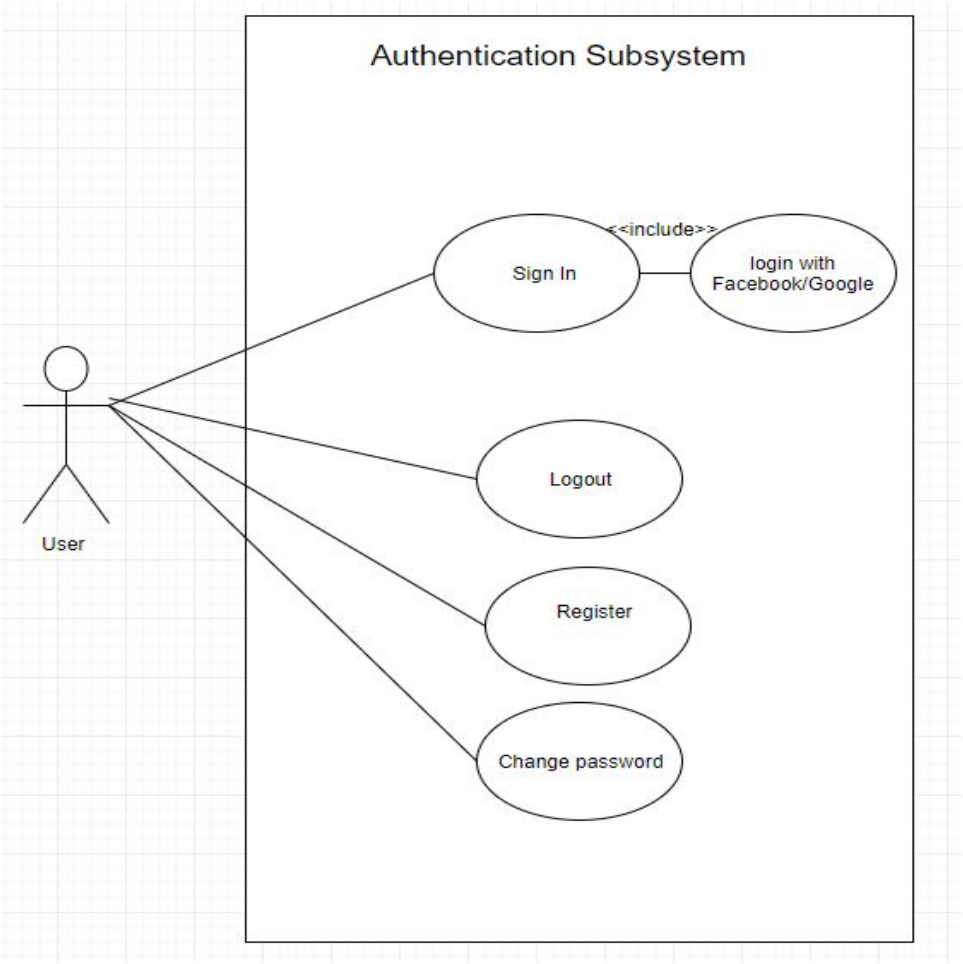
Architecture diagram report

Xudong Zhang (901065),
Hao Liu (900636),
Chenghan Li (900581),
Kevin Liang (864665).

1. Use case diagram

1.1 Authentication Subsystem

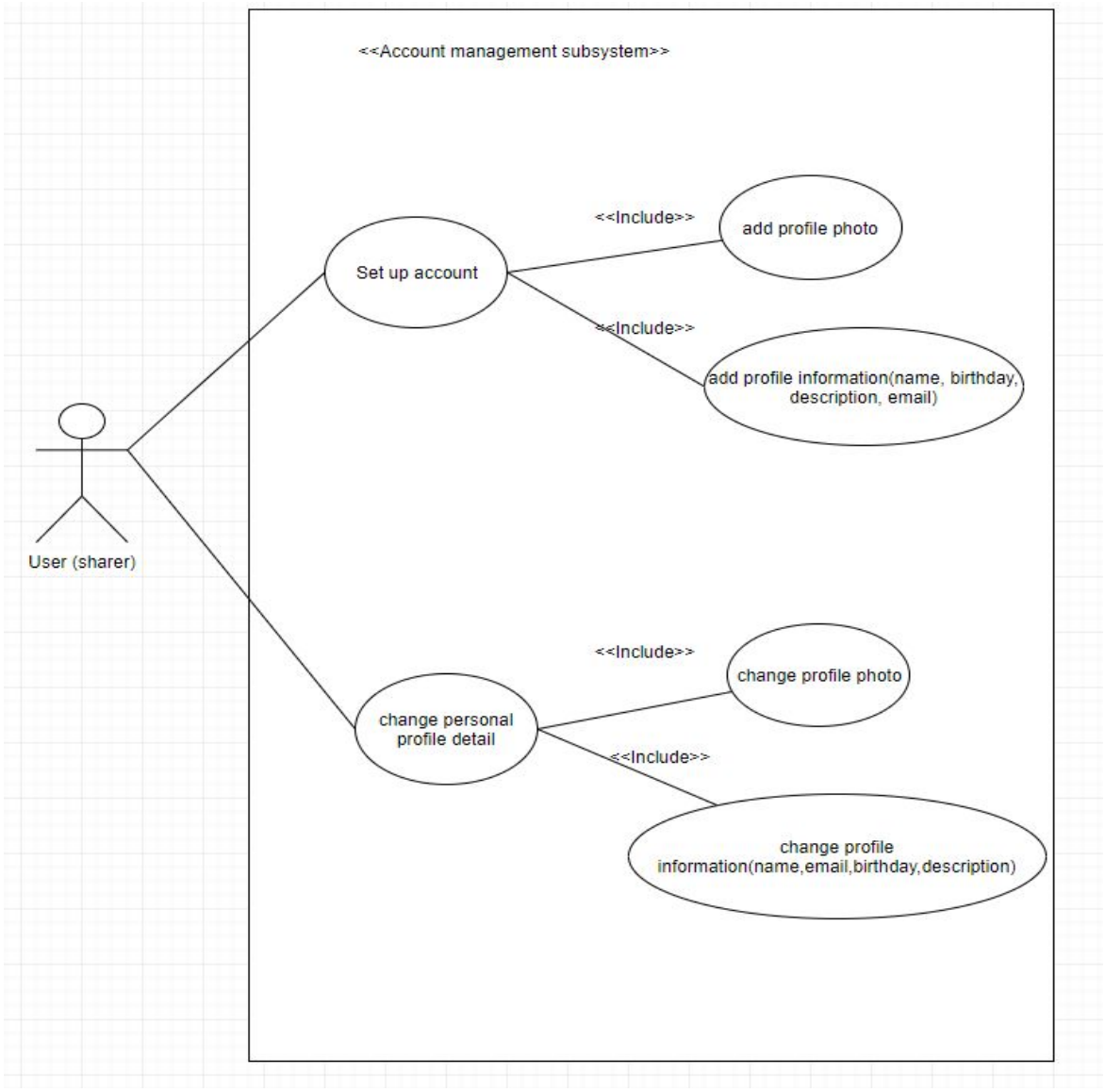


Link:
https://drive.google.com/open?id=109rckEAZx09WrfmzPoiunhTlNZ43ro-k
Description: This is a use case diagram for the authentication subsystem.
Main scenarios and alternate scenarios in authentication subsystem:

1. The user can log in the APP, by entering email and password or log in with Facebook/Google.
   1a. Log in successful when the email and password are correct.
   1b. Login unsuccessful when email is wrong.
   1c. Login unsuccessful when password is wrong.
   1d. Login unsuccessful when email and password are both wrong.
2. The user can log out the APP, by clicking the logout button.

3. The user can Register for a new account.
   3a. New account create successful when all the details are correct.
   3b. Fail to create a new account when the details entered are incorrect.
4. The user can change the password
   4a. Change successful.
   4b. Fail to change the password(password before changing and after changing are same or new password incorrect).

1.2 Account Management Subsystem

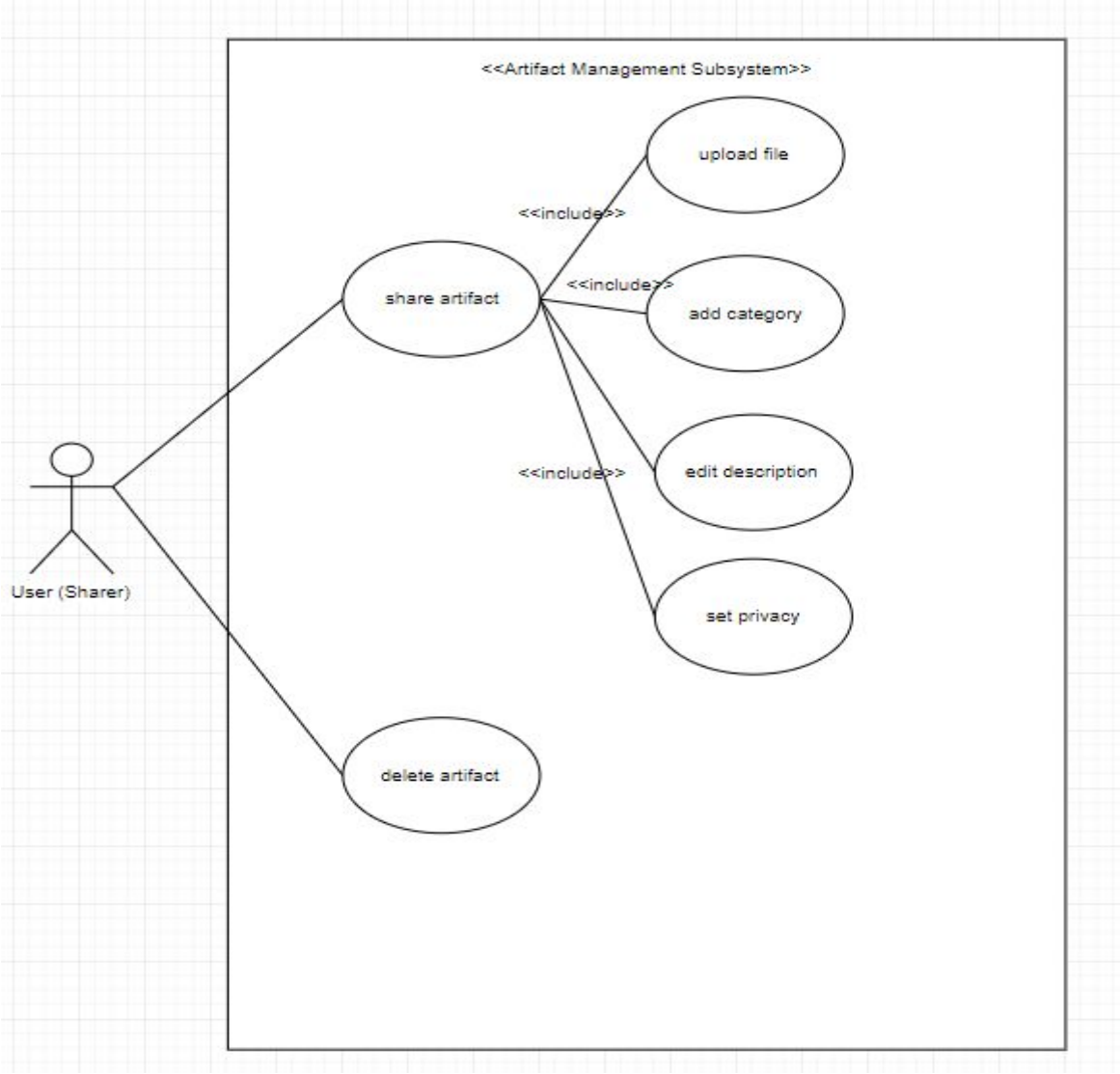


Link:
https://drive.google.com/open?id=1B9EGlCYqoXgEOrPd2ZABGe8LF2DvYBqp

Description: This is a use case diagram for account management subsystem. In this case the user is the sharer that he can mainly do two actions, set up account and change personal profile details.
Main scenarios and alternate scenarios in account management subsystem:

1. The user can set up account by adding profile photo(icon) or adding profile information (name, email, birthday, profile description)
2. The user can change personal profile details by changing profile photo(icon) or changing profile information (name, email, birthday, profile description)

1.3 Artifact Management Subsystem



Link:
https://drive.google.com/open?id=137_58oR02UvnHO-G7_jhgpY-MAXrz-3N

Description: This is a use case diagram for the artifact management subsystem. The user can only be the sharer because only sharer can share(upload) or delete artifact in this subsystem.

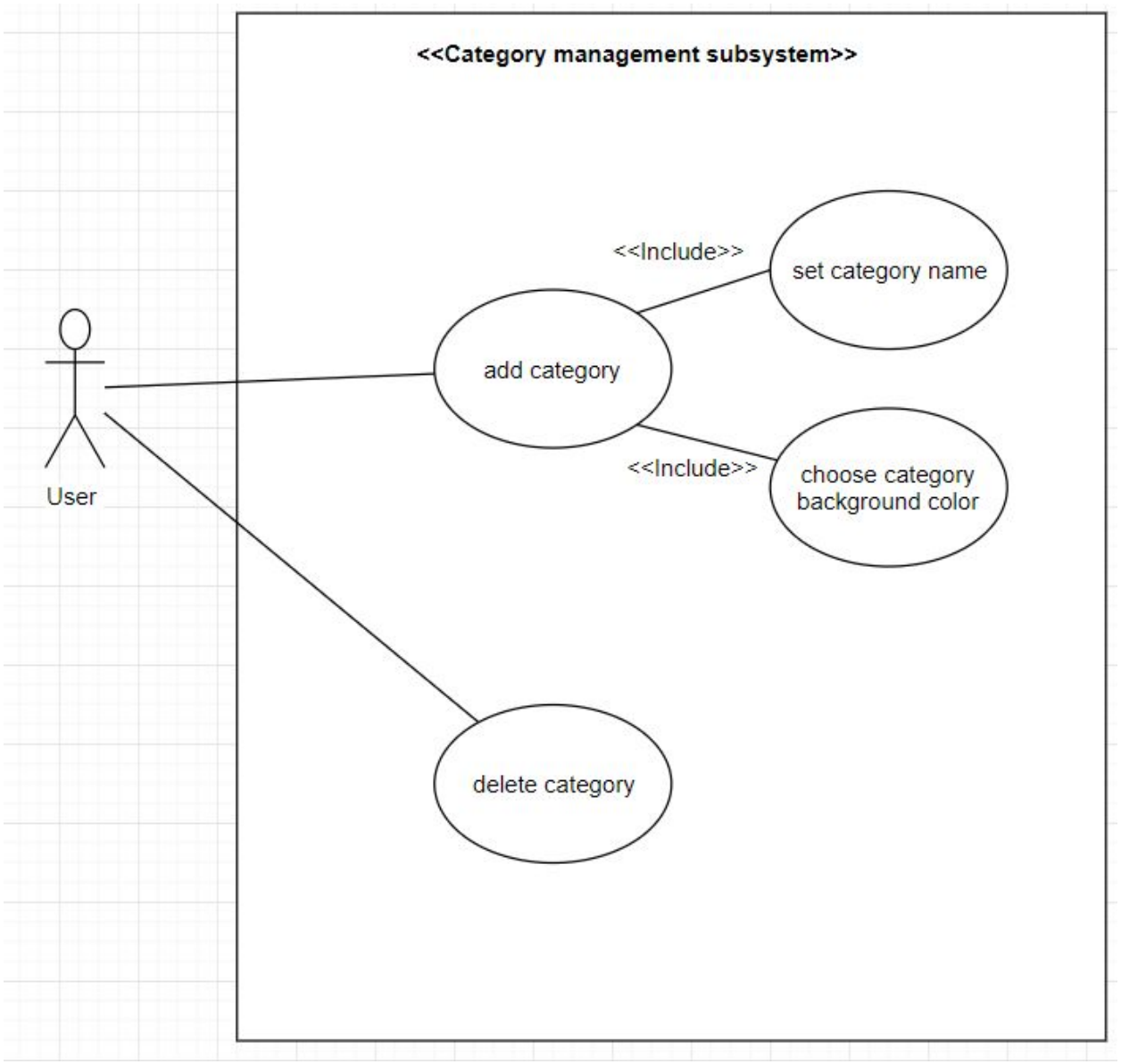
Main scenarios and alternate scenarios in account management subsystem:

1. The user can share the artifacts in the APP by uploading the file, add a category to the artifact, edit description to the artifact and set the privacy(public or private) of the artifact.

   1a. Share artifacts successful

   1b. Share artifacts unsuccessful because the uploaded file is not in a valid type (photo,document,video).
2. The user can delete the artifact

1.4 Category Management Subsystem



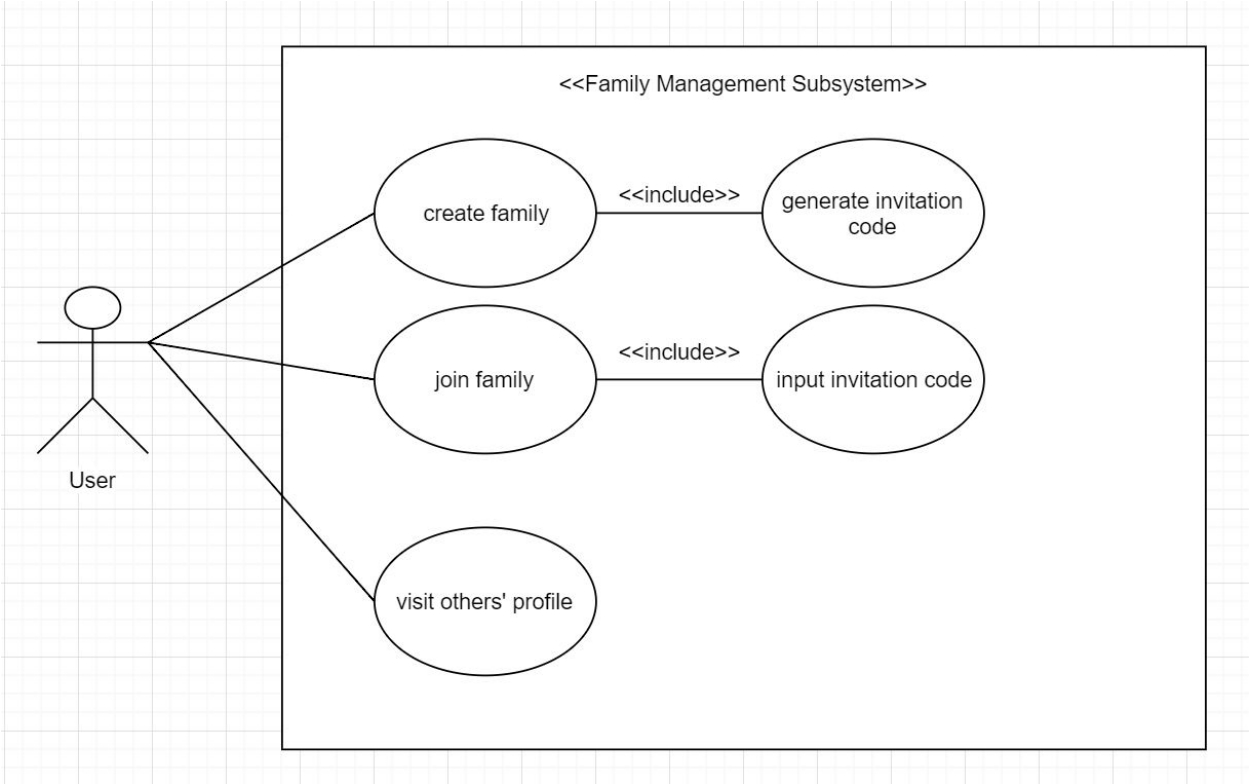
Link:
https://drive.google.com/open?id=1uO0AJrw68Hkf_v8llaYeK9K9Ez-R0X57

Description:

Main scenarios and alternate scenarios in category management subsystem:

1. The user can add a new category. The user is asked to set the name of the category and the background colour from a colour list.
   1a. Successfully added if the category does not exist.
   2b. Failed to add if the category already exists.
2. The user can delete a category added before.

1.5 Family Subsystem



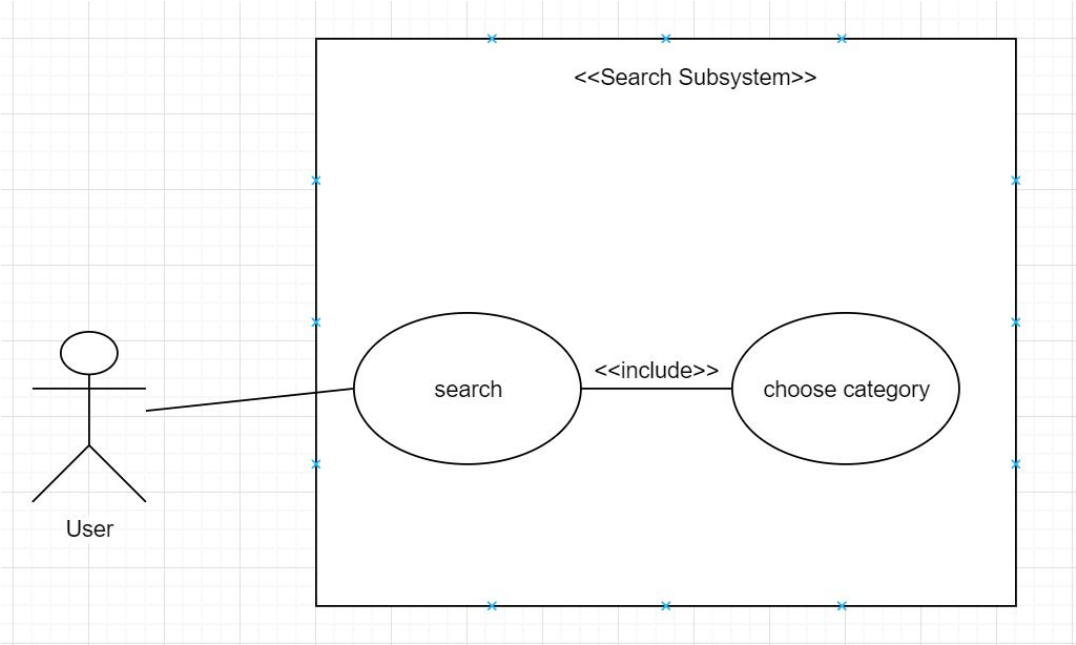
Link:
https://drive.google.com/open?id=1ME-_6mtm615LwBkkyqaczuZFwiE7m8iP

Description:

Main scenarios and alternate scenarios in family subsystem:

1. The user can create family if the user doesn't have a family existing. After family is created, an invitation code will be generated
2. The user can join a family if the user does not have a family and the family has been created.
   2a. Join successfully if the invitation code is valid.
   2b. Failed to join if the code is invalid.
3. The user can access other family members' profiles through family system.

1.6 Search Subsystem

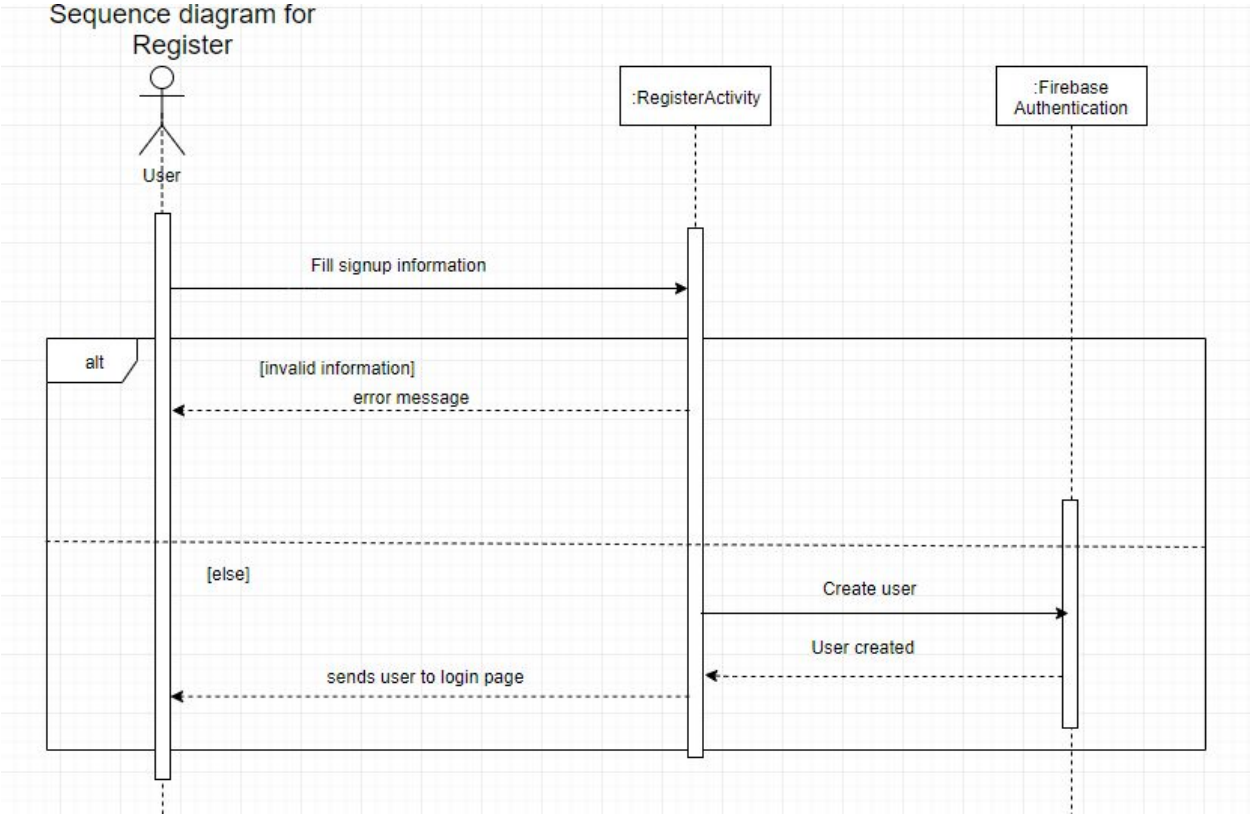


Link:
https://drive.google.com/open?id=1uWoX8LWl6pHEhlpixopWM4mrhR6jS_X9
Description:
Main scenarios and alternate scenarios in search subsystem:

1. The user can search an artifact by choosing a category and find the artifact in side the chosen category

2. Sequence diagram

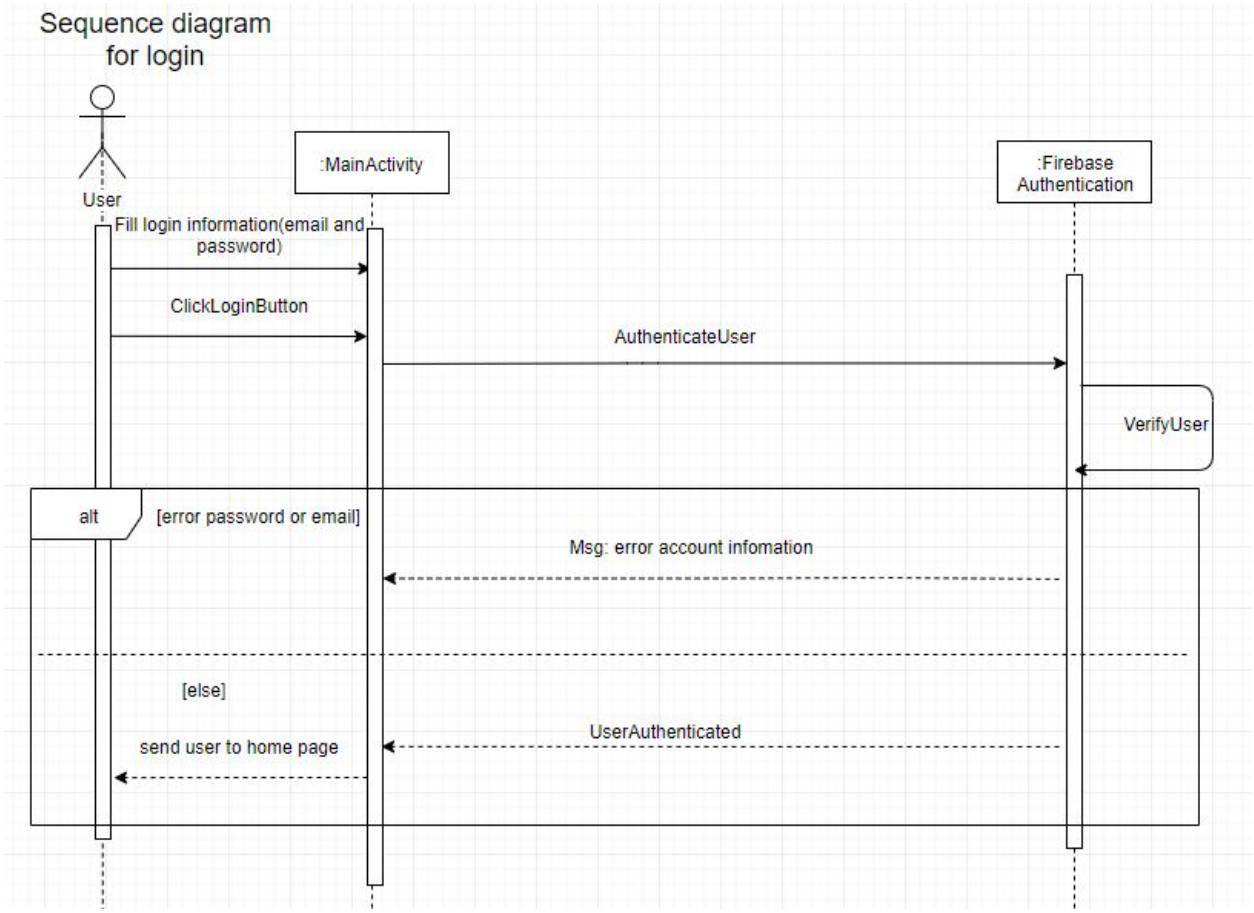
2.1 Register



Link:
https://drive.google.com/open?id=1P5TqgNrCqu4JsDUc09RFItib7ECy0l_7

Description: In the register system, All the user can register for a new account from register pages after fill the sign up information. The information will be verified, (1)If the Information is invaild(too short or too long) the application will send error message to user.(2)if the information is valid (e.g. appropriate length, valid email address) the application will send a request to firebase authentication subsystem for creating an account, once the account is created for the user the application will transfer the user to the login page from register page.

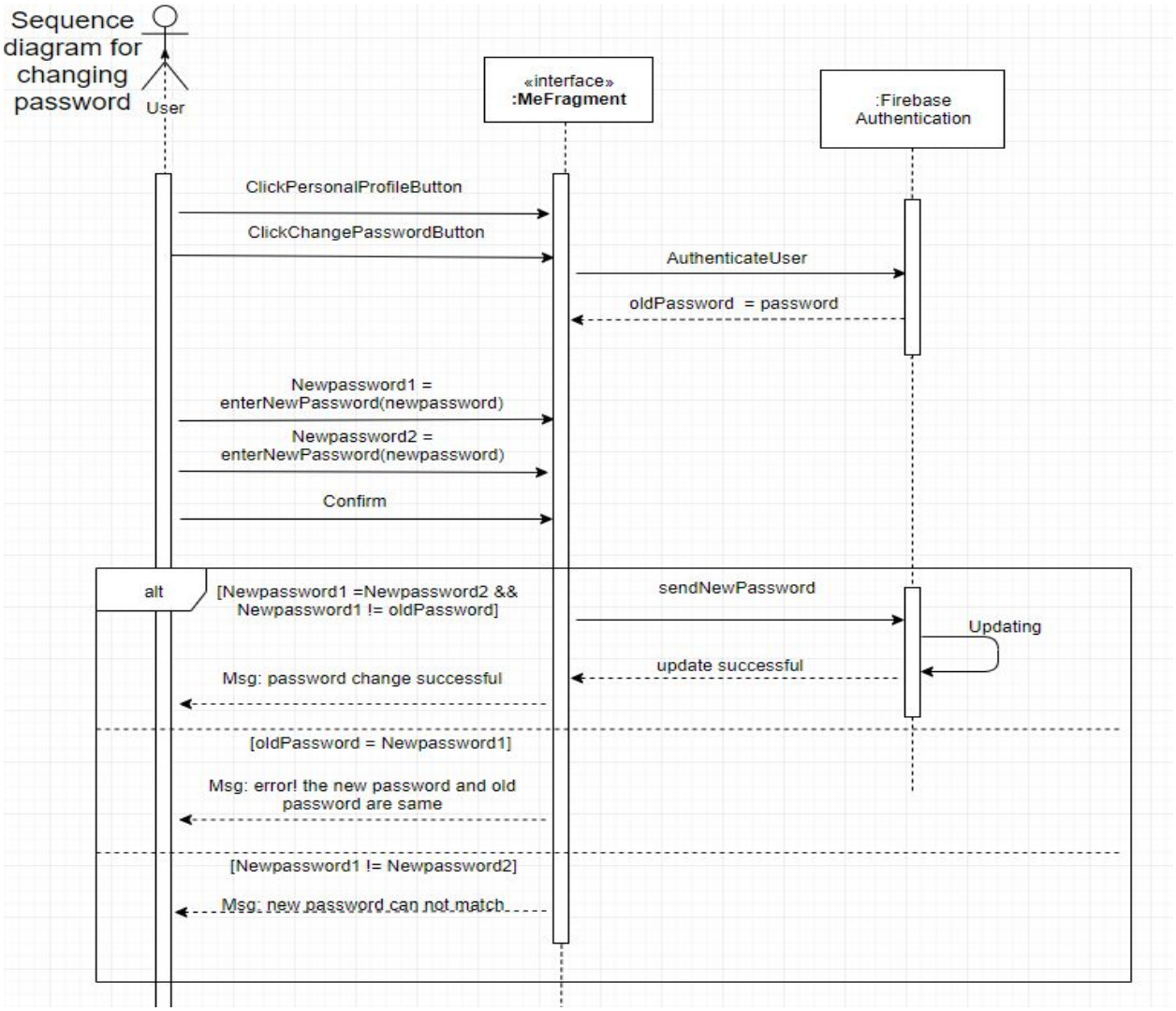
2.2 Login



Link:
https://drive.google.com/open?id=1B7KJ7zlUTqB-allFwXwhzQH-M4HnlxpX

Description: In the login system, Once the user entered the login information(email, password) and press the login button, these information will pass to the firebase to verify the user to check whether the information is correct or not. (1)If there is any error checked by authentication subsystem, it will send an error message to the login page. (2) If the login information is correct, login will be successful the application will transfer the user to the home page from login page.

2.3 Change password

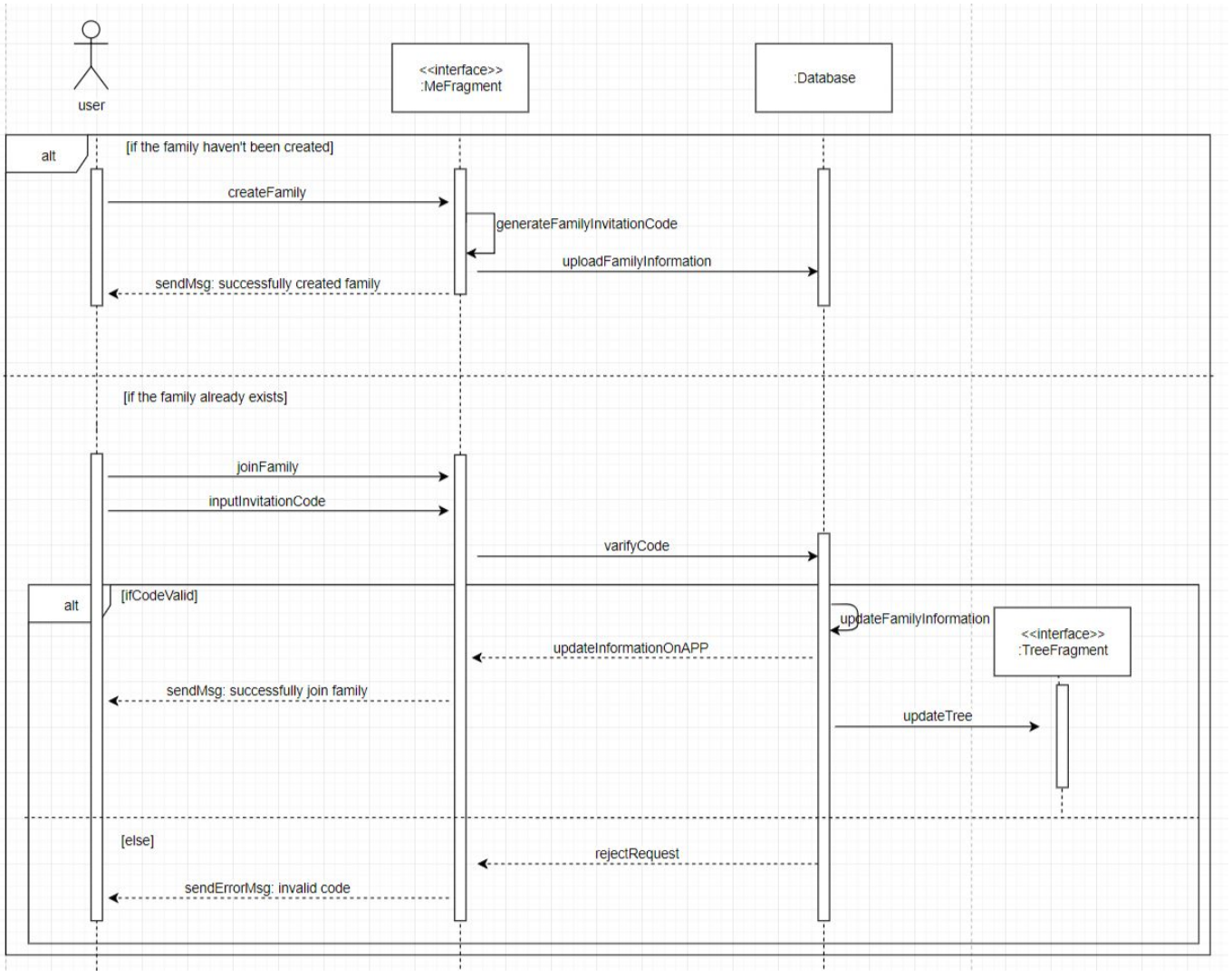


Link:
https://drive.google.com/open?id=1FTF-nj-x09SRwctU_PDP_hQBwUxuQbBt

Description: Users can change password in the password changing system. This action happens when the user visits the personal profile page and press the change password button. When the change password button is clicked the application will authenticate the user in firebase and get the old password of the user. The user is required to enter the new password twice. (1) if the new password is the same as the old password the application will return an error message to tell the user the new password is unchanged.

(2) if the new password that entered twice can not match, the application will send an error message and tell the user the new password can not match.
(3) If the new password is not the same as the old password and two passwords that the user entered are matched, the application will send the new password to the firebase to update the user's password. Once the application receives the update successful message, the application will then send a message to tell the user the password changing is successful.

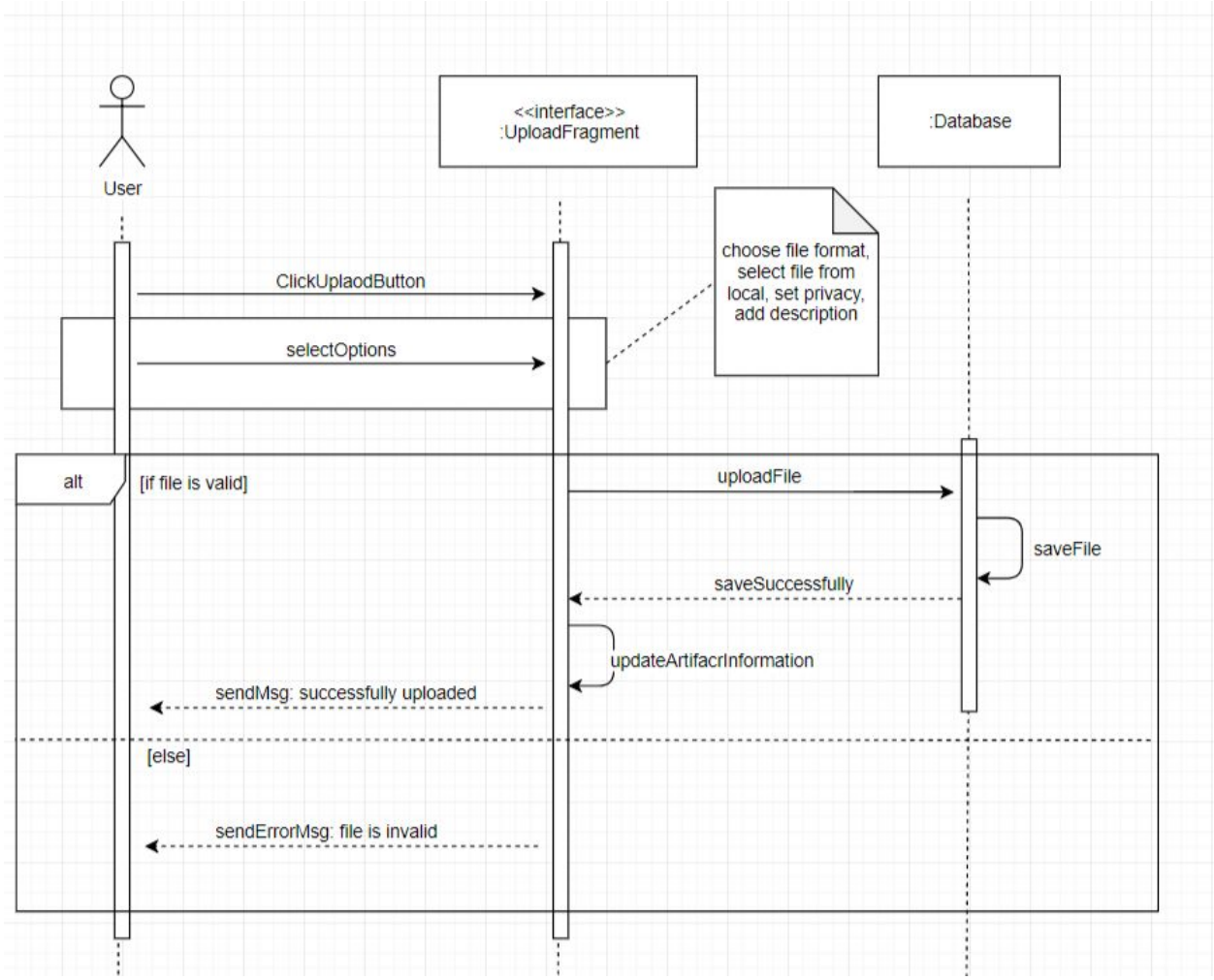
2.4 Create / Join Family



Link:
https://drive.google.com/open?id=1xmrrVCG4qalEWlzLIp8qdlmxYK0fgt9l
Description: In the family system, (1) if the user's family has not been created, the user will need to create a new family. Once the family is created, the APP will generate a unique invitation code. Then all the information about the new family will be updated in

the database. The app will display a message to the user, telling the user that the new family has been successfully created.
(2) If the user wants to join a family that already exists, the user will be asked to input the invitation code. The APP will send the code to the database to verify whether it is valid or not. (a) If the code is valid, the information about the family and user will be updated on both the database and profile page. The tree diagram on the tree fragment will be automatically updated. (b) If the code is invalid, the database will reject the request from APP and APP will send the user an error message.

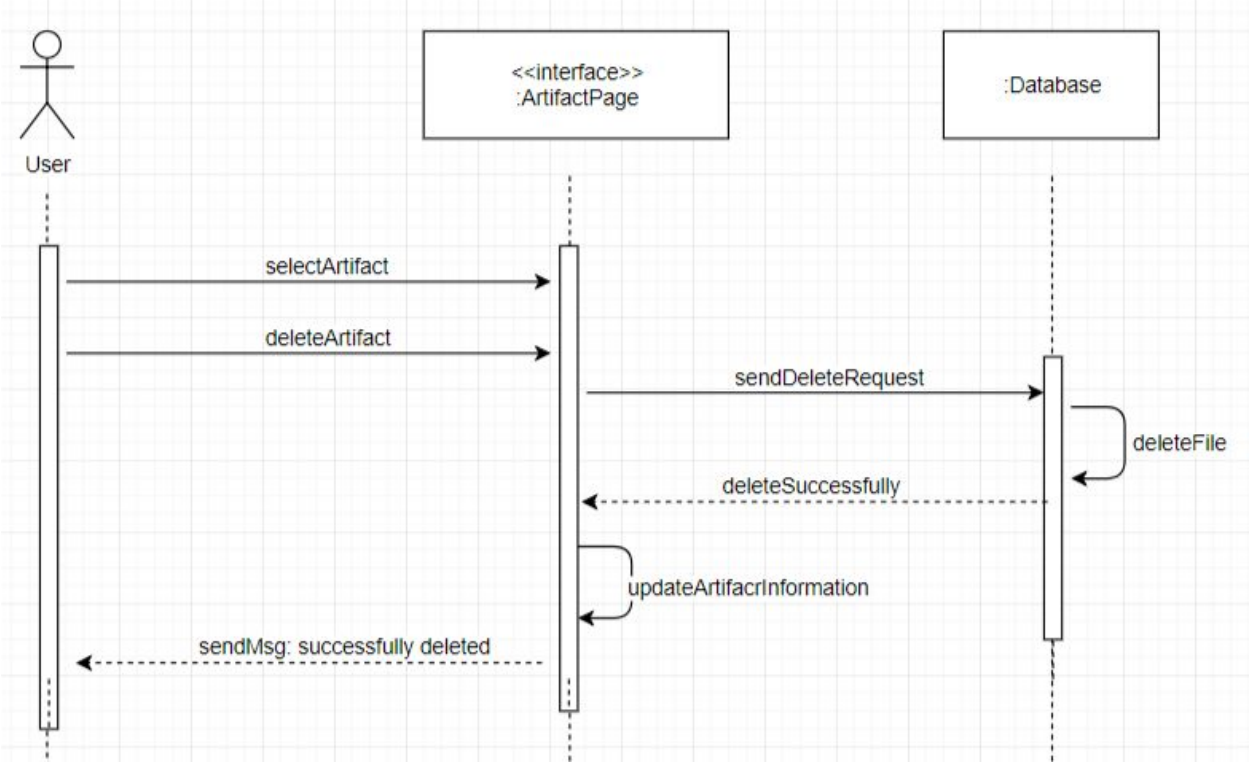
2.5 Update an artifact



Link:
https://drive.google.com/open?id=1mR5Sc8dWHd41pj845fUnfbcCx73uYHSA

Description: If the user clicks upload button on the home page, it will deliver the user to the upload fragment. In the upload page, the user needs to choose the file format, select file from local, set privacy and add a description. Then the APP will verify the information. (1) If everything is valid, the APP will send the file to the database and the database will save the fill and tell the APP that the file has been saved successfully. After receiving the feedback, the APP will update the information on the artifact page and send the user a message telling that the file has been successfully updated. (2) If any information is not valid, the APP will send an error message to the user.

2.6 Delete an artifact



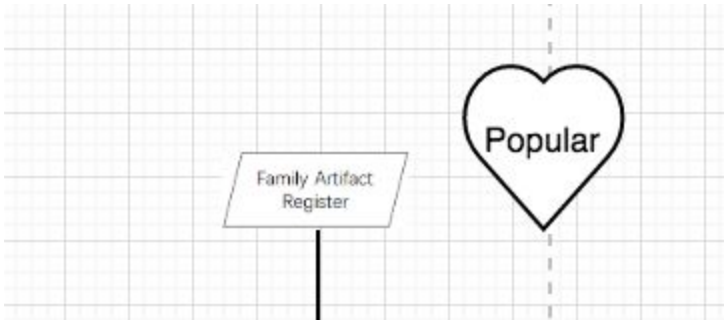
Link:
https://drive.google.com/open?id=135p3ozlbm1376mPzOUjv9kBmsZt7wupY
Description: If the user wants to delete an uploaded artifact, go to the personal artifact page, select the artifact and delete. The APP will send a delete request to the database, then the database will delete the file and send the APP a message. The artifact information will be updated automatically by the APP. A message will be sent to the user telling that the artifact has been successfully deleted.

3.Goal Model

Link:
https://drive.google.com/open?id=1ER4pyJNbNXJmrFKhiCTfETq_8jl6e_Rn

3.1 Family Artifact Register

While we want to have a popular software of a family artifact register, this is the final goal of the project. To achieve this goal, there are 5 functional goals to be achieved to support the success of it: to search documents, artifact management, category management, account management and family management.

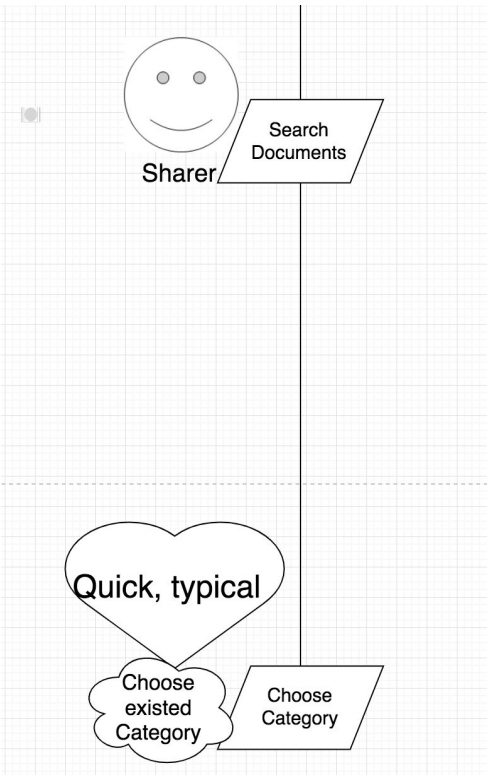


3.2 Search Documents

This is the functional goal that involves the sharer, which indicates a goal that the sharer could search the documents uploaded to the database. It requires a functional goal that allows the sharer to choose categories.

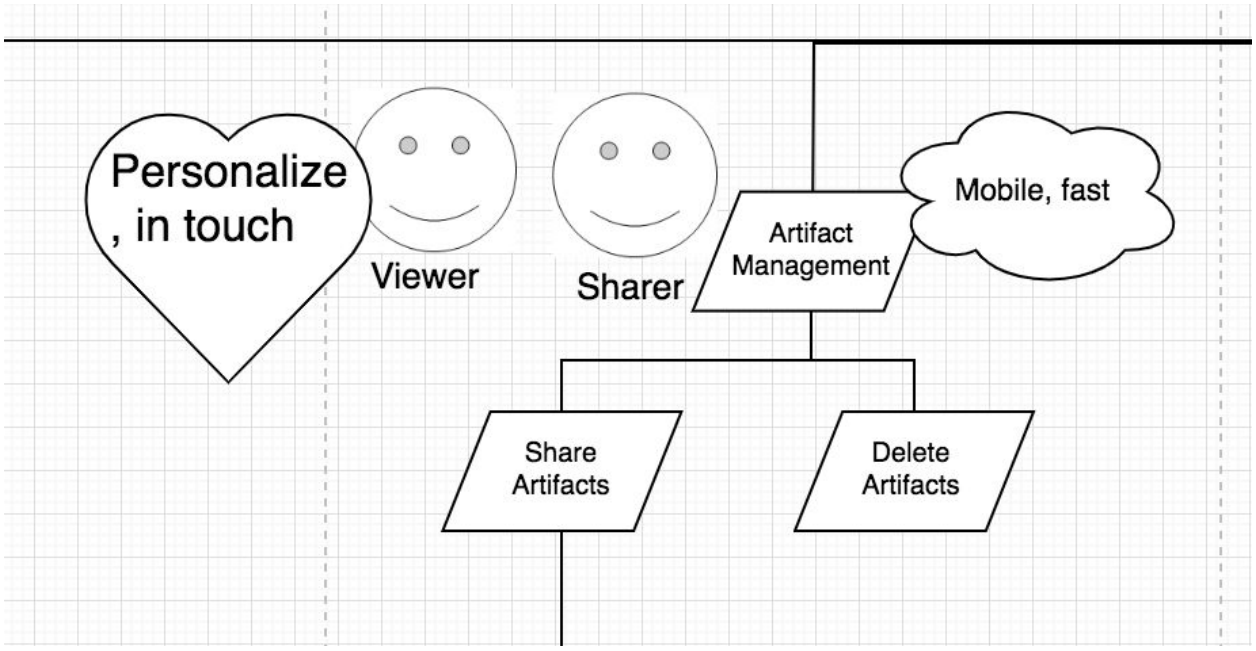
3.2.1 Choose Category

This is a functional goal that allows the sharer to choose specific and existed category that has been created and saved in (the database of) the app. It is expected to allow the sharer having a quick experience of looking for typical documents, with the support of the unfunctional goal - (the sharers) choose existed category.



3.3 Artifact Management

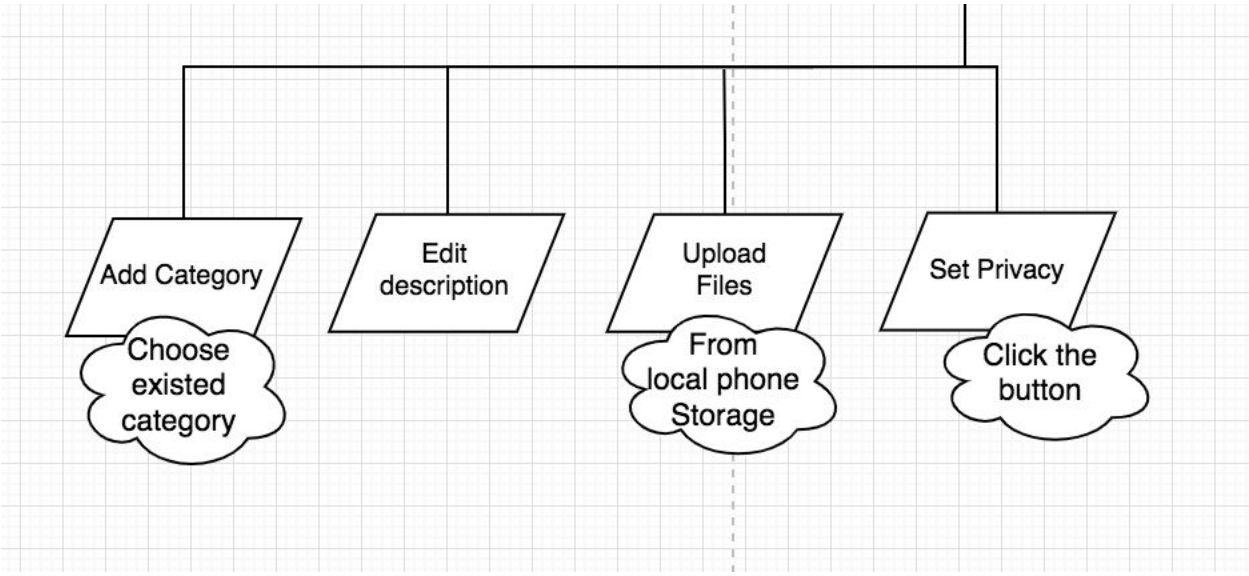
This is the functional goal that involves both the sharer and the viewer. It is expected that both of them could personalize their experience in artifact-related actions in the app, which also allows them to get in touch with others. The soft goal in this part is expected to achieve a mobile and fast control and management. The goal could be further divided into the sharing and the deletion part.



3.3.1 Share Artifacts

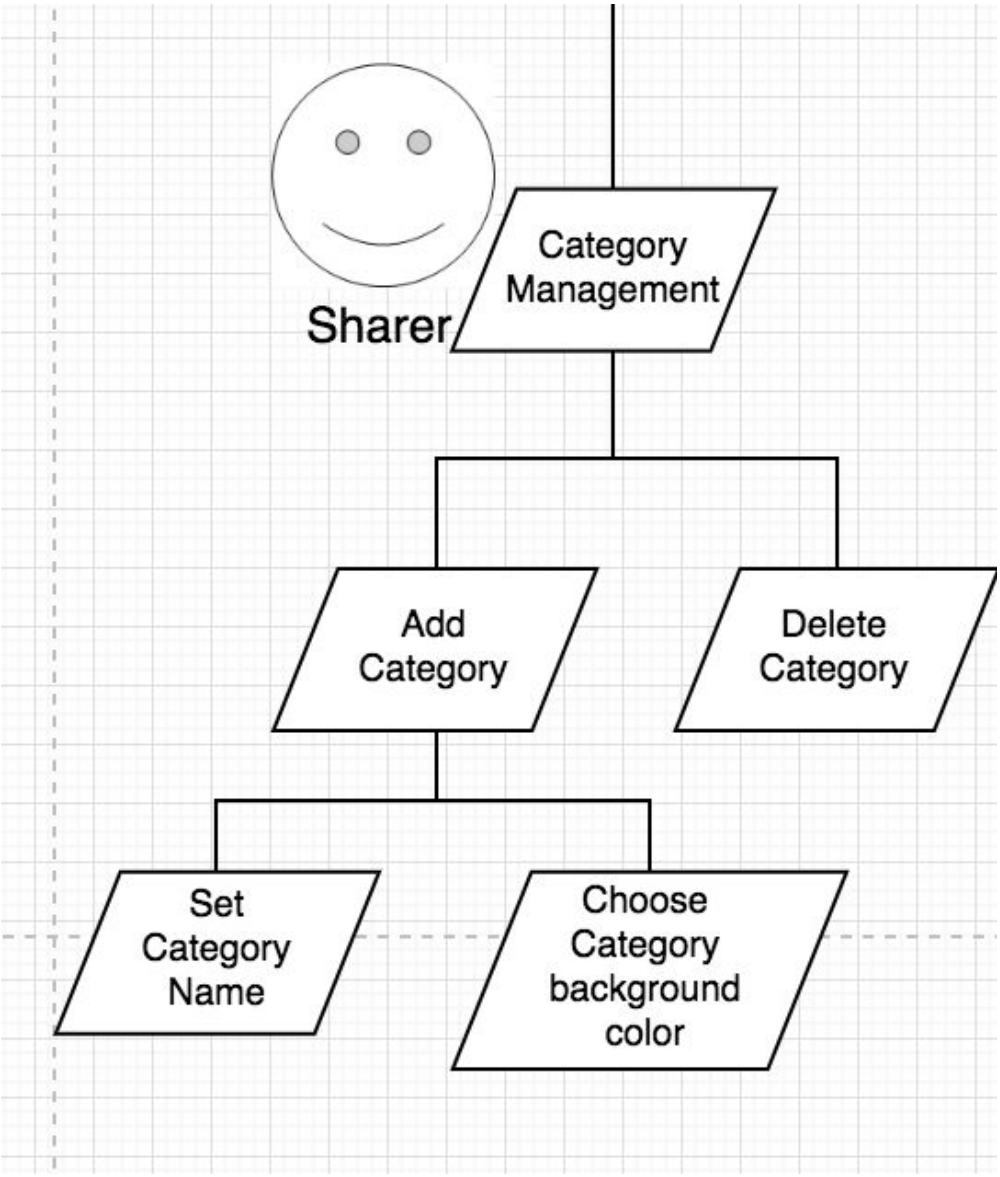
It is a functional goal that includes 4 concrete goals to achieve: add category, edit description, upload files and set privacy. Firstly, by choosing existed category in the app, the artifact could be labelled with a personalized and typical tag created by the sharer. Secondly, the sharer could add and edit the text that describes the artifact being uploaded. Thirdly, with the help of the unfunctional goal that choosing documents from the local phone storage (camera roll, local files and etc.), the sharer could be able to upload files related to the artifact. Lastly, the sharer could set the privacy. There would be a specific button for the sharer to click on, changing the privacy between public and private.

This functional goal allows the sharer to delete any artifact that is uploaded by himself.



3.4 Category Management

Involving the sharer only, this functional goal allows him to manage the category set within the family. Since all the categories are editable by every family member (whether they are the initial creator of the category), the goals adding and deleting categories supports the one.



3.4.1 Add Category

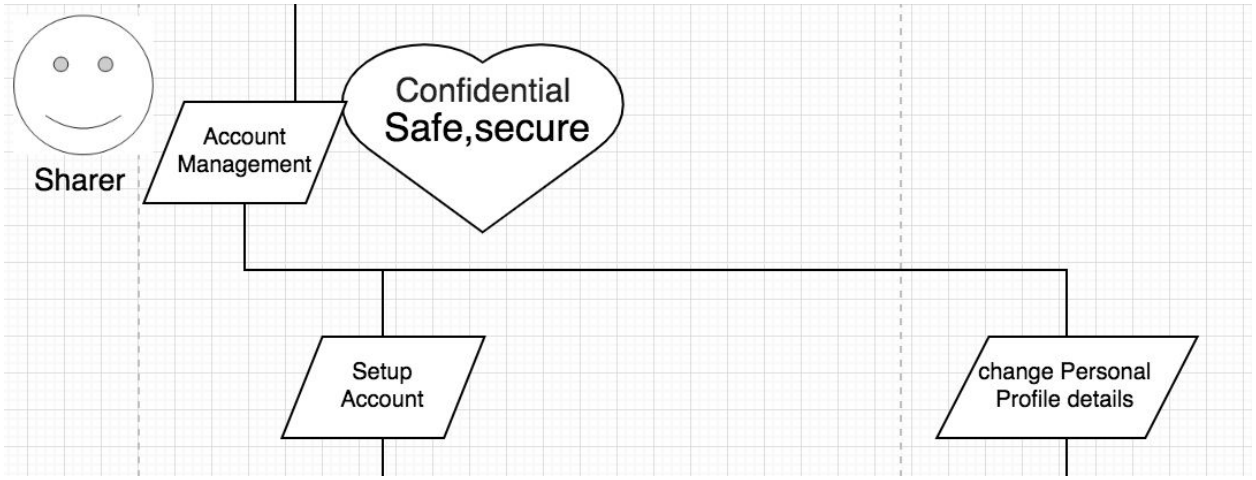
It is a functional goal that allows the sharer to add a new category when the limit of ones are not exceeding the maximum quotas, supported with 2 detailed functional goals. One is to set the name of the category, the other is to choose the background color for the category being created.

3.4.2 Delete Category

This functional goal allows the sharer to delete any category that exists within his family.

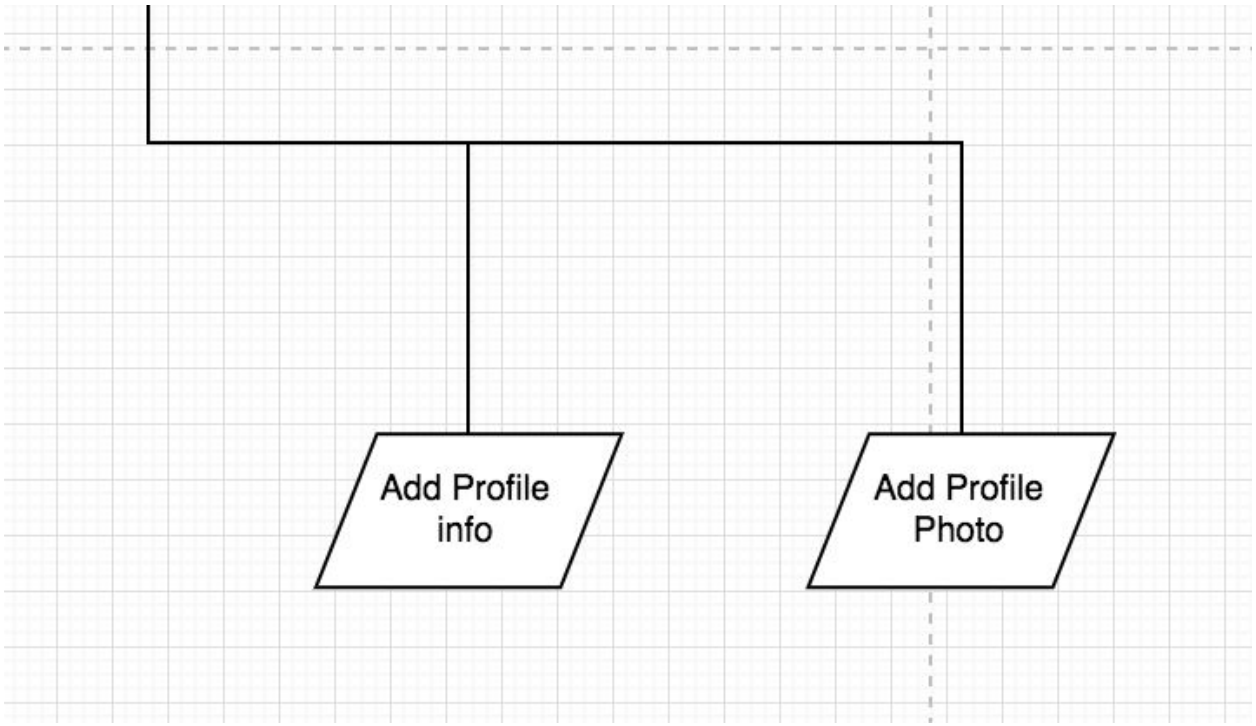
3.5 Account Management

This is the functional goal that involves the sharer only. It contributes to allow a confidential, safe and secure experience for the sharer when managing his account. To achieve this, there are 2 functional goals to support it: account setup and personal profile editing.



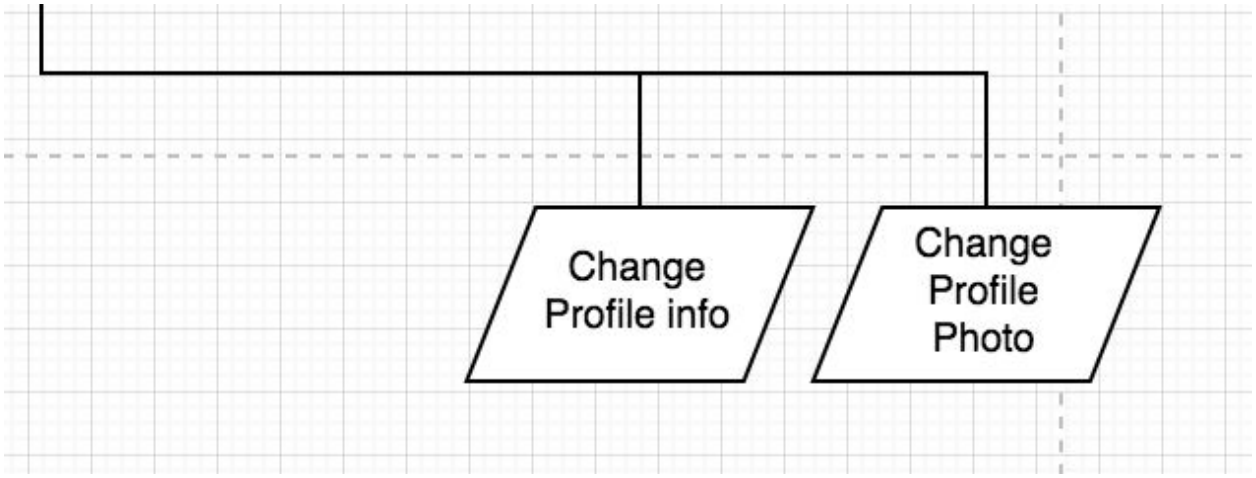
3.5.1 Setup Account

It is a functional goal to allow the sharer build their own account when using the app, and store the data with the account recorded in the database system. It involves 2 functional goals: the one that allows the sharer to add their profile information when creating the account, and the goal that makes the sharer available to upload profile photo to the new account.



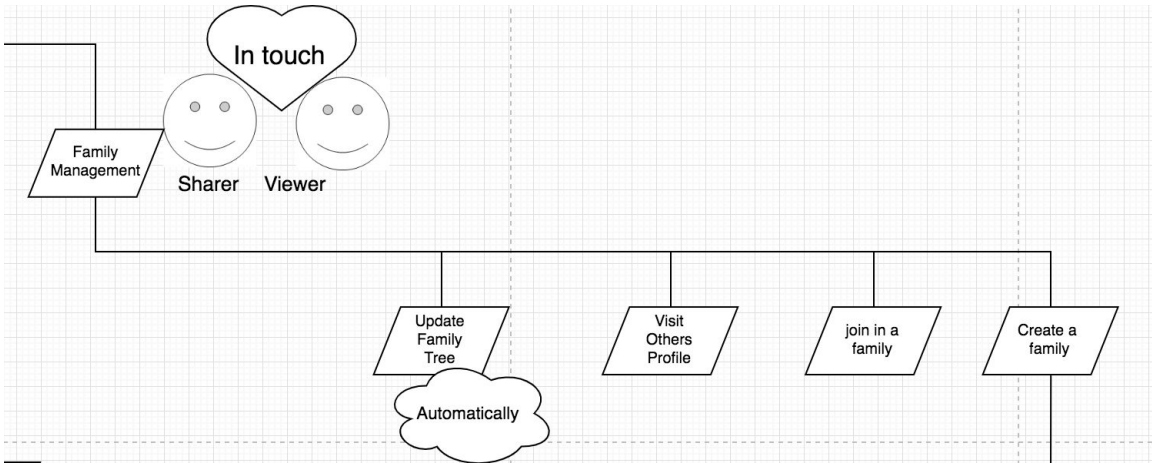
3.5.2 Change Personal Profile details

This functional goal allows the sharer to edit the personal profile that is set up when creating the account. With similar goals to 3.5.1, the sharer could change both of the profile information, and upload a new profile photo for his existing account.



3.6 Family Management

This is the functional goal that involves both the sharer and the viewer, that expects they could have a get-in-touch experience when managing their family system within the app. For the support of achievement of the goal, it includes the goals that updating family tree, visiting others profile, joining in a family and creating a family.



3.6.1 Update Family Tree

This is the functional goal that is automatically done within the system. When the members of the family change (including the role changes, new members and death of existing members),

the family tree within the tree page of the app would be modified itself automatically, to achieve a proper management of the family system.

3.6.2 Visit Others Profile

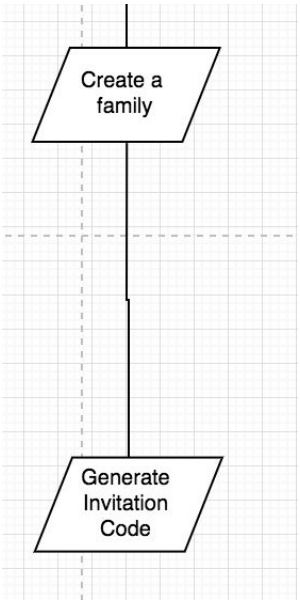
It allows the members within the family to visit the profile page of other members, while they could not be able to see everything nor edit any information within that profile page that is not belonging to the visitor himself.

3.6.3 Join in a family

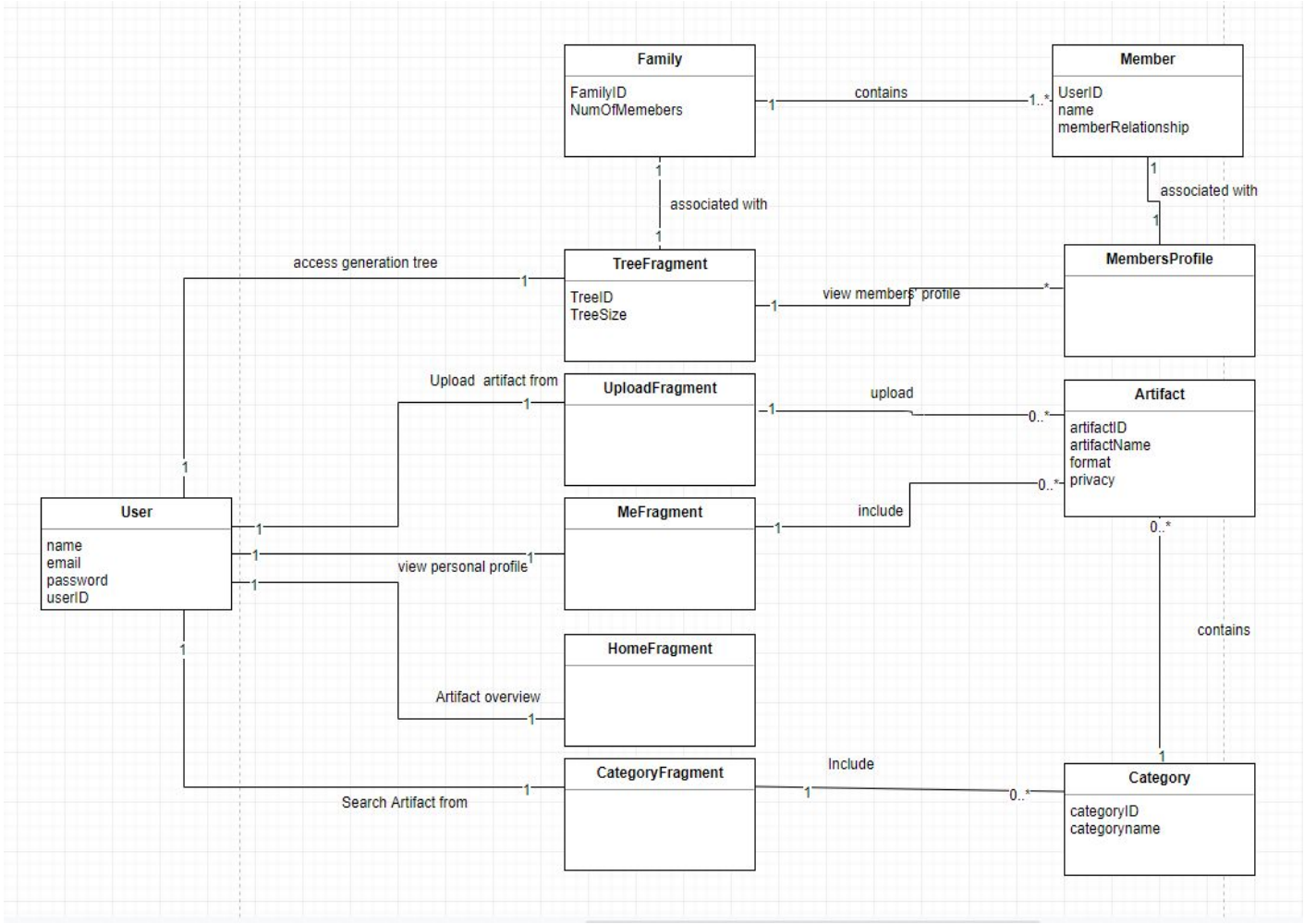
This is a functional goal that allows a new account holder to join in an existing family recorded in the database. An invitation code of the family would be required to get in the correct family.

3.6.4 Create a family

This is a functional goal that allows a new account holder to create their own family system in the app. The new information would be automatically recorded in the database, and a unique invitation code that stands for the family created at the moment would be generated for other family members to join in with (refers to 3.6.3).



4. Domain Class Diagram

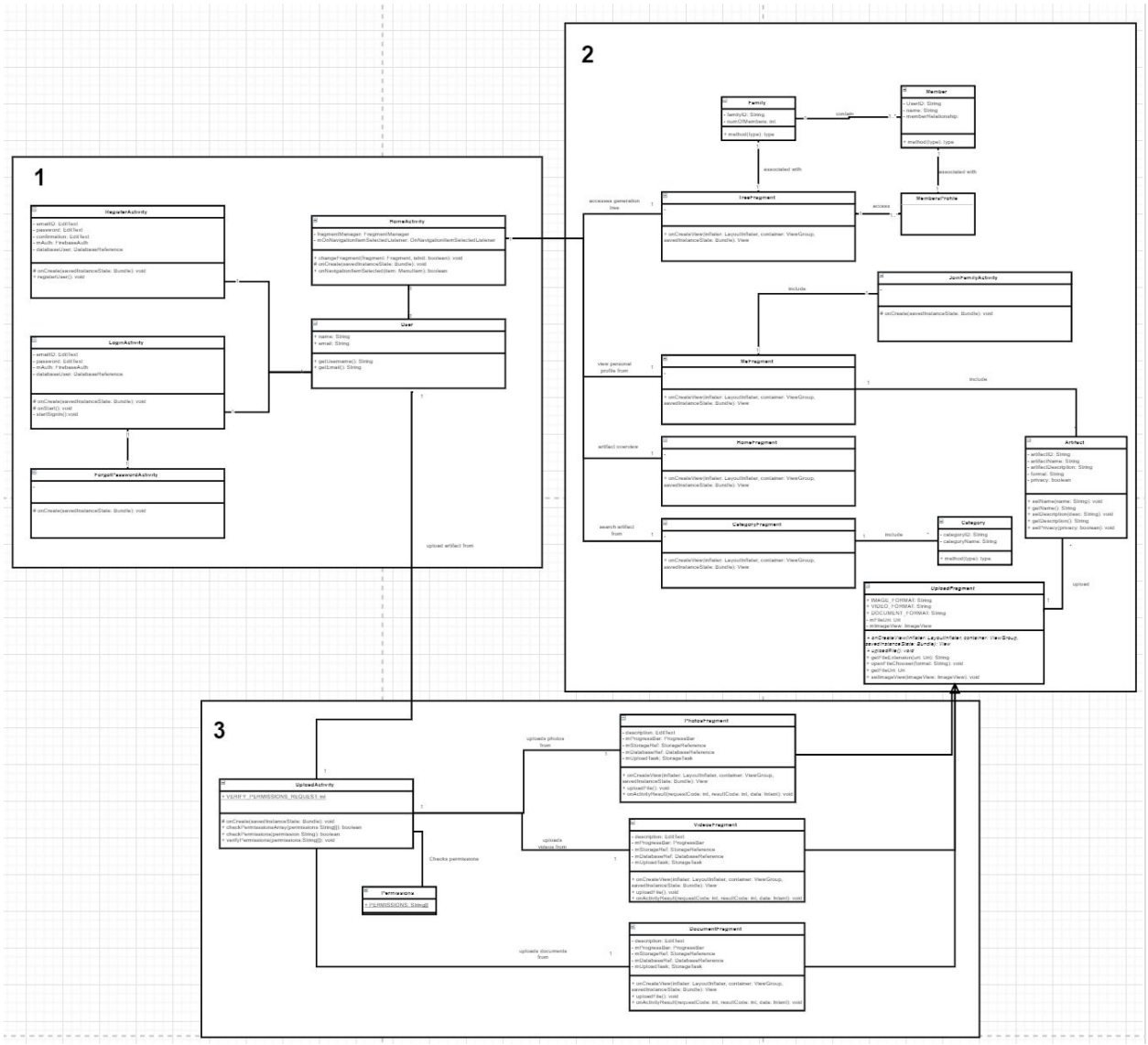


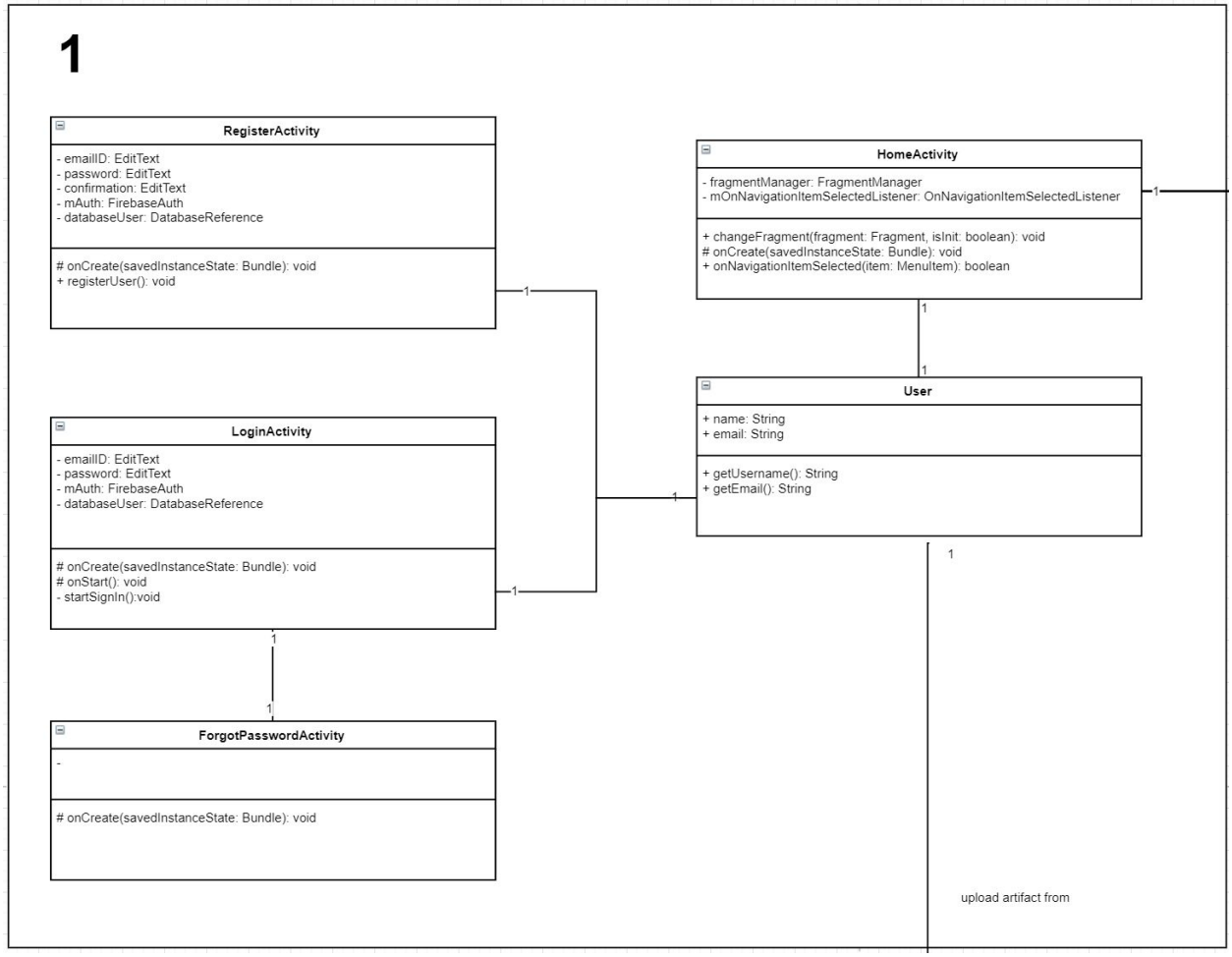
Link:
https://drive.google.com/open?id=1heM7FRTwww1kTUVl6Vi_Pyj0rn5Olutk

Description: This is the basic domain class diagram of our application. In this diagram we have a User class, a User is a person that can do the actions in the APP for example visit others profile, upload artifacts,view personal profile and search for the artifacts. In addition, the user must has his name, email, password and the userID as his basic information.
TreeFragment, UploadFragment, MeFragment, HomeFragment and CategoryFragment are included in our Homepage and can be accessed by clicking the buttons on the Homepage. All the functionalities are achieved through these 5 fragments.
A family tree would be display in the TreeFragment, so in this class the TreeID and TreeSize is indispensable. One family tree can only include the people in one family, the number of people in the family will influence the tree size directly.
In the uploadFragment, the user can choose whether or not to upload the artifacts. As a result, in the artifact class, it needs to include the attributes: artifactID, artifactName, the format(photo,

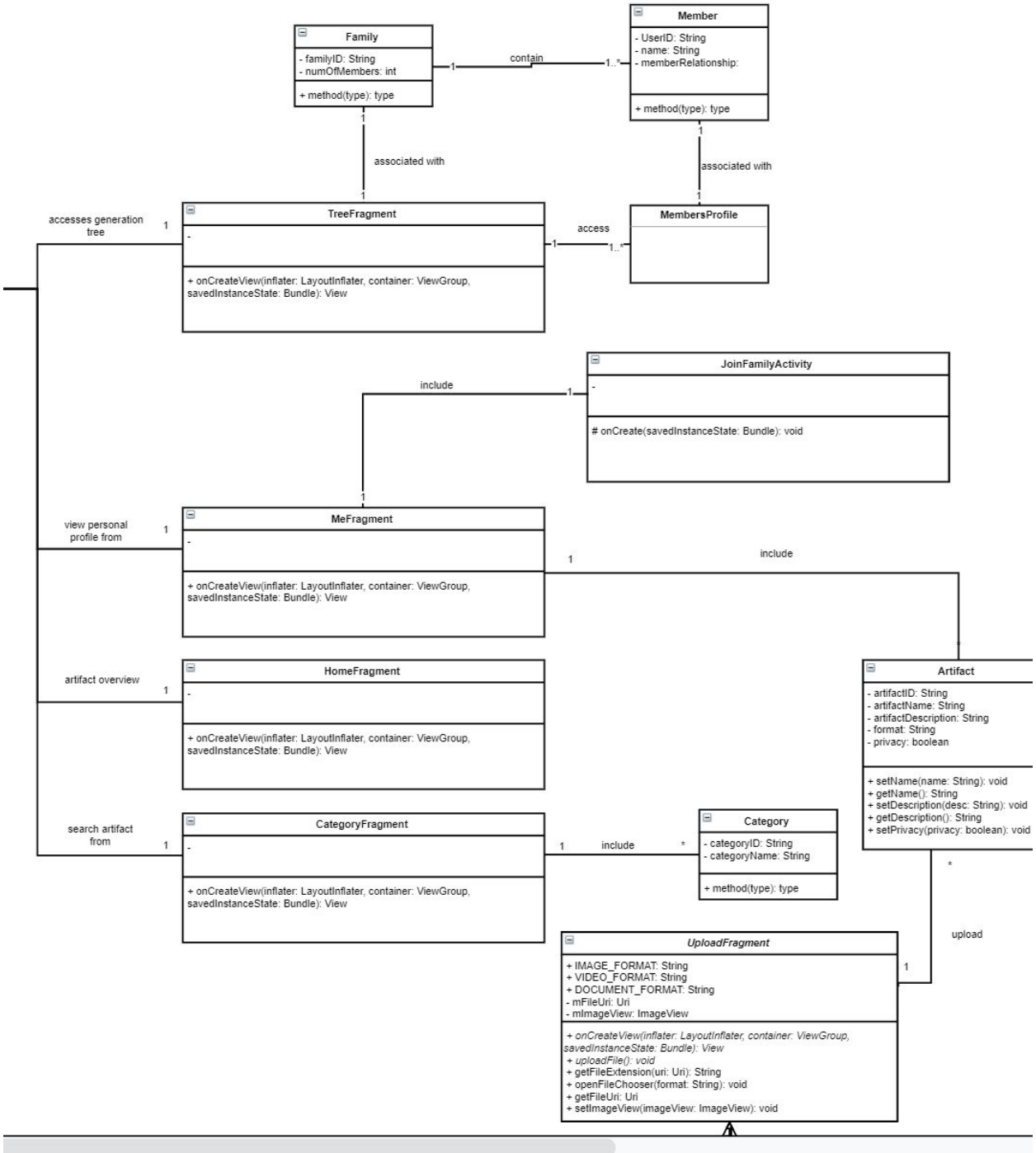
document or video), and privacy (public or private). At the same time, the artifacts may have a category to describe them. The category is listing in the categoryFragment which need to have the catehoryID and the categoryName.

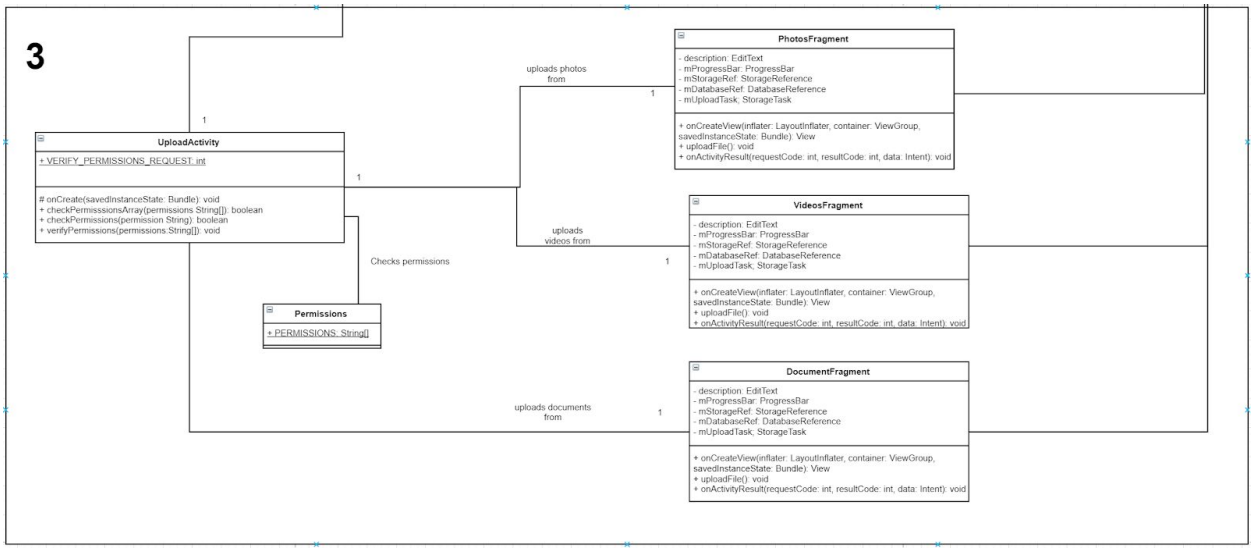
5. Design Class Diagram




1
RegisterActivity
- emailID: EditText
- password: EditText
- confirmation: EditText
- mAuth: FirebaseAuth
- databaseUser: DatabaseReference
onCreate(savedInstanceState: Bundle): void
+ registerUser(): void
HomeActivity
- fragmentManager: FragmentManager
- mOnNavigationItemSelectedListener: OnNavigationItemSelectedListener
+ changeFragment(fragment: Fragment, isInit: boolean): void
# onCreate(savedInstanceState: Bundle): void
+ onNavigationItemSelected(item: MenuItem): boolean
User
+ name: String
+ email: String
+ getUsername(): String
+ getEmail(): String
LoginActivity
- emailID: EditText
- password: EditText
- mAuth: FirebaseAuth
- databaseUser: DatabaseReference
onCreate(savedInstanceState: Bundle): void
onStart(): void
- startSignIn():void
ForgotPasswordActivity
-
onCreate(savedInstanceState: Bundle): void
upload artifact from

2


Family
- familyID: String
- numOfMembers: int
+ method(type): type
contain
1
1..*
Member
- UserID: String
- name: String
- memberRelationship:
+ method(type): type
associated with
associated with
accesses generation tree
TreeFragment
-
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
access
MembersProfile
include
JoinFamilyActivity
-
onCreate(savedInstanceState: Bundle): void
view personal profile from
MeFragment
-
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
include
artifact overview
HomeFragment
-
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
search artifact from
CategoryFragment
-
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
include
Category
- categoryID: String
- categoryName: String
+ method(type): type
Artifact
- artifactID: String
- artifactName: String
- artifactDescription: String
- format: String
- privacy: boolean
+ setName(name: String): void
+ getName(): String
+ setDescription(desc: String): void
+ getDescription(): String
+ setPrivacy(privacy: boolean): void
upload
UploadFragment
+ IMAGE_FORMAT: String
+ VIDEO_FORMAT: String
+ DOCUMENT_FORMAT: String
- mFileUri: Uri
- mImageView: ImageView
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
+ uploadFile(): void
+ getFileExtension(uri: Uri): String
+ openFileChooser(format: String): void
+ getFileUri: Uri
+ setImageView(imageView: ImageView): void



Link:
https://drive.google.com/open?id=1U_NueO8p5sP9OAZRt5dHGjgQzr9EBtyL

5.1 Register Activity

This class is responsible for the registration of the new users. With the method “registerUser”, it stores the email and confirmed password into the database.

5.2 Login Activity

This class has the same attributes with the Register Activity, excluding the confirm of the password. It is responsible for the matching with the typing information and the ones on the database, with the function startSignin. Once the account has matched, the system directs the person to the specific pages where the information of that account has loaded.

5.3 Forgot password Activity

This is the class for the reset of the password of any existing account in the database. With the attribute email and the function sendResetEmail, it could resend the email to the user for the password reset.

5.4 User

This is the class that represents the user with attributes name and email. Both register and login activities would only direct to one user, and one user would respectively have one Home activity and one Upload activity for further actions in the app.

5.5 Home Activity

This is the component that is responsible for the pages changing of the app, including 4 fragment classes: the TreeFragment class (for the access of generation tree), the MeFragment class (for the visit of personal profile), the HomeFragment class (illustration of latest artifacts from the whole family) and the CategoryFragment class (for the searching of specific types of artifacts).

5.6 Upload Activity

This component we have the UploadActivity class which is responsible for requesting permission to access sensitive data or system features like storage, a list of permissions is stored in the Permissions class. This class includes three different fragment classes: the PhotoFragment class, VideoFragment class, and DocumentFragment Class. All three classes inherit from an abstract class UploadFragment which is responsible for selecting the artifact from the phone through the openFileChooser method. The method uploadFile in UploadFragment is responsible for uploading the artifact to the backend, and storing the artifact information to the database.

5.7 Join Family Activity

It is the class for the joining of the family with the unique invitation code generated at the beginning of the creation of a family. It requests permission to access the database to match the invitation code that the user typed in, and allows the addition of the profile information of the user into that family if the code matches.

5.8 Artifact

This is the class that represents the artifact information, being included from the MeFragment class. With the attributes artifactID, artifactName, artifactDescription, format and privacy, the details of an artifact would be recorded and stored as private ones. By calling the according functions, the name, description and the privacy would be set and obtained for any further actions.

5.9 Category

This class is included by the CategoryFragment class, representing the category subject. Each of such has a categoryID and a categoryName, which both store the related information for the reuse or recall from the other pages in the app.

5.10 Family & Member

This is a class that stands for the family subject. Each family has a unique familyID, and the statistics of the family members. It contains a Member class, which represents the member

subject. Each of them has a UserID, name for the identity. Moreover, one attribute called memberRelationship illustrates the basic relationship with the immediate family.