Team Name: Liquid

Team Members:

1. Chenghong Zhu
2. Erya Wen
3. Lehan Gong
4. Gilbert Vincenta

Supervisor: Geoffrey

Client: Stella Angelita (Gilbert's sister-in-law)

(**Private**) Repository link:
https://github.com/lehang123/liquid

Table of Content

PART 1 : About The Project

In this IT project, we develop a family register app based on our client's requirements under our supervisor's guidance. We started off with finding a suitable client for our project. Next, contacted our client to find out the requirements. From these requirements, then we define our project's scope/features and architecture. We've documented the process of the whole project in this report.

PART 2 & 3 : Scope and Requirements

In week 1, we decided between Leon Sterling, our lecturer, as our client, or Stella Angelita, Gilbert's sister in law as our client. Since the theme is a family register app, Leon has recommended us to have our family members as our clients as we'd understand the context more if we do so. Thus, we decided to have Stella Angelita as our client.

Client's background:

Stella Angelita, Gilbert's sister-in-law, was chosen to be the client of the project. She is a housewife and a mother to 2 kids. One of her main hobbies is to travel around the world, and her family shares this hobbies together. As a result, she travels quite often with her family. During travelling, she and her family members takes a lot of family pictures (and videos). They would like to share these memories together and pass these on to their kids.

On technologies used, she uses both laptop as well as an iOS device. Majority of the family also uses iOS devices and are frequently using their phones rather than desktop pc. So, it was decided that an iOS app would be appropriate for the family.

Client's Requirements(User Story):

After choosing Stella Angelita as our client, we proceed to gathering requirements of the product from her through a voice call with the client. this section outlines what the client required from the app (described in a do - be - feel user stories) :

1. As a user, I have my own account to manage my family's photos.
2. As a user, I can create albums with some descriptions as a way to manage my family's photos.
3. As a user, I can add a location to each album so that I can recall where the pictures within the album were taken from.
4. As a user, I can assign photos with some descriptions into albums to manage my photos and record the memory.
5. As a user, with a few other family members, I can join my family in the app, so that we can share memories together.
6. As a user, within a family, all albums created are automatically shared to other family members.

7. As a user, I can comment on photos to share my thoughts and interact with other users in the family.
8. As a user, for each photo, I can see how many family members have liked and have seen it, to track how interesting the photo is (i.e. whether the photo should be kept or deleted if it's not really interesting).
9. As a user, for each photo, I can see it in full screen, to review the photo quality.
10. As a user, besides photos, videos can also be added and viewed later, to capture different memories.
11. As a user, i can view other family members' name, date of birth and position in the family. This helps to primarily remind me of their birthdays.

Be -non- functional requirements:

1. Easy UI interactions.
2. Simple navigations.
3. Localised dates.

Feel - emotional requirements:

1. Informal
2. Reflective
3. Fun

App Features (Scope):

After getting clients' requirements, we proceeded to discuss and design the app's features for a few days. When designing our app features, we also researched over some useful photo management resources. Some app features we got were inspired from WeChat (messenger app) as well as Instagram (photo management app), as we frequently use them and our client and her family are familiar with Instagram. We have also provided a Goal Model diagram to give an overview of the features of the system. The features are finalised as follows:

1. Create an account to manage family photos.
    a. As a user, i can have my own account to manage all my albums.
        i. New user creates his/her own account.
        ii. Existing user can log in with email & password.
        iii. Recommended for client for extra security: password must be at least 8 characters long for better password quality.
    b. For more personalisation in the app, we store a few user details that are editable :
        i. DOB - in DD-MM-YYYY format (localisation, be #3)
        ii. Username
        iii. Password
        iv. Gender
        v. Position / relationship in the family
        vi. Profile picture

2. User's family (addressing Do#5) :
    a. Each user can connect with other family members in existing family.
    b. A user can join a family or create his/her own so he/she collaborate with his/her families.
    c. For more personalisation in the app, we store a few family details that are editable :
        i. Motto
        ii. Profile picture (for family)

3. Album and pictures:
    a. To begin with, as a user of a family, i can make album (with name and description), so that we can see each other's artifacts. These albums would be automatically shared to all of my family members. (addressing do#2 and do#6).
    b. Later on, user can see the list of albums that the family has created.
    c. The list of albums is sorted chronologically, newest - oldest, to help user locate their albums.
    d. After creating an album, user can now upload photos into albums. (addressing do#4).
    e. As a user checking an album, he/she can see all pictures associated.
    f. It is possible to view photos in full screen mode, to review the picture quality. (addressing do#9)
    g. Each family member can like and comment each picture to enhance personalisation and interactivity of the album (addressing do#7 and do#8).
    h. Besides photos, user can upload videos (and watch them later) into the album too, to store different kinds of memories (addressing do#10).
    i. When opening an album's content, users can see videos' thumbnail to indicate that videos have been stored in that album.
    j. As a user that travels a lot, he/she can add real-time location to the album (addressing do#3).

4. Family views (addressing do#11):
    a. As a part of a family, he/she can see all members of his/her family in the app, including their usernames, date of birth and relationship (e.g. mom, dad, daughter, son, etc) in the family.

PART 4 : The Team

Team's related past experiences:

- Gilbert and Erya have taken a web development course.
- Erya moved from design faculty, so she is knowledgeable on UI prototyping and development.
- Lehan has previously worked for a mobile dev start up.
- Chenghong knows a lot about object oriented principles.

Managing roles (i.e. Task assignment)

On deciding roles, everyone just naturally fits into a role based on their past experiences. During this time, we already know that we are going to build an iOS app with firebase as its backend already. So, as Lehan has previously worked on a mobile development project, he would know more on what needs to be done. Thus, he is entitled as the project manager.

In the previous web development course, Gilbert mostly worked on the backend (in MongoDB) and integrating the backend to frontend itself. On the other hand, Erya mostly focused on the frontend. So, they are fit into the same role again this time.

As the project entails the need to store pictures for families, we decided to use firebase's authentication, database, and storage services. Now, as for the roles, Since Gilbert has taken the database management role, so Chenghong decided to take the authentication (i.e. users) management, and Lehan took the storage management.

TL;DR

1. Lehan: project leader, storage, Coding App Structures.
2. Erya: UI design, Coding App Structures, app logic.
3. Chenghong: app logic, Coding App Structures, drawing diagrams.
4. Gilbert: database, documentation

Team logistics

1. We initially started communicating with slack.
2. We then migrated the communication towards Facebook Messenger's group conversation tool.
3. We maintained a google drive folder for any documentation purposes.
4. As all members share a lot of subjects in common, the members meet on (almost) daily basis, i.e. 3-4 days per week.
5. Every beginning of the week we assign a few tasks for each member to be completed by the end of the week.

PART 5 : Architecture

Front-end : Swift (as we're building an iOS app).
Backend : Firebase : Auth +Firestore (Database) + Storage services.

User use their phones and touch on the buttons on each view controller. Once they click the button, it will send a request to our backend service. Then it will based on the request search for information in database and give back all the information that the user needed.

Architecture Diagram is shown in appendix.

PART 6 : Sprint Artifacts

PART 6 A: Sprint 1

Overview

In this sprint 1 (last between week 1 to mid of week 5), we focused mostly on client's requirements and defining member roles. By the end of the sprint, we've picked a suitable platform to work on. We've even established our backend and our frontend and started a bit of coding, which is actually ahead of our timeline.

What We've done

1. Communicated with client and extracted client's requirements.
2. Decided what each member's generally responsible for.
3. Explored on backend and frontend platforms that we can use.
4. Initiated backend and frontend connections.
5. Created basic structure of this app
6. Finished storage, authorisation.
7. Finished user log in component
8. Created fundamental UI

What we'll do next

1. UI planning (and possibly prototyping).
2. Software architecture planning.
3. Building basic functionalities based on given requirements.

PART 6 B: Sprint 2

Overview

In this sprint 2 (last between week 5 up to the break week), we focused mostly on building the basic functionalities required by our client. By the end of the sprint, we got our functionalities done, with the UI connected with the backend as well.

What We've done

1. UI planning and implementation.
2. Database structuring and implementation.
3. Storage management.
4. User authorisation management.
5. Backend : Database & authorisation & storage integrated for the app.
6. Frontend & Backend integration over basic functionalities.

7. Documented about :
    a. User stories
    b. Architecture diagram
    c. Database structure diagram
    d. Code & comments (with Swift's built-in documentation)

What we'll do next

1. Polishing UI (add effects and better layouts).
2. Adding more capabilities/functionalities on the app.
3. Backend & UI Testing.

PART 6 C: Sprint 3

Overview

In this sprint 3 (last between week 10 up to week 12) we improve our app functionalities, executed some app tests and prepared for our presentation on week 12. By the end of the sprint, we finished improving functionalities (mainly: audio & video support, and adding location in) , executed some tests, and presented our app to the class. We've also contacted the client again to get feedbacks on how they feel about the app. The client gave an overall positive feedback and was amazed by the amount of functionalities we've offered in the app.

What We've done

1. Enhancing UI.
2. Database additional structuring and implementation.
3. Improve storage management to support more functionalities.
4. Backend : Database & storage integrated for the app.
5. Frontend & Backend integration over added functionalities.
6. Presentation rehearsal & demo.
7. Testing plan & execution on:
    a. Backend testing.
    b. UI testing.
8. Documented on:
    a. Team report
    b. Individual report
9. Contacted client for product feedback.
10. Wrapping up the report & app for final submission.

PART 7 : Process Guide

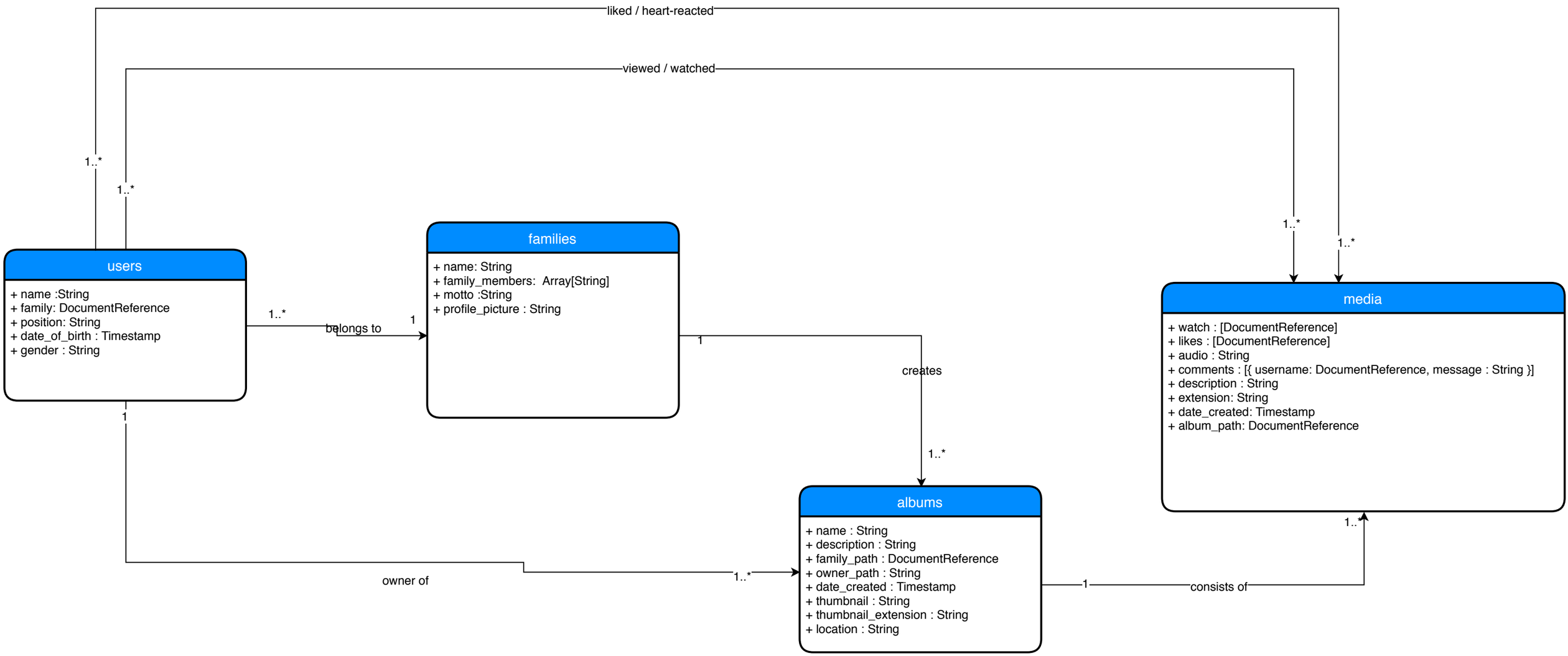
PART 7 A : Github management plan

We use sourcetree to manage all the commits and branches in github. Normally sourcetree will automatically create one new branch once another contributor push the project. Then it will also help the group merge the branches once another contributor pull the project. Overall, the sourcetree help us keep track of what is everyone doing and what we have done in the project in github.

PART 7 B : Test plan

All functionalities were all manually tested by the authors. Once the contributor pushed the code to github, they are required to verify the correctness of their programs.

It is possible to execute automated testing on frontend. Once written, the unit test will automatically run the test case to make sure our format is correct. We decided to send the product to client to have them test our app in general as well. Client was happy with the product and there were no notice-able UI / backend bugs when they use it on daily basis.



Database structure

Part 8: Appendix

1..*

(even more)
Architecture Diagrams