUri = data.getData();
        CropImage.activity()
        .setGuidelines(CropImageView.Guidelines.ON)
        .setAspectRatio(AspectRatioFrame.aspectRatioV1)
        .start(getActivity(), this);
    }
    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
        CropImage.ActivityResult result = CropImage.getActivityResult(data);
        if (resultCode == RESULT_OK) {
            leadingBar.setTitle("Profile Image");
            leadingBar.setMessage("Please wait while updating your profile image...");
            leadingBar.show();
            leadingBar.setCanceledOnTouchOutside(true);
            Uri resultUri = result.getURI();
            StorageReference ref = UserProfileImageRef.child(currentUserID + ".jpg");
            filePath.putFile(resultUri).addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    Toast.makeText(SetupActivity.this, R.string.ProfileImageUpdated, Toast.LENGTH_SHORT).show();
                    final String downloadUri = task.getResult().getDownloadUri().toString();
                } else 

On download completion, the task is finished.

```java
if (task.isSuccessful()) {
    Intent selfIntent = new Intent(MainActivity.this, SetUpActivity.class);
    startActivity(selfIntent);

    Toast.makeText(MainActivity.this, "Profile image stored to Firebase", Toast.LENGTH_SHORT).show();
    loadingBar.dismiss();
} else {
    String message = task.getException().getMessage();
    Toast.makeText(MainActivity.this, "Error: " + message, Toast.LENGTH_SHORT).show();
    loadingBar.dismiss();
}
```

In the `SetAccountSettings` method, the `Snackbar` is used.

```java
private void SetAccountSettings() {
    String username = UserNew.getText().toString();
    String full_name = FullNew.getText().toString();
    String country = CountryNew.getText().toString();

    if (TextUtils.isEmpty(username)) {
        Toast.makeText(MainActivity.this, "Please write your Username...", Toast.LENGTH_SHORT).show();
    }
```
if (TextUtils.isEmpty(fullName)) {
    Toast.makeText(context, "Please write your Full Name...", Toast.LENGTH_SHORT).show();
}

if (TextUtils.isEmpty(country)) {
    Toast.makeText(context, "Please write your country...", Toast.LENGTH_SHORT).show();
} else {

    loadingBar.setTitle("Saving Information");
    loadingBar.setMessage("Please wait, while adding information to your Account...");
    loadingBar.show();
    loadingBar.setCancelableOnTouchOutside(true);

    HashMap userMap = new HashMap();
    userMap.put("username", username);
    userMap.put("fullname", fullName);
    userMap.put("country", country);
    userMap.put("Art Category", "none");
    userMap.put("status", "Hung artist 23 years old");
    userMap.put("gender", "Male");

    UserRef.updateChildren(userMap).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            SendUserToMainActivity();
            Toast.makeText(context, "Account created Successfully...", Toast.LENGTH_LONG).show();
            loadingBar.dismiss();
        } else {
            String message = task.getException().getMessage();
            Toast.makeText(context, "Error: " + message, Toast.LENGTH_SHORT).show();
            loadingBar.dismiss();
        }
    });

private void SendUserToMainActivity() {
    Intent mainIntent = new Intent(packageContext, SetupActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(mainIntent);
    finish();
}
Settings activity code:

```java
public class SettingsActivity extends AppCompatActivity {

    private Toolbar mToolBar;

    private EditText userName, userProfName, userStatus, userCountry, userGender, userArtType;
    private Button UpdateAccountSettingsButton;
    private CircleImageView userProfImage;

    private DatabaseReference SettingsUserRef;
    private FirebaseAuth mAuth;

    private String currentUserId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);

        mAuth = FirebaseAuth.getInstance();
        currentUserId = mAuth.getCurrentUser().getUid();
        SettingsUserRef = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserId);

        mToolBar = (Toolbar) findViewById(R.id.settings_toolbar);
        setSupportActionBar(mToolBar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        userName = (EditText) findViewById(R.id.settings_username);
        userProfName = (EditText) findViewById(R.id.settings_profile_full_name);
        userStatus = (EditText) findViewById(R.id.settings_status);
        userCountry = (EditText) findViewById(R.id.settings_country);
        userGender = (EditText) findViewById(R.id.settings_gender);
        userArtType = (EditText) findViewById(R.id.settings_art_type);
        userProfImage = (CircleImageView) findViewById(R.id.settings_profile_image);

        UpdateAccountSettingsButton = (Button) findViewById(R.id.update_account_settings_buttons);
    }
}
```
Profile activity code:

```java
public class ProfileActivity extends AppCompatActivity {

    private TextView userName, userProNome, userStatus, userCountry, userGender, userArtType;
    private CircleImageView userProfileImage;
    private DatabaseReference profileRef;
    private FirebaseAuth mAuth;
    private String currentUserId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        mAuth = FirebaseAuth.getInstance();
        currentUserId = mAuth.getCurrentUser().getUid();
        profileRef = FirebaseDatabase.getInstance().getReference().child("Users").child(currentUserId);

        userName = (TextView) findViewById(R.id.my_username);
        userProNome = (TextView) findViewById(R.id.my_profile_full_name);
        userStatus = (TextView) findViewById(R.id.my_profile_status);
        userCountry = (TextView) findViewById(R.id.my_country);
        userArtType = (TextView) findViewById(R.id.my_art_type);
        userProfileImage = (CircleImageView) findViewById(R.id.my_profile_pic);

        profileRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    String myProNome = dataSnapshot.child("profileName").getValue().toString();
                    String myProImage = dataSnapshot.child("imageName").getValue().toString();
                    String myProStatus = dataSnapshot.child("status").getValue().toString();
                    String myCountry = dataSnapshot.child("country").getValue().toString();
                    String myArtType = dataSnapshot.child("Art Category").getValue().toString();
                }
            }
        });
```
Post activity code:

```java
public class PostActivity extends AppCompatActivity {

    private Toolbar mToolbar;
    private ProgressDialog loadingBar;

    private ImageButton SelectPostImage;
    private Button UpdatePostButton;
    private EditText PostDescription;

    private static final int Gallery_pick = 1;
    private Uri ImageUri;
    private String Description;

    private StorageReference PostImagesReference;
    private DatabaseReference UsersRef, PostsRef;
    private FirebaseAuth mAuth;

    private String saveCurrentDate, saveCurrentTime, postRandomName, downloadUrl, current_user_id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_post);

        mAuth = FirebaseAuth.getInstance();
        current_user_id = mAuth.getCurrentUser().getUid();

        PostImagesReference = FirebaseStorage.getInstance().getReference();
        UsersRef = FirebaseDatabase.getInstance().getReference().child("Users");
        PostsRef = FirebaseDatabase.getInstance().getReference().child("Posts");

        SelectPostImage = (ImageButton) findViewById(R.id.select_post_image);
        UpdatePostButton = (Button) findViewById(R.id.update_post_button);
        PostDescription = (EditText) findViewById(R.id.post_description);
        loadingBar = new ProgressDialog(this);
    }
```
mToolbar = (Toolbar) findViewById(R.id.update_post_page_toolbar);
setSupportActionBar(mToolbar);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);
getSupportActionBar().setTitle(“Update Post”);

SelectPostImage.setOnClickListener(v -> { OpenGallery(); });
UpdatePostButton.setOnClickListener(v -> { ValidatePostInfo(); });

private void ValidatePostInfo() { 

    Description = PostDescription.getText().toString();

    if(ImageUri == null) {
        Toast.makeText( context, “Please select an image…”, Toast.LENGTH_SHORT).show();
    } else if(TextUtils.isEmpty(Description)) {
        Toast.makeText( context, “Please fill in the description…”, Toast.LENGTH_SHORT).show();
    } else {
        loadingBar.setTitle(“Add new post.”);
        loadingBar.setMessage(“Please wait, while updating your post…”);
        loadingBar.show();
        loadingBar.setCancelableOnTouchOutside(true);
        StoringImageToFirebaseStorage();
    }
}
private void storingImageInImageStorage() {
    Calendar calForDate = Calendar.getInstance();
    SimpleDateFormat currentDate = new SimpleDateFormat("dd-MMM-yyyy");
    saveCurrentDate = new Date().format(calForDate, getTime());

    Calendar calForTime = Calendar.getInstance();
    SimpleDateFormat currentTime = new SimpleDateFormat("HH:mm:ss");
    saveCurrentTime = new Date().format(calForTime, getTime());

    postRandomName = saveCurrentDate + saveCurrentTime;

    StorageReference filePath = PostImageStorageReference.child("Post Images").child(ImageUri.getLastPathSegment() + postRandomName + ".jpg").
        putFile(ImageUri).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                downloadRef = task.getDownloadUrl().toString();
                Toast.makeText(this, R.string.image_uploded_successfully, Toast.LENGTH_SHORT).show();
                postRandomName = saveCurrentDate + saveCurrentTime;
                SavingPostInformationToDatabase();
            } else {
                String message = task.getException().getMessage();
                Toast.makeText(this, R.string.error + message, Toast.LENGTH_SHORT).show();
            }
        });
}

private void SavingPostInformationToDatabase() {
    UsersRef.child(currentUser_id).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) {
                String UserFullName = dataSnapshot.child("fullName").getValue().toString();
                String UserProfileImage = dataSnapshot.child("profileImage").getValue().toString();

                HashMap postsMap = new HashMap();
                postsMap.put("id", currentUser_id);
                postsMap.put("date", saveCurrentDate);
                postsMap.put("time", saveCurrentTime);
                postsMap.put("subscription", Subcription);
                postsMap.put("profileImage", UserProfileImage);
                postsMap.put("fullName", UserFullName);

                PostsRef.child(currentUser_id + postRandomName).updateChildren(postsMap);
                dataSnapshot.child("PostImage").addOnCompleteListener(task) {
                    if (task.isSuccessful()) {
                        SendUserToMainActivity();
                        Toast.makeText(this, R.string.post_uploded_successfully, Toast.LENGTH_SHORT).show();
                        loadingBar.dismiss();
                    } else {
                        Toast.makeText(this, R.string.error, Toast.LENGTH_SHORT).show();
                        loadingBar.dismiss();
                    }
                });
            }
        }
    });
}

@Override
public void onCancelled(DatabaseError databaseError) {
}
}
private void OpenGallery() {

    Intent galleryIntent = new Intent();
galleryIntent.setAction(Intent.ACTION_GET_CONTENT);
galleryIntent.setType("image/*");
startActivityForResult(galleryIntent, Gallery_pick);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if(requestCode == Gallery_pick && resultCode == RESULT_OK && data != null){
        ImageUri = data.getData();
        SelectPostimage.setImageURI(ImageUri);
    }
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();

    if(id == android.R.id.home){
        SendUserToMainActivity();
    }

    return super.onOptionsItemSelected(item);
}

private void SendUserToMainActivity() {

    Intent mainIntent = new Intent(packageContext, PostActivity.this, MainActivity.class);
    startActivity(mainIntent);
}
public class ClickPostActivity extends AppCompatActivity {

    private ImageView PostImage;
    private TextView PostDescription;
    private Button DeletePostButton, EditPostButton;
    private DatabaseReference ClickPostRef;
    private FirebaseAuth mAuth;

    private String Postkey, currentUserID, databaseUserID, description, image;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_click_post);

        mAuth = FirebaseAuth.getInstance();
        currentUserID = mAuth.getCurrentUser().getUid();

        Postkey = getIntent().getExtras().get("PostKey").toString();
        ClickPostRef = FirebaseDatabase.getInstance().getReference().child("Posts").child(Postkey);

        PostImage = (ImageView) findViewById(R.id.click_post_image);
        PostDescription = (TextView) findViewById(R.id.click_post_description);
        DeletePostButton = (Button) findViewById(R.id.delete_post_button);
        EditPostButton = (Button) findViewById(R.id.edit_post_button);

        DeletePostButton.setVisibility(View.INVISIBLE);
        EditPostButton.setVisibility(View.INVISIBLE);

        ClickPostRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()){
                    description = dataSnapshot.child("description").getValue().toString();
                    image = dataSnapshot.child("postimage").getValue().toString();
                    databaseUserID = dataSnapshot.child("uid").getValue().toString();
                }
            }
        });
    }
PostDescription.setText(description);
Picasso.with(ClickPostActivity.this).load(image).into(PostImage);

if(currentUserID.equals(databaseUserID)) {
    PostButton.setVisibility(View.VISIBLE);
    EditPostButton.setVisibility(View.VISIBLE);
}
EditPostButton.setOnClickListener(v -> {
    EditText description = getCurrentPostDescription();
});

@Override
public void onCancelled(DatabaseError databaseError) {
    DeletePostButton.setOnClickListener(v -> {
        onDeleteCurrentPost();
    });
}

private void onDeleteCurrentPost(String description) {
    AlertDialog.Builder builder = new AlertDialog.Builder(ClickPostActivity.this);
    builder.setTitle("Edit Post");

    final EditText inputField = new EditText(ClickPostActivity.this);
    inputField.setText(description);
    builder.setView(inputField);

    builder.setPositiveButton("Update", (dialog, which) -> {
        PostDescription.setText(inputField.getText().toString());
        Toast.makeText(ClickPostActivity.this, "Post updated successfully...", Toast.LENGTH_SHORT).show();
    });
    builder.setNegativeButton("Cancel", (dialog, which) -> {
        dialog.cancel();
    });

    Dialog dialog = builder.create();
    dialog.show();
    dialog.getWindow().setBackgroundDrawableResource(android.R.color.background_light);
}

private void onDeleteCurrentPost() {
    PostDescription.removeValue();
    SendUserToMainActivity();
    Toast.makeText(context, "Post deleted", Toast.LENGTH_SHORT).show();
}

private void SendUserToMainActivity() {
    Intent mainIntent = new Intent(packageContext, ClickPostActivity.this, MainActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(mainIntent);
    finish();
}