# Group 39: FixTheWifi

Kevin, Joe, Andy, David, Brenden, Kotya, Marcus

# TABLE OF CONTENTS

## Specifications

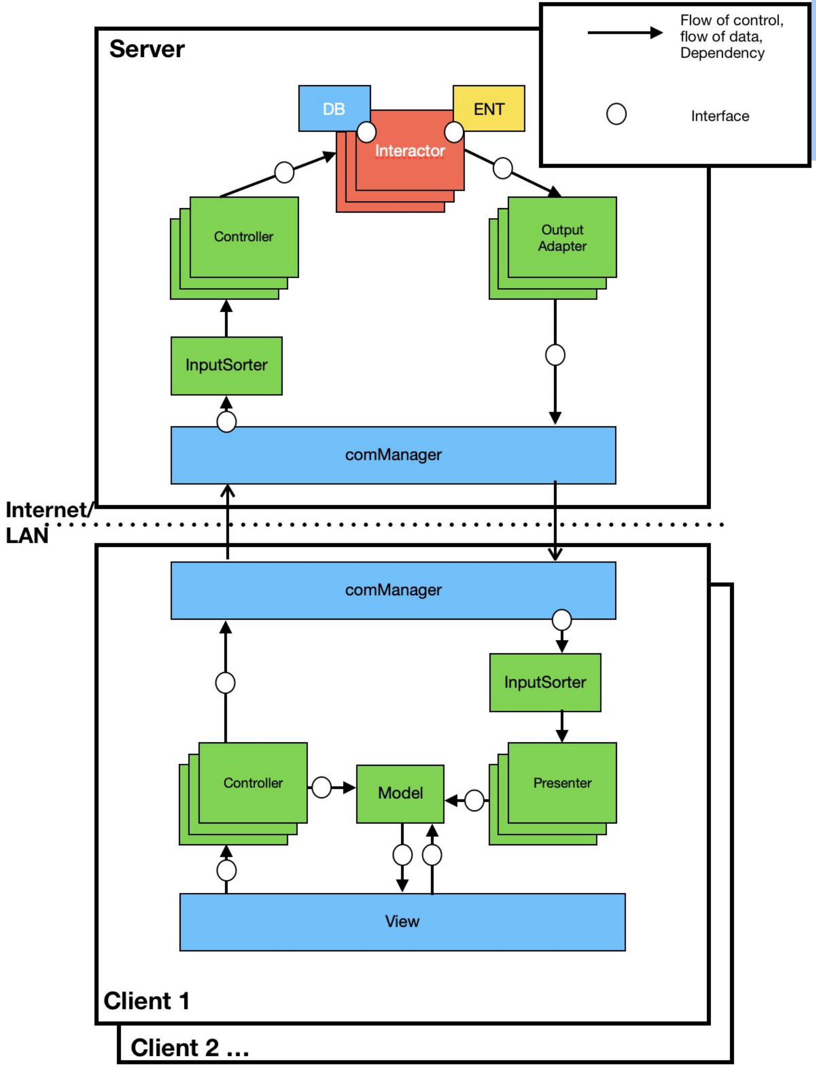
The program allows users to chat. Starting with registering an account and profile settings. Users will add friends and create a group chat or direct message to start the conversation using text messages, videos, and images.

## Intro of usecases

1) Login, Logout, Register, Delete Account
2) Request, Accept/Decline, Delete Friends
3) Delete and Edit Messages
4) Group Chats
5) Private  Chats
6) Profile settings
7) User Ratings
8) Send message

# General Architecture

- 2 or more Sub systems Clients and Server
-
- Logic only on server side
- Client side only convert data format, doesn't add or change data



## 1) Login, Logout, Register, Delete Account

These use cases enable the client to manage their accounts.

The client types in commands $reg $logout $lgn and $del_account to use these use cases.

The use cases ultimately change the user.csv file, which is where the data is stored.

Login changes the online status to T and sends the profile to the client so the client can send messages using the console.

Logout changes the online status to F.

Register creates a new row (which is the profile of the user).

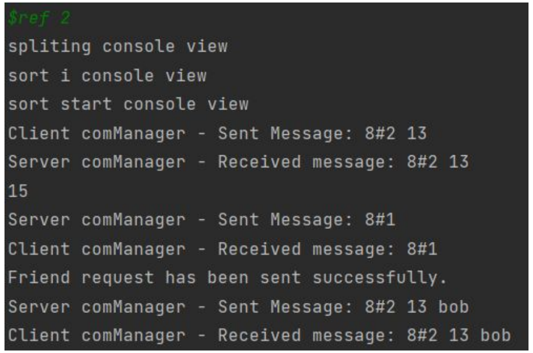
Delete account changes all the columns of the profile into DELETED.

## 2) Request Friend, Accept/Decline Friend, Delete Friend

1. Request Friend: It takes either the name or user ID of the friend to add. It sends back to both information about who is the requester and if entering a wrong name or user ID.
2. Accept/Decline Friend: It takes accept or reject action to the request, with enter user ID.
3. Delete Friend: To delete a friend, just to enter the user ID of the friend in the console. It sends back if the action is success or failed. It can extends some features to take the friend`s name as input in the future.
4. Console commands for friend usecases:

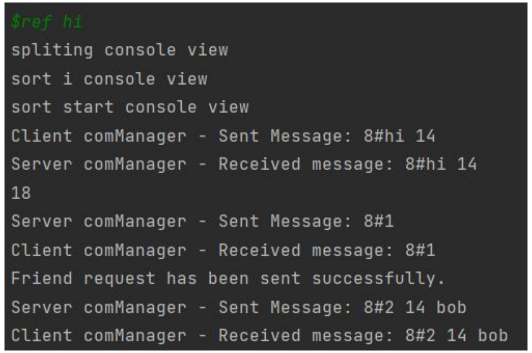
   ```
   $ref <uid/name>, $acp <uid>, $rej <uid>, $dtf <uid>
   ```

# Demo



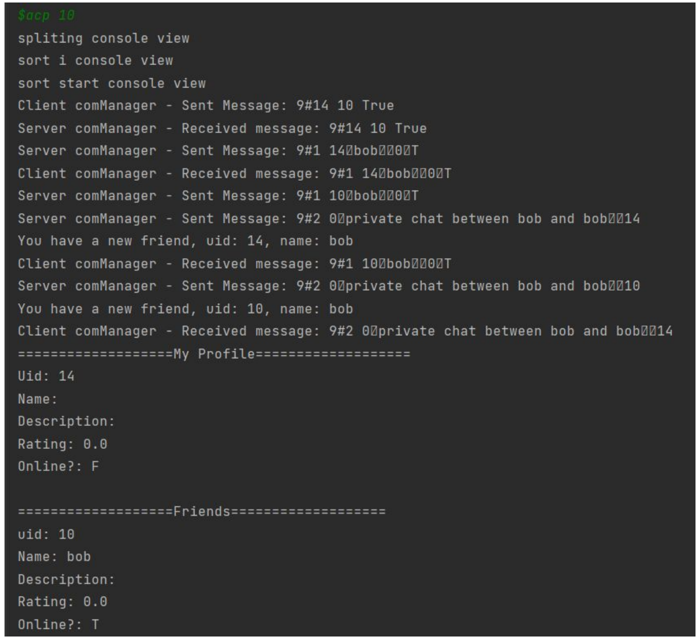
```
$ref 2
spliting console view
sort i console view
sort start console view
Client comManager - Sent Message: 8#2 13
Server comManager - Received message: 8#2 13
15
Server comManager - Sent Message: 8#1
Client comManager - Received message: 8#1
Friend request has been sent successfully.
Server comManager - Sent Message: 8#2 13 bob
Client comManager - Received message: 8#2 13 bob
```

Request friend by uid

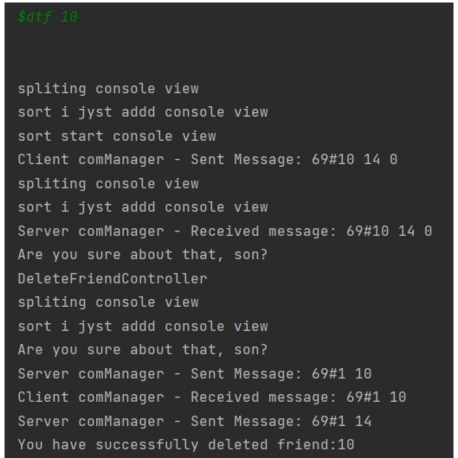


```
$ref hi
spliting console view
sort i console view
sort start console view
Client comManager - Sent Message: 8#hi 14
Server comManager - Received message: 8#hi 14
18
Server comManager - Sent Message: 8#1
Client comManager - Received message: 8#1
Friend request has been sent successfully.
Server comManager - Sent Message: 8#2 14 bob
Client comManager - Received message: 8#2 14 bob
```

Request friend by name



```
$acp 10
spliting console view
sort i console view
sort start console view
Client comManager - Sent Message: 9#14 10 True
Server comManager - Received message: 9#14 10 True
Server comManager - Sent Message: 9#1 14▯bob▯▯0▯T
Client comManager - Received message: 9#1 14▯bob▯▯0▯T
Server comManager - Sent Message: 9#1 10▯bob▯▯0▯T
Server comManager - Sent Message: 9#2 0▯private chat between bob and bob▯▯14
You have a new friend, uid: 14, name: bob
Client comManager - Received message: 9#1 10▯bob▯▯0▯T
Server comManager - Sent Message: 9#2 0▯private chat between bob and bob▯▯10
You have a new friend, uid: 10, name: bob
Client comManager - Received message: 9#2 0▯private chat between bob and bob▯▯14
===================My Profile===================
Uid: 14
Name:
Description:
Rating: 0.0
Online?: F

===================Friends===================
uid: 10
Name: bob
Description:
Rating: 0.0
Online?: T
```



```
$dtf 10


spliting console view
sort i jyst addd console view
sort start console view
Client comManager - Sent Message: 69#10 14 0
spliting console view
sort i jyst addd console view
Server comManager - Received message: 69#10 14 0
Are you sure about that, son?
DeleteFriendController
spliting console view
sort i jyst addd console view
Are you sure about that, son?
Server comManager - Sent Message: 69#1 10
Client comManager - Received message: 69#1 10
Server comManager - Sent Message: 69#1 14
You have successfully deleted friend:10
```

Delete friend

Accept friend

## 3) Delete and Edit Messages

This feature allows the users to delete and edit messages

The functions fetch the uid of the message in the database and delete or manipulate toe message content.

There are limitations of this feature in our program since it is ran in console, that we cannot right click on the messages for options to delete and edit from the screen like the apps in the market, but the message contents are manipulated in the server database, an can be checked with command  $edt

## 4) Group Chats

This Feature allows the user to message multiple users at once

First the user initiates group chat creation, then the user selects which users to join the group chat

One challenge I faced: refactoring interactor repeatedly from some details not planned before coding

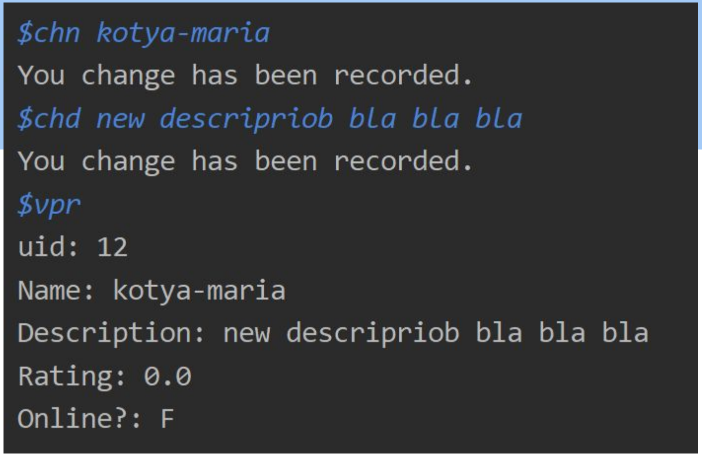
Solution: The magic of clean architecture🌈🦄

## 5) Private Chat and send Message

Enables users to send messages between different computers

- Sends to all members of a chat
- Private chat is automatically created when accepting a friend, and deleted when deleting friend.

## 6) Changing profile



```
$chn kotya-maria
You change has been recorded.
$chd new descripriob bla bla bla
You change has been recorded.
$vpr
uid: 12
Name: kotya-maria
Description: new descripriob bla bla bla
Rating: 0.0
Online?: F
```

User can:

- view their profile information which is stored in model at the client side
- edit their name and description sending request to the server side and if it was correct the information is stored at the database, as a confirmation of the success request the data is stored in the model at client side as well and the profile info is shown
- add or delete photo should work the same but due to console UI limitation we get rid of this feature

# 7): Rating

**Submit and View Rating**

- A novelty feature to differentiate and increase curiosity towards our application

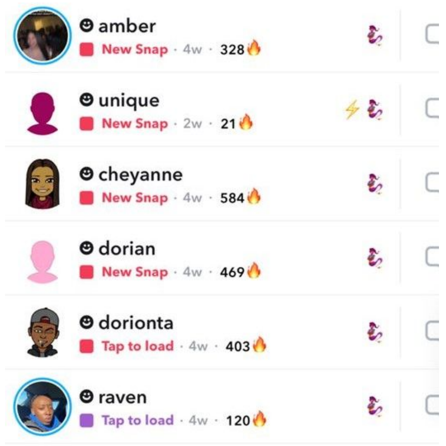
- It takes in the Receiver's User ID and a rating from 0 to 5. It recomputes the Receiver's new total rating, new number of ratings, and new average rating, stores them, and updates the Rater through the Console View if the rating is successful. (If the rating is unsuccessful, an exception will be caught and the Console View will notify the user of rating failure)

- $rate <UID> <Rating from 0-5>

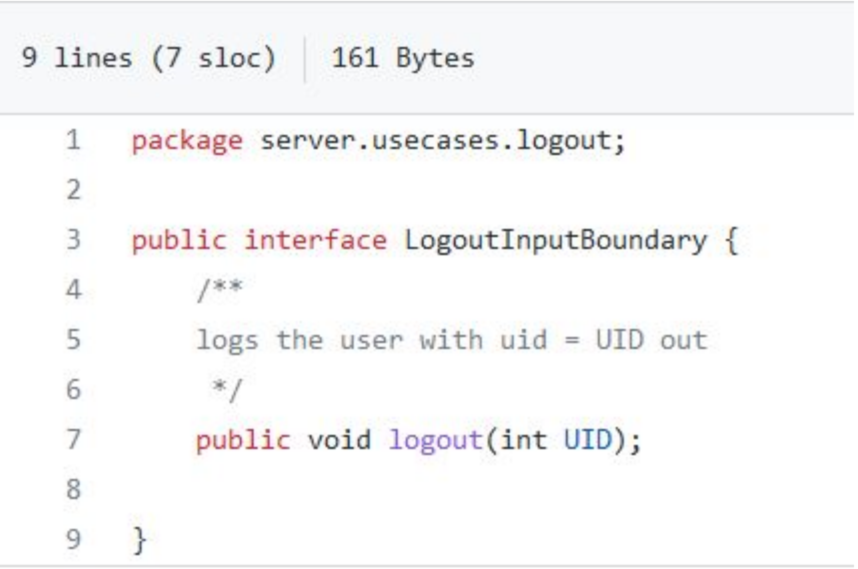
**Challenges**

- Issue with storing user's rating information in the database → met up for a pair programming session

2 minutes to share what you're up to
All your friends post at the same time
Comment and react with RealMojis

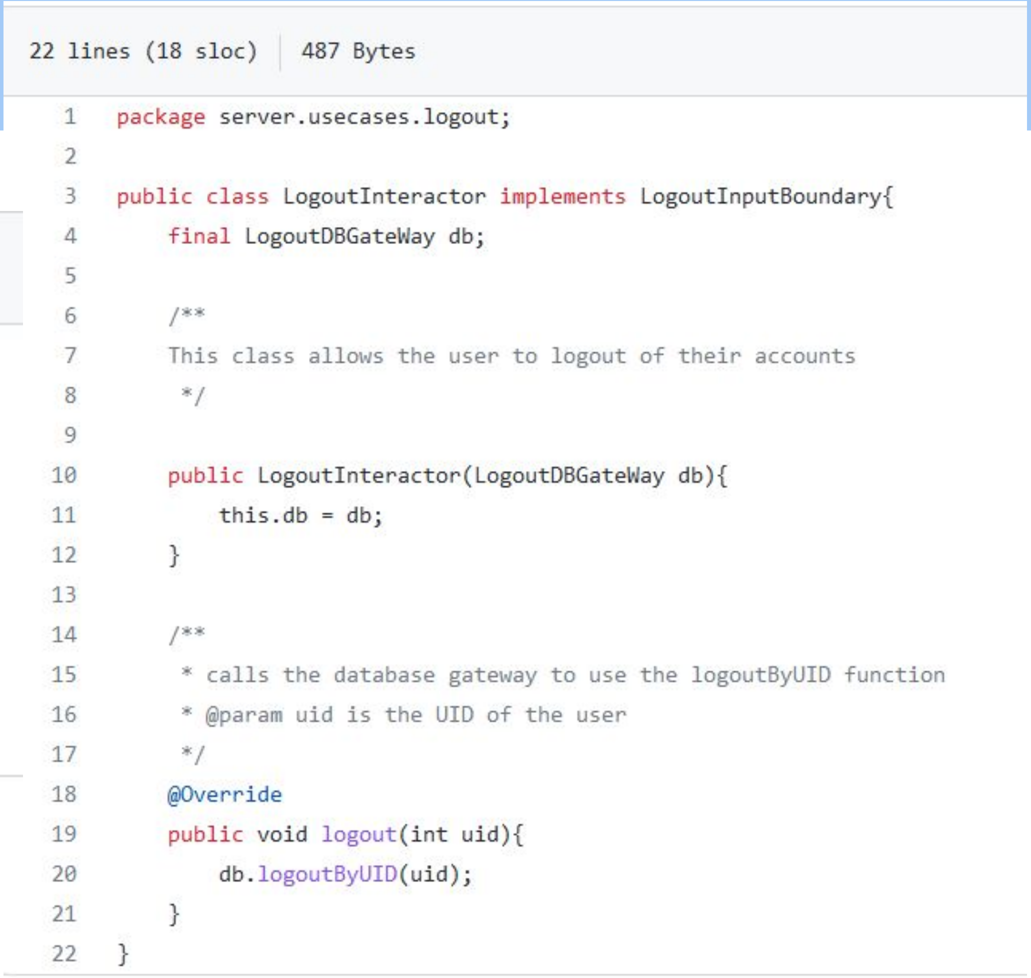

amber · New Snap · 4w · 328🔥
unique · New Snap · 2w · 21🔥
cheyanne · New Snap · 4w · 584🔥
dorian · New Snap · 4w · 469🔥
dorionta · Tap to load · 4w · 403🔥
raven · Tap to load · 4w · 120🔥

## SOLID (S)



9 lines (7 sloc) | 161 Bytes

```java
package server.usecases.logout;

public interface LogoutInputBoundary {
    /**
    logs the user with uid = UID out
     */
    public void logout(int UID);

}
```



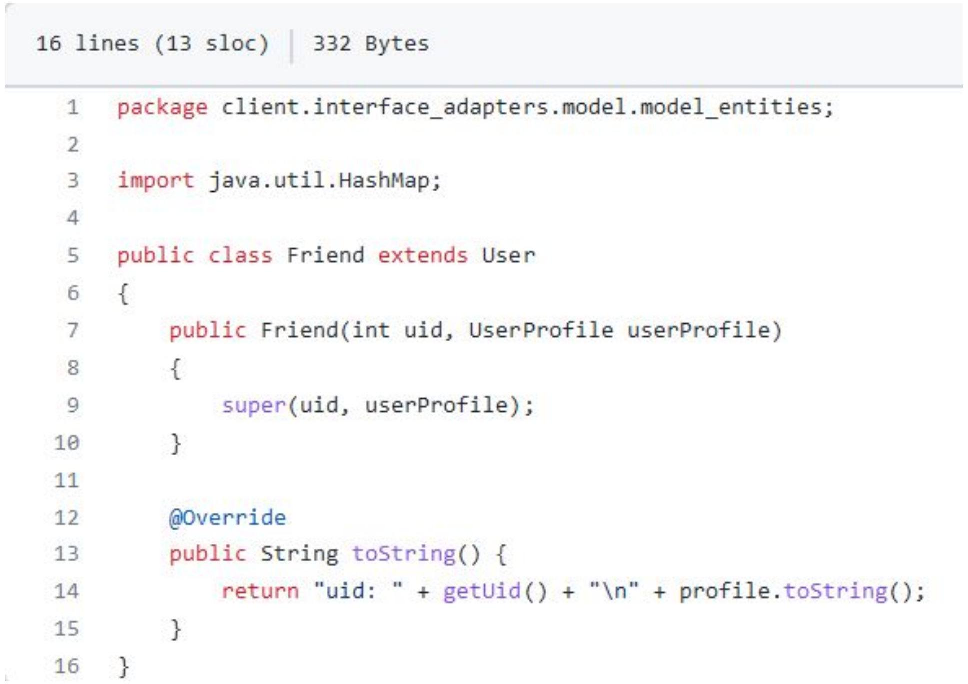
22 lines (18 sloc) | 487 Bytes

```java
package server.usecases.logout;

public class LogoutInteractor implements LogoutInputBoundary{
    final LogoutDBGateWay db;

    /**
    This class allows the user to logout of their accounts
     */

    public LogoutInteractor(LogoutDBGateWay db){
        this.db = db;
    }

    /**
     * calls the database gateway to use the logoutByUID function
     * @param uid is the UID of the user
     */
    @Override
    public void logout(int uid){
        db.logoutByUID(uid);
    }
}
```

## SOLID (O and L)

We have classes that are subclasses and they extend, not modify their parent class.

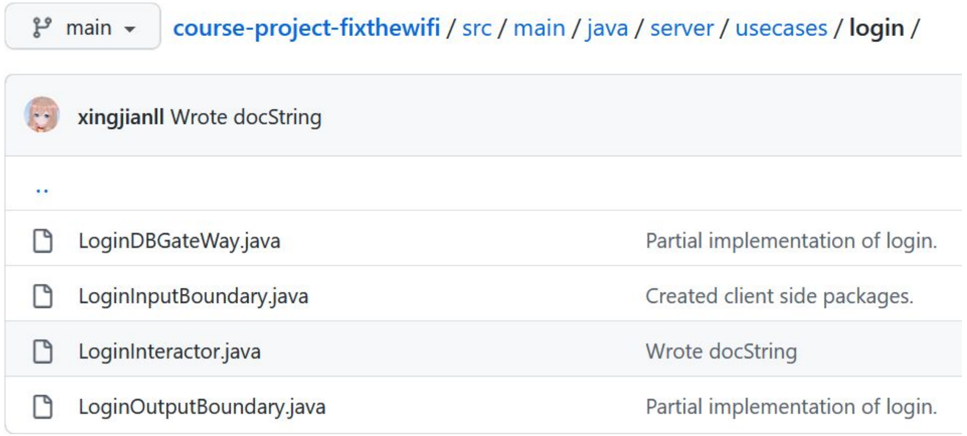
Therefore, they can be used as their parent class



16 lines (13 sloc) | 332 Bytes

```java
package client.interface_adapters.model.model_entities;

import java.util.HashMap;

public class Friend extends User
{
    public Friend(int uid, UserProfile userProfile)
    {
        super(uid, userProfile);
    }

    @Override
    public String toString() {
        return "uid: " + getUid() + "\n" + profile.toString();
    }
}
```

# SOLID (D)



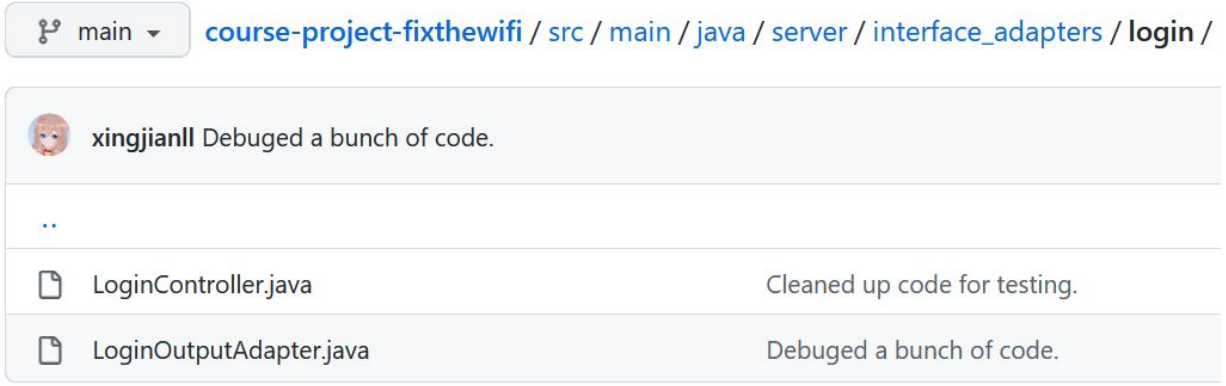
main

course-project-fixthewifi / src / main / java / server / usecases / **login** /

**xingjianll** Wrote docString

..

| | |
|---|---|
| LoginDBGateWay.java | Partial implementation of login. |
| LoginInputBoundary.java | Created client side packages. |
| LoginInteractor.java | Wrote docString |
| LoginOutputBoundary.java | Partial implementation of login. |



main

course-project-fixthewifi / src / main / java / server / interface_adapters / **login** /

**xingjianll** Debuged a bunch of code.

..

| | |
|---|---|
| LoginController.java | Cleaned up code for testing. |
| LoginOutputAdapter.java | Debuged a bunch of code. |

## Design Patterns Examples

Creational patterns: we used try and catch method in both request friend and delete friend to try to process two different inputs instead of if else method which may not work in this case.

Structural patterns: We followed the structural design pattern of adaptor pattern. Each of friend feature has an input adaptor both in front end and the back end.

Behavior patterns: For null object, in comManager if debug parameter is always false to avoid null reference by providing a default object.

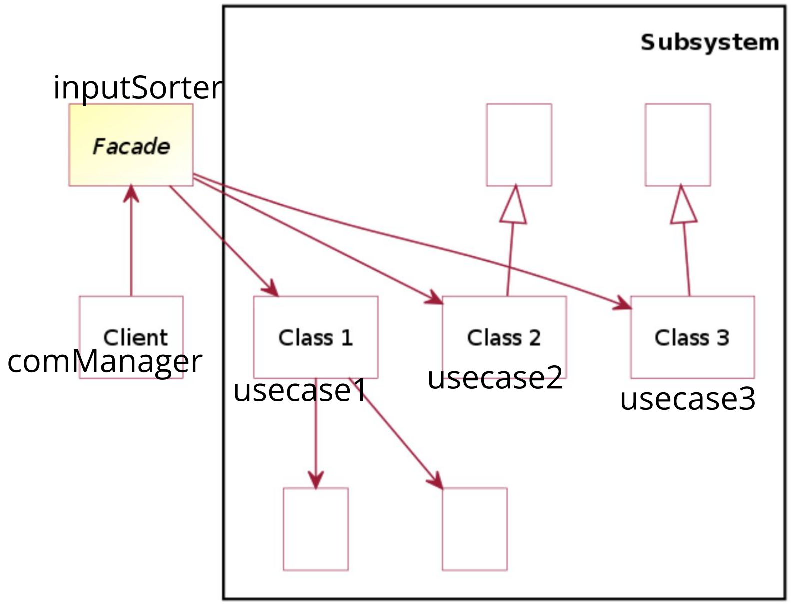
# More design patterns

Observer - View and Mode

- Model needs to update view
- May need many views

Adapter/Facade - InputSorter

- Adpater between two interface
- Makes controlling usecases easy
- Makes comManager portable



## Ethics

Since our program is a console app, it only works for people with good eyesight and people who can use a keyboard and mouse normally.

I’d also say that everyone who cannot read or write (children) will be at a disadvantage because then they would not be able to use the app to contact their parents.

I think a good feature to add for this application is to add voice messages so that it could be accessible to more people. (If we had more time)

## Future Implementations

GUI

Read out text

Audio messages

Audio input, video input, photo input

Different languages

Encryption

## Challenges and solutions

Merge conflicts

Code styles were different among groupmates