UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HA NOI

FACEBOOK CLIENT

Project report

MOBILE APPLICATION DEVELOPMENT

Đặng Thái Sơn BA9-053
Nguyễn Minh Hiếu BA9-020
Đào Hải Long BA9-041
Đỗ Công Hòa BA9-022
Đoàn Văn Chương BA9-008

November 17, 2021

Contents

1 ABSTRACT

This is group 4's report of the final project, which makes up 20% of our total results of the "Mobile application development" course. This course, including the most fundamental parts that we should know about a Mobile application namely: Activities, Fragments, Threads and Networking..., was led by the enthusiasm of Dr.Tran Giang Son.

In this report, we will explain carefully about the program that our team presented in the Presentation and Demo part on Wednesday 10/11/2021: Introduction about the program; The methods we used; and The main results we got.

Thank you, Dr. Tran Giang Son, for giving us the chance to access so much valuable knowledge and precious practical sessions about the Mobile application.

2 INTRODUCTION

2.1 What is the Facebook client?

To begin with, Facebook is a social networking website where users can post comments, share pictures, post links to news or other interesting content, chat live, and watch short-form videos.... It was established in 2004 by Mark Zuckerberg and Edward Saverin as a school-based social network at Harvard University. Until now, it has become one of the most popular social media in the world and gained billions of users thanks to its convenience and simplicity.

Facebook client is a stand-alone application running on clients' machines. It utilizes the core technologies of the Facebook platform to integrate Facebook's News Feed, Notifications and other feature for the users.

In this project, our group tried to make a fundamental and primitive Facebook client.

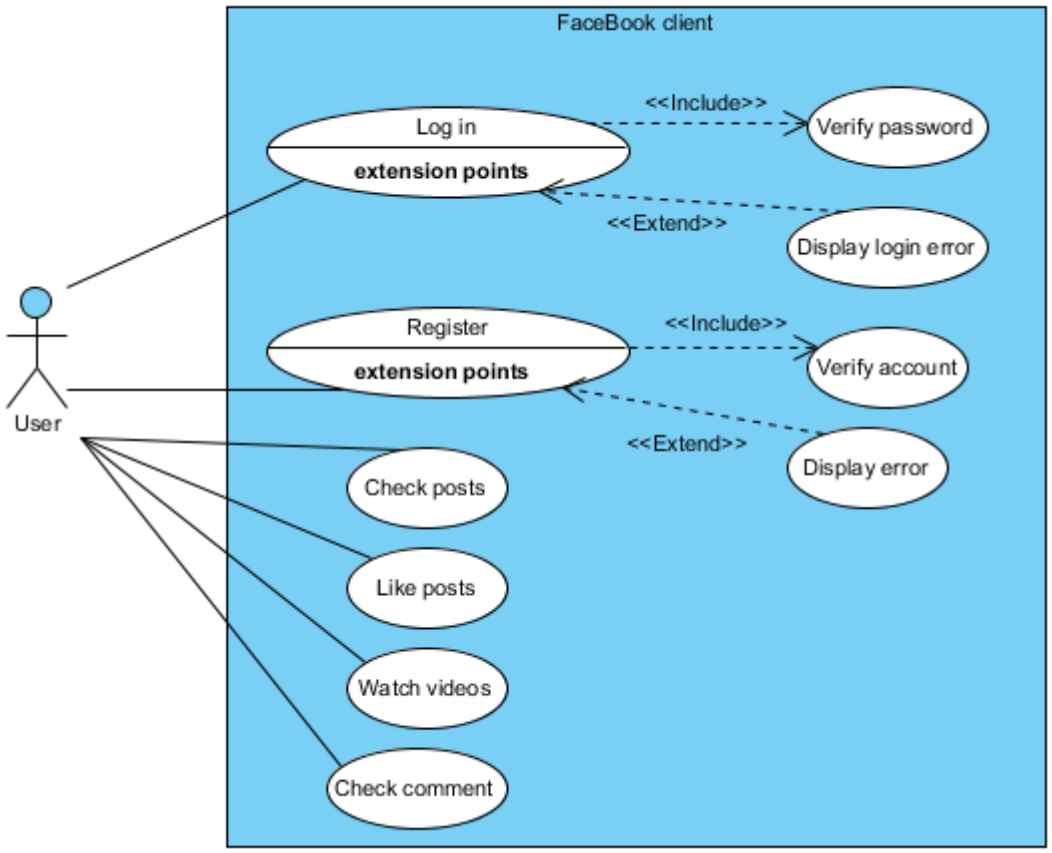
2.2 Why do people need our application?

In our opinion, we think that the Facebook client our group created could benefit users in many ways. Firstly, the most obvious reason is that this application bases on the Facebook platform, so basically, it can be considered as a tool to connect to the social network where people can perform tasks such as reading friends' status, watching short videos, commenting... Moreover, our program provides a convenient and easy way of interaction with Facebook. Due to the fact that this is a mobile application, people can access to the social network anywhere just by their phones.

3 METHODS

3.1 Use cases

To summarize the details of our application's users and their interactions with the system, we created the following UML use case diagram:



3.2 Components

3.2.1 Activities

There are six main types of activities with different functions in our project:

1. **Login/ Register activity**: provides people a way to create new accounts or access to the application if they already have ones.

2. **Main activity**: this is the default activity when users sign in. It is used to connect five main fragments: Homepage, Profile, Facebook Watch, Notifications and Menu.

3. **Zoom-in activities**: help users to watch full-screen videos and photos when needed.

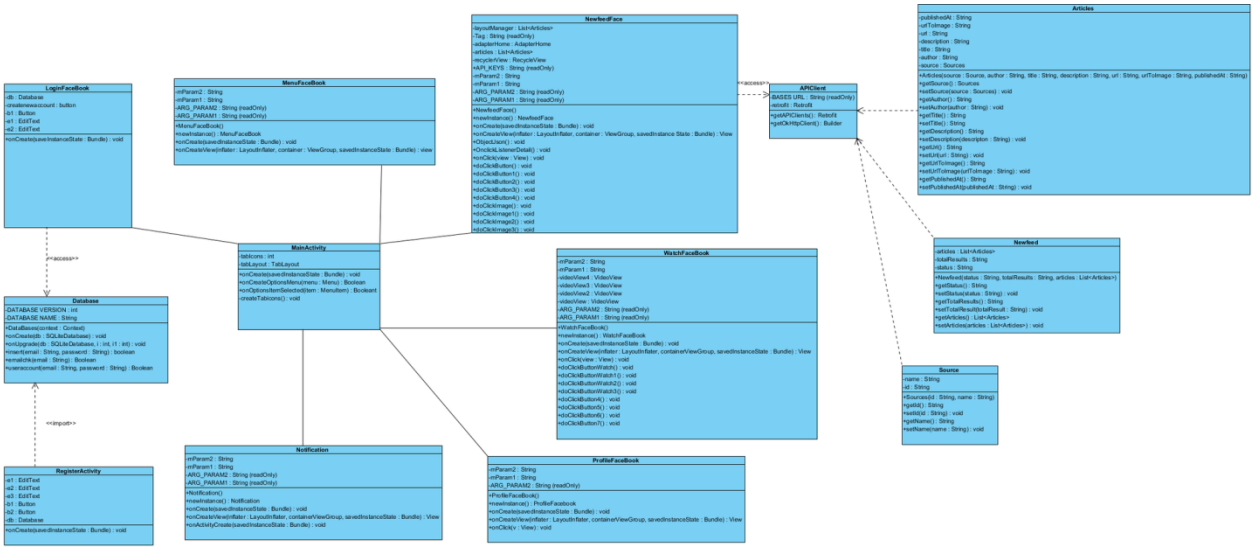
4. **Notification detail activities**: support people to see further information of the notifications.

5. **Comment activities**: provide the user discussion about specific posts.

6. **Messenger activity**: stimulates the chatting section.

3.2.2 Class diagram

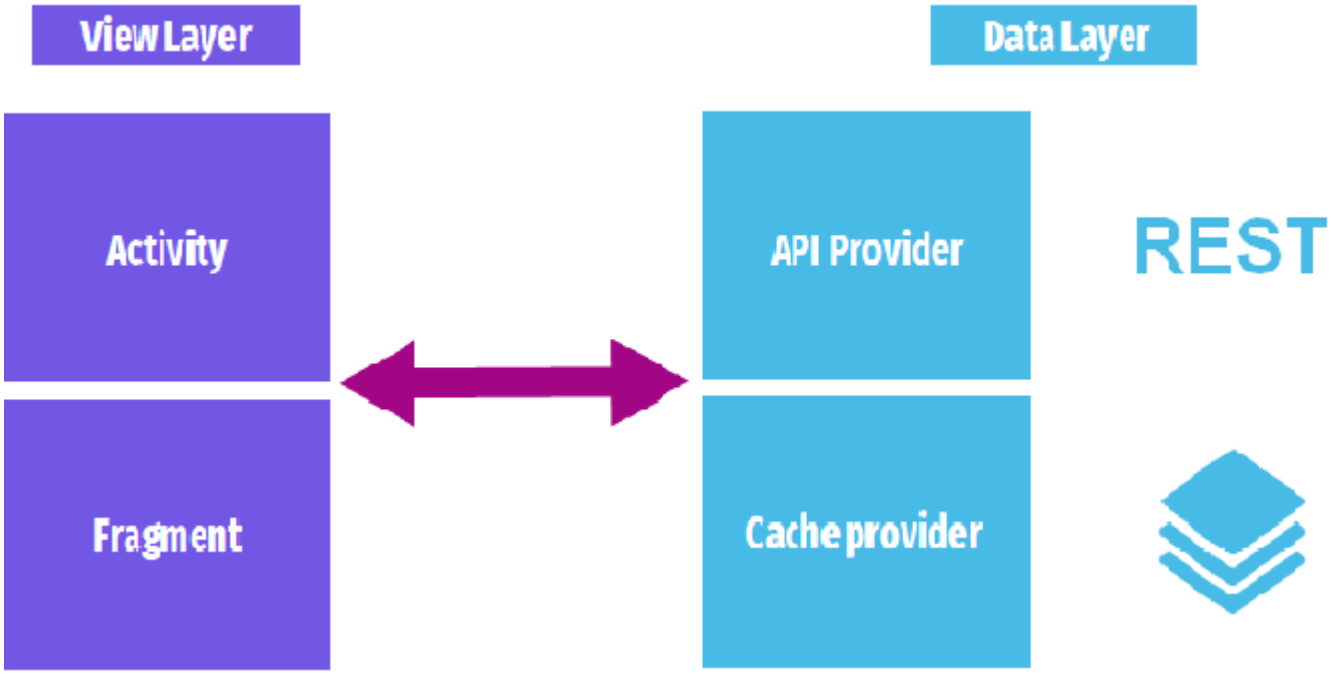
Below is the UML class diagram of the application to provide further information about those activities' classes, their attributes, operations, and the relationships among objects:



3.3 Architecture

3.3.1 Layers

In order to achieve the use cases we listed above, our application must have two main layers:



- The first one is the VIEW LAYER which contains all the activities and fragments that users can see and interact with.
- The second one is the DATA LAYER. This layer is responsible for getting the data from remote data source using Representational State Transfer protocol and also from the database in the device cache.

These two layers connect to each other through the request-respond relationship: users request data from the VIEW LAYER, then the DATA LAYER prepares and returns the needed information.

3.3.2 Sequence diagram

To be more specific about how operations are carried out in our application, here is the system's UML sequence diagram (Device is the VIEW LAYER; Account Database and Server are the DATA LAYER):

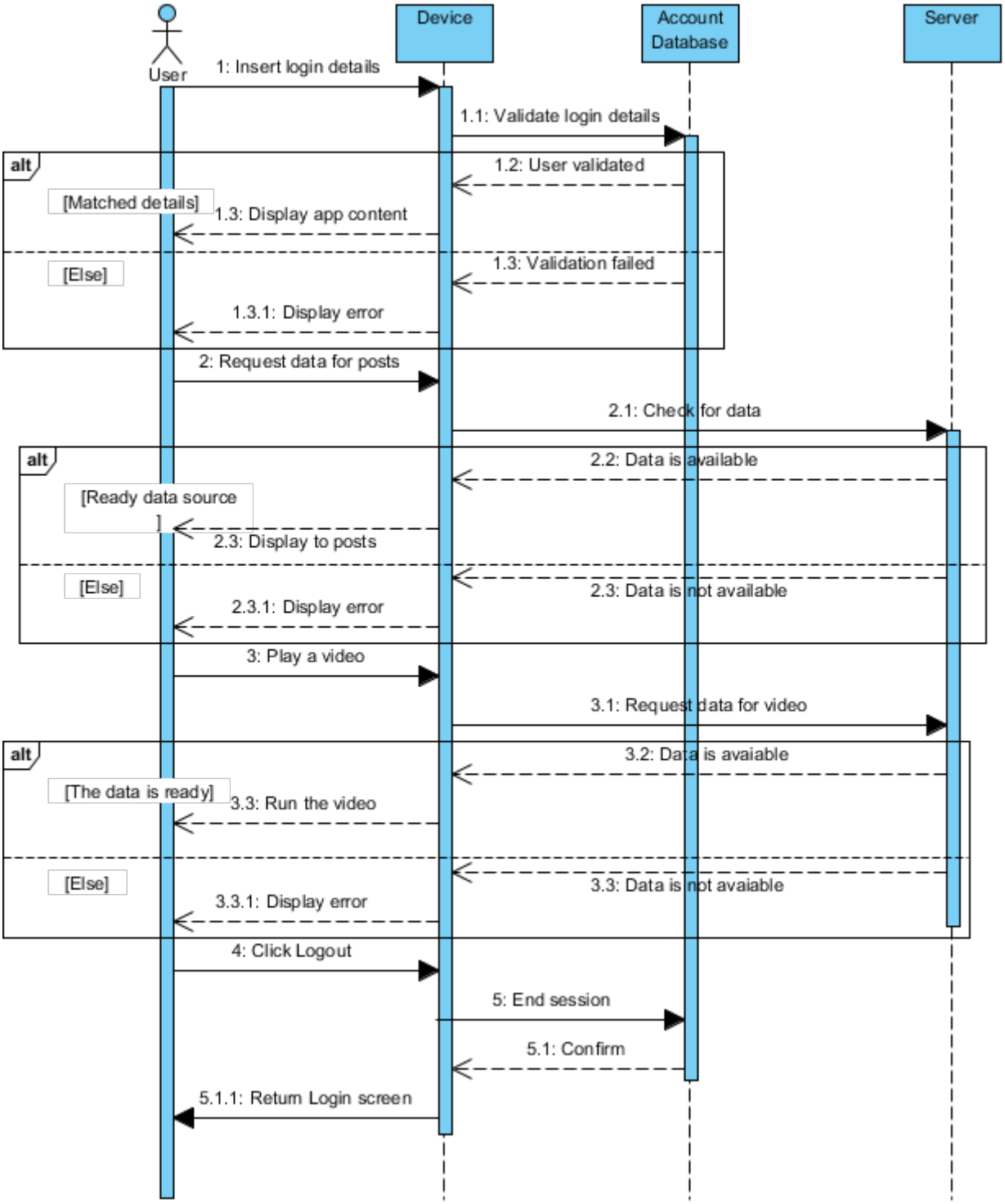


Figure 1: Case when the user already has an account

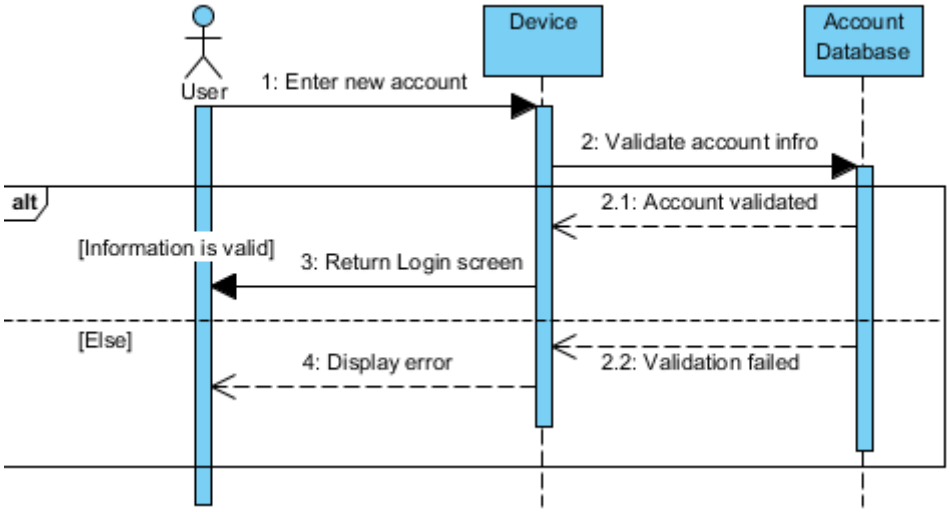


Figure 2: Case when the user registers a new account. Then login the same as Figure 1.

3.4 Networking

3.4.1 News feed

We created a post connected to "The US business news site" using openAPI with Representational State Transfer protocol (REST):

Step 1: We made an API URL including three main parts to request data from the server:

- **Domain**: "https://newsapi.org/v2"
- **Attribute**: to get data from the domain: "top-headlines"
- **API key**: to track and control how API is called: "23d4ed0e5fe84741ac530eb21b6ecc8a"

Below is the need chunks of code to request and format API respectively:

```
public static Retrofit retrofit;
public static final String BASES_URL="https://newsapi.org/v2/";
public static Retrofit getAPIClients() {
if (retrofit==null){
    retrofit=new Retrofit.Builder()
                .baseUrl(BASES_URL)
```

```
                .client(getOkHttpClient().build())
                .addConverterFactory(GsonConverterFactory.create()).build();
    }
return retrofit;
}
```

```
public interface APIKey {
    @GET("top-headlines")
        Call<Newfeed>getNewfeed(
    @Query("country") String country,
    @Query("apiKey") String apiKey);
}
```

Step 2: When the request is sent, the server will verify it and look for the appropriate resources to create the content returned accordingly.

Step 3: When the verification is done, the server will return the result in JSON or XML format via HTTP or HTTPS.

Step 4: Finally, when the application receives the data, it will display on the News feed screen. Below is the chunks of code to convert the status of an object into the byte string:

```
@SerializedName("source")
@Expose
private  Sources source;
@SerializedName("author")
@Expose
private  String author;
@SerializedName("title")
@Expose
```

```
9  private  String title;
10 @SerializedName("description")
11 @Expose
12 private  String description;
13 @SerializedName("url")
14 @Expose
15 private  String url;
16 @SerializedName("urlToImage")
17 @Expose
18 private  String urlToImage;
19 @SerializedName("publishedAt")
20 @Expose
21 private  String publishedAt;
```

```
1  @SerializedName("status")
2  @Expose
3  private  String status;
4  @SerializedName("totalResults")
5  @Expose
6  private  String totalResults;
7  @SerializedName("articles")
8  @Expose
9  private List<Articles>articles;
```

```
1  @SerializedName("id")
2  @Expose
3  private String id;
4  @SerializedName("name")
```

```
@Expose
private String name;
```

In addition, we used a few libraries to support the API calling:

- **Bumptech library**: Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide includes a flexible API that allows developers to plug in to almost any network stack.
- **Retrofit library**: supports developers who convert APIs into Java Interfaces to easily connect to a REST service on the web.
- **Retrofit-gson library**: a Java library used to convert Java objects into their JSON performances and vice versa.
- **PrettyTime library**: used to normalize, calculate, and format time.

3.4.2 Facebook watch

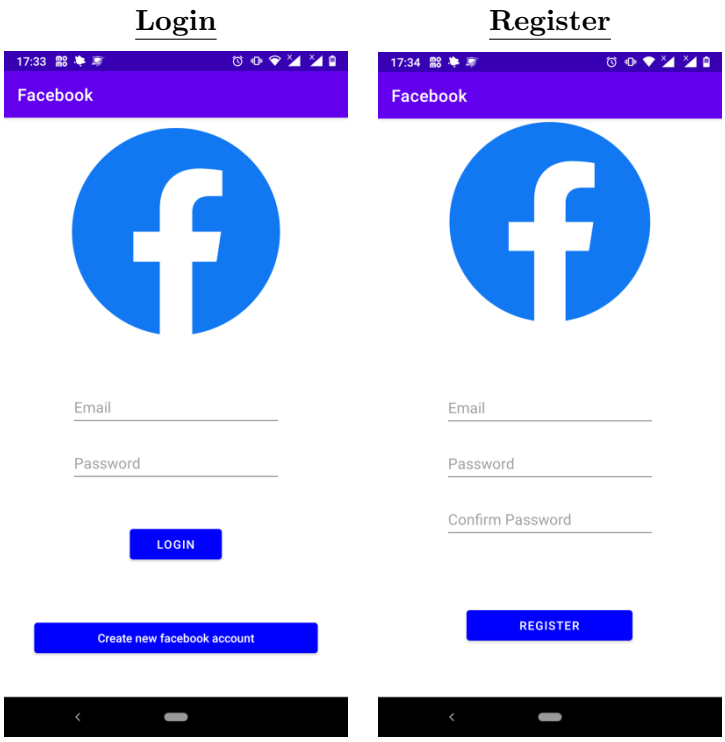
In this part, we used a stream server to upload videos. After that, we took the URL links from the server and called it back by using the embedded package. We also used "The Network Security Configuration" feature through an XML file in order to reach the needed links:

```
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

4 RESULTS

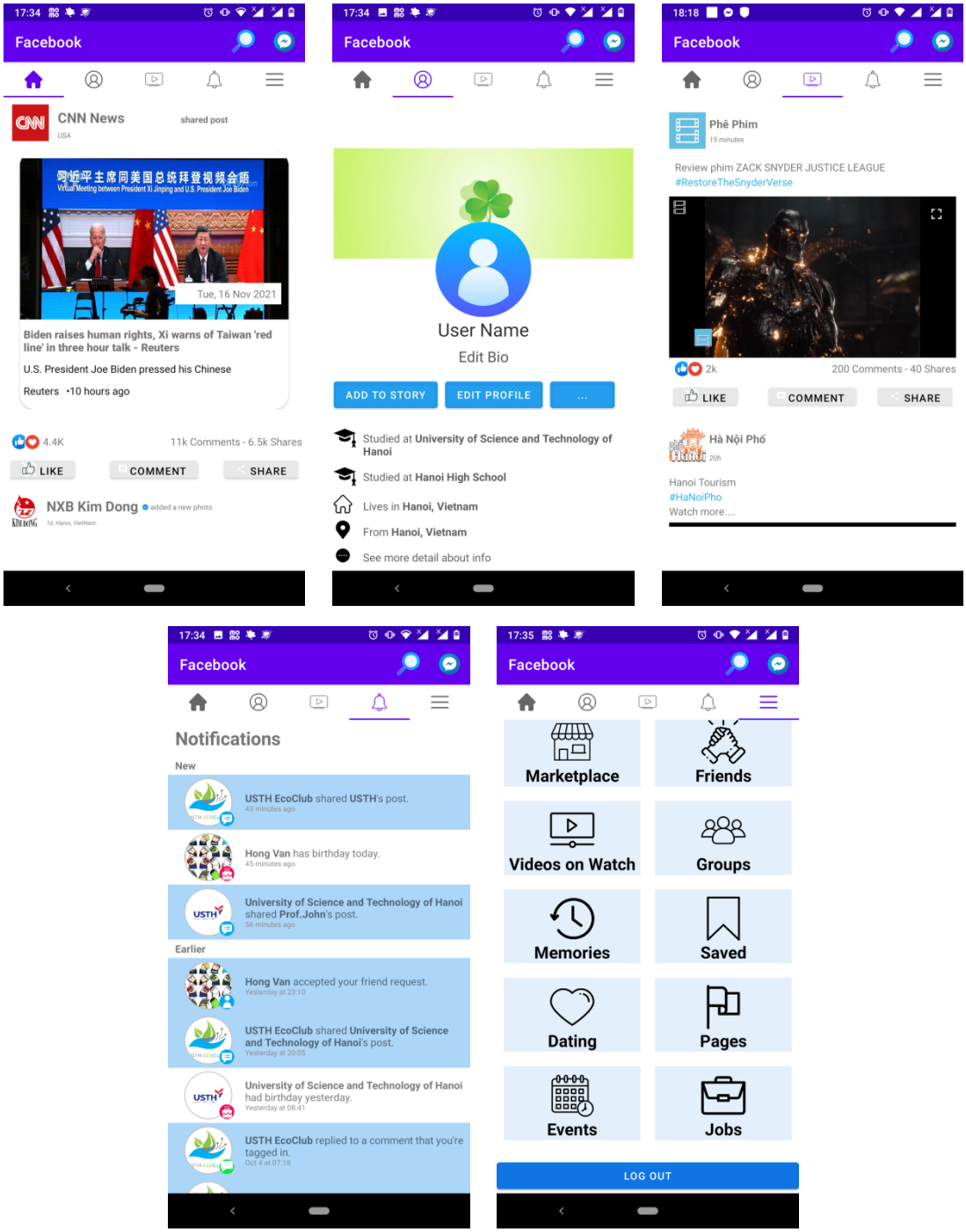
Here are some overall results from our project:

4.1 Login/Register



4.2 Main fragments

In the first row, from left to right, there are: News feed, User's Profile, Facebook watch. In the second row, there are: Notifications and Menu respectively:



5 CONCLUSION

To sum up, we have succeeded in making the Login/Logout/Register functions, getting posts and videos from the Internet, and creating a diverse user interface. However, we failed to apply API to all the content in our application, to post status and comments directly from the UI, and to modify the account's information. In the future, the first thing that we would love to work on is to improve all those flaws, and then maybe we can enhance the interface or add several extra features such as sharing posts, creating stories, posting pictures and videos from the device...

Again, thanks a million to your precious lectures and practical sessions, we could manage to apply so much knowledge into the project to enhance our mobile application.