試験報告書

プロジェクト2

グループ A

提出日：2018年7月20日

1. はじめに

本報告書は，ネットワーク共有型教えあいアプリの試験方法および試験結果を記すものである．試験対象であるシステムのソースコードおよびクラス図は以下の通りである．

ソースコード一覧：

User.java UserDriver.java (テスト用ドライバ)

Question.java QuestionDriver.java (テスト用ドライバ)

Group.java GroupDriver.java(テスト用ドライバ)

Server.java Client.java

Androidアプリ用のソースコード

|  |
| --- |
| **サーバ (Server)** |
| －クライアント接続状態:Boolean  －ポート番号:int  －ユーザ情報:String[] |
| ＋コンストラクタ(ポート番号:int): Void  ＋クライアントの接続: Boolean  ＋ログイン認証(プレイヤ名：String,パスワード:String) ：Boolean  ＋アカウント作成(プレイヤ名：String,パスワード:String):Boolean  ＋メッセージの転送(メッセージ：String):Void  ＋メッセージの受信（メッセージ:String）:Void  ＋他ユーザ情報の転送()：Void  ＋ユーザ情報の更新(情報:String or int):Void  ＋ |

|  |
| --- |
| **クライアント (Client)** |
|  |
| ＋サーバに接続(IP-address：String , Port-number：int): Void  ＋ログイン申請(ユーザ名：String,パスワード:String):Boolean  ＋アカウント作成(ユーザ名:String,パスワード:String):Boolean  ＋サーバにユーザ情報を転送(ユーザ情報:String[]):Void  ＋サーバからユーザ情報を受信(ユーザ情報:String[]):Void  ＋サーバからグループメンバー情報を取得(グループ名:String):String[][]  ＋サーバにグループの作成を要求(グループ名:String):Void  ＋サーバに質問（回答）内容を送信(送信する相手:String, メッセージ:String):Void  ＋サーバから質問（回答）内容を受信(送信した相手:String, メッセージ:String):Void  ＋サーバから質問群を受信(質問者:String, 質問内容:String, 回答者:String, 回答内容:String):Void  ＋サーバに評価値を送信(ユーザ名:String, 評価値:int):Void  ＋サーバから評価値を受信(評価値:int)  ＋サーバにオファーを送信(ユーザ名:String):Void  ＋サーバからオファーを受信(ユーザ情報:String[]):Void  ＋サーバから立候補者情報を受信(各ユーザ情報:String[][]):Void  ＋サーバに送金申請(金額:int):Void  ＋サーバに着金申請(金額:int, アドレス:String):Void  ＋サーバから保持金額を取得():int  ＋メッセージの受信（メッセージ:String）:Void  ＋画面更新:Void  ＋通報する(通報相手:String, 理由:String):Void |

|  |
| --- |
| **ユーザ (User)** |
| －ユーザ名:String  －パスワード:String  －合言葉：String  －職業String  －所属String  －評価値　double  －質問数　question  －回答数 answer  －所属するグループ　ArrayList<String>  －コイン　int |
| ＋コンストラクタ(): 名前String,パスワードString,合言葉String  ＋ユーザ名を受付(ユーザ名：String): Void  ＋ユーザ名を取得()：String  ＋パスワードの受付(パスワード：String)：Void  ＋合言葉の受付():String  ＋アドレスの受付(アドレス:String):Void  ＋アドレスの取得():String  ＋評価値の受付(評価値:double):Void  ＋評価値の取得()：double  ＋質問数の受付():Void  ＋質問数の取得()：int  ＋質問数変更（int）void  ＋回答数の受付():Void  ＋回答数の取得()：int  ＋回答数変更(int):void  ＋職業の受付(職業所属:String):Void  ＋職業の取得():String  ＋所属の受付(職業所属:String):Void  ＋所属の取得():String  ＋グループ名の受付(グループ名:String):Void  ＋グループ名の取得():String  ＋グループリセット（）：void  ＋コインのプラス（int）:void  ＋コインのマイナス(int):void  +コインの取得():int |

|  |
| --- |
| **Question** |
| －質問者:User  －回答者:User  －オファー人:User  －オファー用フラグ:boo;lean  －質問内容:String  －回答内容:String  －質問の属するグループ:String  －立候補者のリスト:ArrayList<User>  －回答されたかどうか:boolean  －回答の評価値:value  －質問にかけられたコイン:int |
| ＋コンストラクタ(質問者:User, 質問内容:String, グループ:String, コイン:int):  ＋立候補者の設定(User):void  ＋オファー決定(User):void  ＋オファーキャンセル():void  ＋オファー取得():User  ＋オファーが設定されてるか確認():boolean  ＋回答した人設定(User):void  ＋回答文設定(String):void  ＋評価値設定(double);void  ＋コイン設定(int):void  ＋質問者取得():User  ＋質問内容取得():String  ＋グループ取得():String  ＋回答者取得():User  ＋回答取得():String  ＋立候補者取得():ArrayList<User>  ＋立候補者リセット():void  ＋評価値取得():double  ＋コイン取得():int  ＋回答済みかどうかを取得():boolean |

|  |
| --- |
| **Group** |
| －グループ名：String  －グループのチャット内容:ArrayList<Question>  －グループのメンバー：ArrayList<User>  －グループの紹介文：String |
| ＋コンストラクタ():グループ名:gname、紹介文:String、ユーザー一人目:User  ＋グループ名取得():String  ＋質問文追加(Question):void  ＋質問文の取得():Arraylist<Question>  ＋メンバーの追加(User):void  ＋メンバーの削除(User):void  ＋メンバーの取得():ArrayList<User>  ＋紹介文取得():String |

クラス図ここまで

2. 単体テスト

(ア)Userクラス

　　以下のドライバを用いて、Userクラスの単体テストを行った。

[ソースコード]

**package** pro2\_Driver;

**import** pro2.User;

**public** **class** UserDriver {

**public** **static** **void** main(String[] args) {

System.**out**.println("name:yamada,pass:1234,aikotoba:hello　でユーザを作成します。");

User **user**=**new** User("yamada","1234","hello");

System.**out**.println("name: "+user.getName());

System.**out**.println("pass "+user.getPassword());

System.**out**.println("aikotoba: "+user.getAikotoba());

System.**out**.println("name: "+user.getName());

System.**out**.println("jobをstudentにセットします。");

user.setJob("student");

System.**out**.println("job: "+user.getJob());

System.**out**.println("belongをynuにセットします。");

user.setBelong("ynu");

System.**out**.println("belong: "+user.getBelong());

System.**out**.println("valueを3.5にセットします。");

user.setValue(3.5);

System.**out**.println("value: "+user.getValue());

System.**out**.println("questionをセットします。");

user.setQuestion();

System.**out**.println("questionNum: "+user.getQuestion());

System.**out**.println("answerをセットします。");

user.setAnswer();

System.**out**.println("answerNum: "+user.getAnswer());

System.**out**.println("question数を0にセットします。");

user.decQuestion(0);

System.**out**.println("questionNum: "+user.getQuestion());

System.**out**.println("answer数を0にセットします。");

user.decAnswer(0);

System.**out**.println("answerNum: "+user.getAnswer());

System.**out**.println("gruopにsoccerをセットします。");

user.setGroup("soccer");

System.**out**.println("gruop: "+user.getGroup());

System.**out**.println("coinを100にセットします。");

user.setCoin(100);

System.**out**.println("coin: "+user.getCoin());

System.**out**.println("coinを100+します。");

user.plusCoin(100);

System.**out**.println("coin: "+user.getCoin());

System.**out**.println("coinを100-します。");

user.minusCoin(100);

System.**out**.println("coin: "+user.getCoin());

}

}

[出力結果]

name:yamada,pass:1234,aikotoba:hello　でユーザを作成します。

name: yamada

pass 1234

aikotoba: hello

name: yamada

jobをstudentにセットします。

job: student

belongをynuにセットします。

belong: ynu

valueを3.5にセットします。

value: 3.5

questionをセットします。

questionNum: 1

answerをセットします。

answerNum: 1

question数を0にセットします。

questionNum: 0

answer数を0にセットします。

answerNum: 0

gruopにsoccerをセットします。

gruop: [soccer]

coinを100にセットします。

coin: 100

coinを100+します。

coin: 200

coinを100-します。

coin: 100

(イ) Questionクラス

以下のドライバを使ってQuestionクラスの単体テストを行った。

[ソースコード]

**package** pro2\_Driver;

**import** pro2.Question;

**import** pro2.User;

**public** **class** QuestionDriver {

**public** **static** **void** main(String[] args) {

// **TODO** 自動生成されたメソッド・スタブ

User **u1** = **new** User("u1", "1", "しつもんしゃ");

System.**out**.println("(u1, 質問１, グループ１, 100)でQuestionクラスのオブジェクト生成");

Question **q1** = **new** Question(u1, "質問１", "グループ１", 100);

System.**out**.println("getQuestioner出力: "+q1.getQuestioner().getName());

System.**out**.println("getQuestion出力: "+q1.getQuestion());

System.**out**.println("getGroup出力: "+q1.getGroup());

System.**out**.println("getCoin出力: "+q1.getCoin());

User **u2** = **new** User("u2", "2", "りっこうほしゃ");

User **u3** = **new** User("u3", "3", "りっこうほしゃ");

System.**out**.println("u2,u3をsetCandidatesで立候補者に追加");

q1.setCandidates(u2);

q1.setCandidates(u3);

**for**(User **u**: q1.getCandidates()) {

System.**out**.println("getCandidates出力: "+u.getName());

}

System.**out**.println("u2をsetOfferでオファー");

q1.setOffer(u2);

System.**out**.println("getOffer出力: "+q1.getOffer().getName());

System.**out**.println("u2がcanselOfferでオファー拒否");

q1.canselOffer();

System.**out**.println("checkOffered出力: "+q1.checkOffered());

System.**out**.println("u3がdelCandidateで立候補キャンセル");

q1.delCandidate(u3);

**for**(User **u**: q1.getCandidates()) {

System.**out**.println("getCandidates出力: "+u.getName());

}

System.**out**.println("再びu2にオファー");

System.**out**.println("u2が回答");

q1.setAnswer("回答１");

q1.setAnswerer(u2);

System.**out**.println("getAnswerer出力: "+q1.getAnswerer().getName());

System.**out**.println("getAnswer出力: "+q1.getAnswer());

System.**out**.println("評価値4.5でsetValueで評価");

q1.setValue(4.5);

System.**out**.println("getValue出力: "+q1.getValue());

}

}

[出力結果]

(u1, 質問１, グループ１, 100)でQuestionクラスのオブジェクト生成

getQuestioner出力: u1

getQuestion出力: 質問１

getGroup出力: グループ１

getCoin出力: 100

u2,u3をsetCandidatesで立候補者に追加

getCandidates出力: u2

getCandidates出力: u3

u2をsetOfferでオファー

getOffer出力: u2

u2がcanselOfferでオファー拒否

checkOffered出力: false

u3がdelCandidateで立候補キャンセル

getCandidates出力: u2

再びu2にオファー

u2が回答

getAnswerer出力: u2

getAnswer出力: 回答１

評価値4.5でsetValueで評価

getValue出力: 4.5

(ウ)Groupクラス

　　以下のドライバを使って、Groupクラスの単体テストを行った。

[ソースコード]

**package** pro2\_Driver;

**import** java.util.ArrayList;

**import** pro2.Group;

**import** pro2.Question;

**import** pro2.User;

**public** **class** GroupDriver {

**public** **static** **void** main(String[] args) {

String **a1**="an";

String **a2** ="ap";

String **a3**="あっぷる";

String **b1**="bn";

String **b2**="bp";

String **b3**="ぶどう";

ArrayList<Question> **qlist**=**new** ArrayList<Question>();

ArrayList<User> **ulist**=**new** ArrayList<User>();

User **ua**=**new** User(a1,a2,a3);//ユーザーaさんの作成

User **ub**=**new** User(b1,b2,b3);//ユーザーbさんの作成

Group **g**=**new** Group("グループ名A","Aです",ua);//aさんによるグループの作成

Question **q**=**new** Question(ua,"質問内容です","グループ名A",100);

System.**out**.println("グループクラスの試験を開始する");

System.**out**.println();

System.**out**.println("getgname()でグループ名を取得");//getgnameによるグループ名の取得

System.**out**.println(g.getgname());

System.**out**.println();

System.**out**.println("setchat()でグループの質問リストに追加します");

g.setchat(q);

System.**out**.println("getchat()でグループの質問リストを取得");

qlist=g.getchat();

System.**out**.println("取得した質問リストの質問内容の表示");

**for**(Question **l**:qlist) {

System.**out**.println(l.getQuestion());

}

System.**out**.println();

System.**out**.println("setmember()によるグループメンバーの追加");//bさんをsetmember()でメンバーに追加

g.setmember(ub);

System.**out**.println("getmember()によるグループメンバーーの取得");//aさんとbさんが現時点でのメンバー

ulist=g.getmember();

System.**out**.println("取得したメンバーの名前の表示");

**for**(User **u**:ulist) {

System.**out**.println(u.getName());

}

System.**out**.println();

System.**out**.println("delmember()によるグループメンバーの削除");

g.delmember(ub);//ｂさんをグループから削除

System.**out**.println("削除後のメンバーの表示");//aさんのみが現在のメンバー

ulist=g.getmember();

**for**(User **u**:ulist) {

System.**out**.println(u.getName());

}

ulist.clear();

System.**out**.println("getintro()によるグループの紹介文の取得・表示");//グループ作成時に設定した紹介文が出れば成功

System.**out**.println(g.getintro());

System.**out**.println();

System.**out**.println("グループクラスの試験を終了する");

}

}

[出力結果]

グループクラスの試験を開始する

getgname()でグループ名を取得

グループ名A

setchat()でグループの質問リストに追加します

getchat()でグループの質問リストを取得

取得した質問リストの質問内容の表示

質問内容です

setmember()によるグループメンバーの追加

getmember()によるグループメンバーーの取得

取得したメンバーの名前の表示

an

bn

delmember()によるグループメンバーの削除

削除後のメンバーの表示

an

getintro()によるグループの紹介文の取得・表示

Aです

グループクラスの試験を終了する

(エ)Clientクラス、Serverクラス

　　これらのクラスはシステム上互いに接続しないと動かないので、単体テストではなく、結合テストで試験を行う。

3. 結合テスト

以下の手順に従い、結合テストを行った。

1. サーバーを起動する
2. 1つ目のクライアントを起動する（クライアントは全部で４つ起動させる）
   * サーバーと接続する
   * アカウントを新規作成する
   * グループを新規作成する
   * アカウント情報を設定する
   * ユーザー情報をサーバーから取得し、確認する
   * 質問を３つ作り、サーバーに送る
   * サーバーから自分がした質問を取得し、確認する
   * 先ほど作ったグループの情報をサーバーから取得し、確認する
3. 2つ目のクライアントを起動する
   * サーバーと接続する
   * アカウントを新規作成する
   * アカウント情報を設定する
   * ユーザー情報をサーバーから取得する
   * 質問に対し立候補する
   * 立候補した質問をサーバーから取得し、確認する
4. 3つ目のクライアントを起動する
   * サーバーと接続する
   * 1つ目のクライアントで作ったアカウントでログインする
   * 自分がした質問を受け取り、立候補者を確認する
   * 立候補者にオファーを出す
5. 4つ目のクライアントを起動する
   * サーバーと接続する
   * 2つ目のクライアントで作ったアカウントでログインする
   * 自分に来ているオファーをサーバーから取得し、確認する

[出力結果]

・クライアント側

(１つ目のクライアント)

サーバーと接続できました

1番です

グループ作成成功

グループ作成成功

アカウント情報送信成功

自分のUserオブジェクト取得成功

ub

東京大学

横浜国立大学

自分がした質問の受け取り成功

質問２

東京大学

質問１

横浜国立大学

質問０

横浜国立大学

グループ取得成功

横浜国立大学

グループ横浜国立大学内に質問あり

質問０

質問１

グループ横浜国立大学内のメンバー

ub

グループ横浜国立大学内の質問

質問０

質問１

(2つ目のクライアント)

サーバーと接続できました

2番です

アカウント情報送信成功

自分のUserオブジェクト取得成功

自分が立候補した質問の受け取り成功

立候補した質問は以下の通り

質問１

質問０

(3つ目のクライアント)

サーバーと接続できました

3番です

自分のUserオブジェクト取得成功

ub

自分がした質問の受け取り成功

質問１に対する立候補者はucです

質問０に対する立候補者はucです

(4つ目のクライアント)

サーバーと接続できました

4番です

自分のUserオブジェクト取得成功

uc

自分に来ているオファーの受け取り成功

質問１

質問０

・サーバー側

サーバーが起動しました．

ubが接続しました

グループ東京大学作成完了

グループ横浜国立大学作成完了

更新したユーザーの情報

ub,ub,ふたりめ,,,0.0,0,0,[東京大学, 横浜国立大学]

ub

質問作成成功

質問作成成功

質問作成成功

グループ送信成功

ubがログアウトしました

ucが接続しました

更新したユーザーの情報

uc,uc,さんにんめ,,,0.0,0,0,[東京大学, 横浜国立大学]

ub,ub,ふたりめ,,,0.0,3,0,[東京大学, 横浜国立大学]

uc

ucがログアウトしました

ubが接続しました

ub

ubがログアウトしました

ucが接続しました

uc

質問１

質問０

ucがログアウトしました

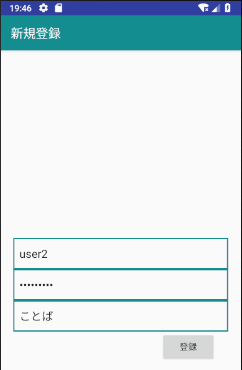
4. androidでの結合テスト

これまでのクラスをandroid studioに組み込んで、実機でアプリを実際に動かして結合テストを行った。手順は以下の通り。

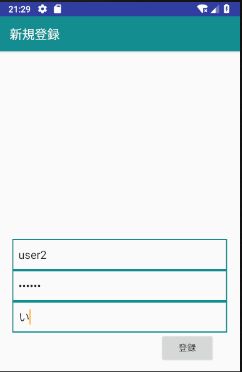
1. 新しいユーザーを二人分用意する。一人目（user1）は「大学生、理系」、二人目（user2）は「大学生、数学好き」でアカウント設定した。
2. user1で200コイン送金する。
3. user1で、「アロハ、ハワイについて」というグループを作成する。
4. user2で「アロハ」グループに入る。
5. user1で、「アロハ」グループ内で質問を送信。写真も添付。
6. user2で、(5)の質問に立候補する。
7. user1で、(5)の質問の立候補を確認し、オファーを出す。
8. user2で、(5)の質問に回答する。
9. user1で、(8)の回答を確認し、回答を評価する。
10. それぞれのユーザーで、ユーザー情報を確認する。
11. それぞれのユーザーで、グループ内の質問画面、グループメンバー確認画面を確認する。
12. user2で、user1に対して直接質問する。
13. user1で、(12)の質問のオファーを確認して、そのオファーを拒否する。その後、オファーが０であることを確認する。
14. user2で、ログアウトし、再度ログインできることを確認する。
15. user1,user2をログアウトする。

[出力結果]

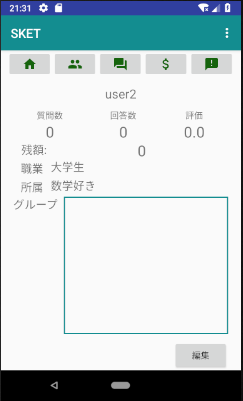
user2







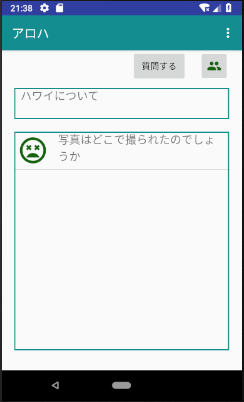


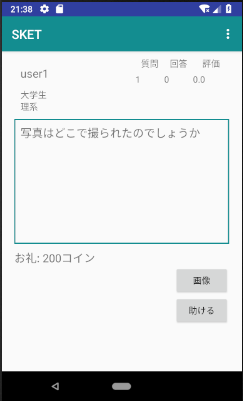




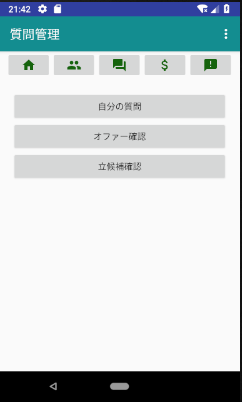


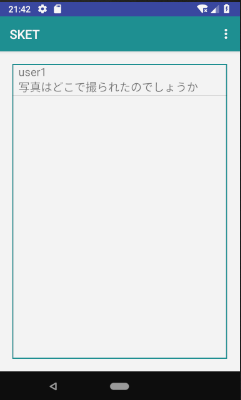


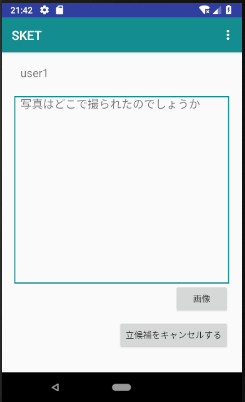


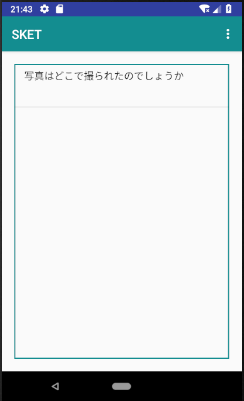


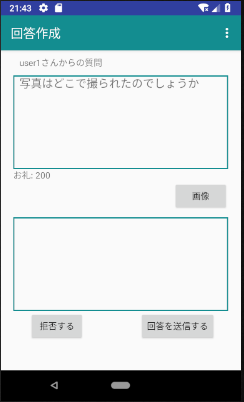


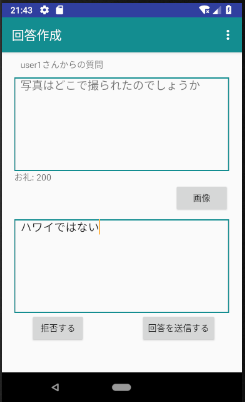




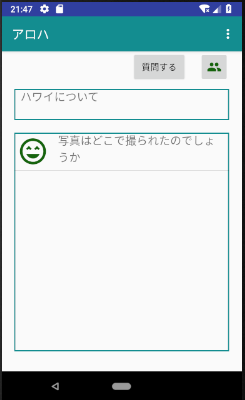


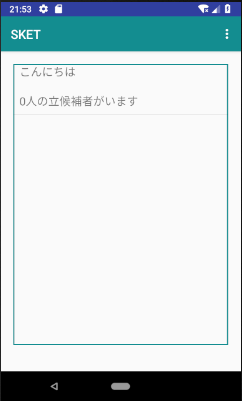
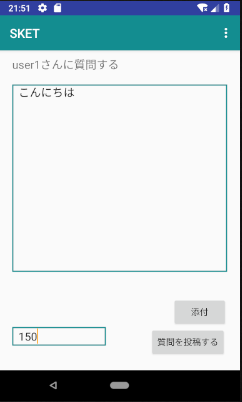
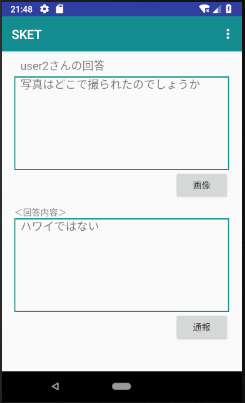




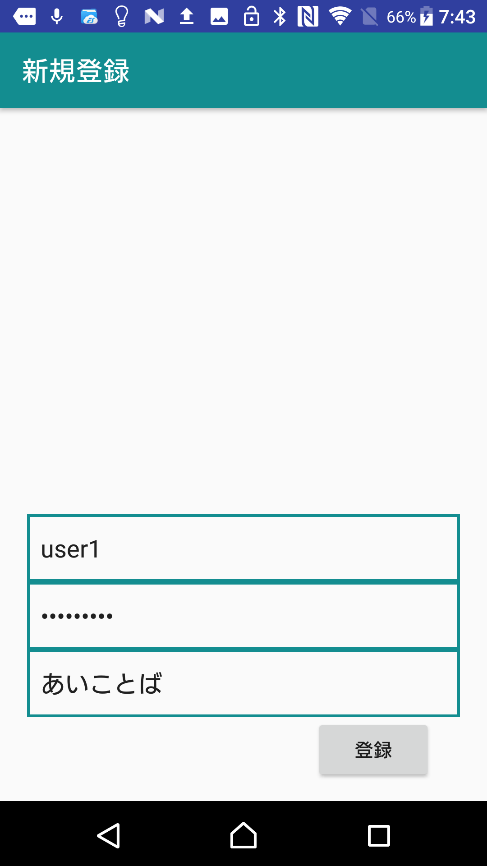




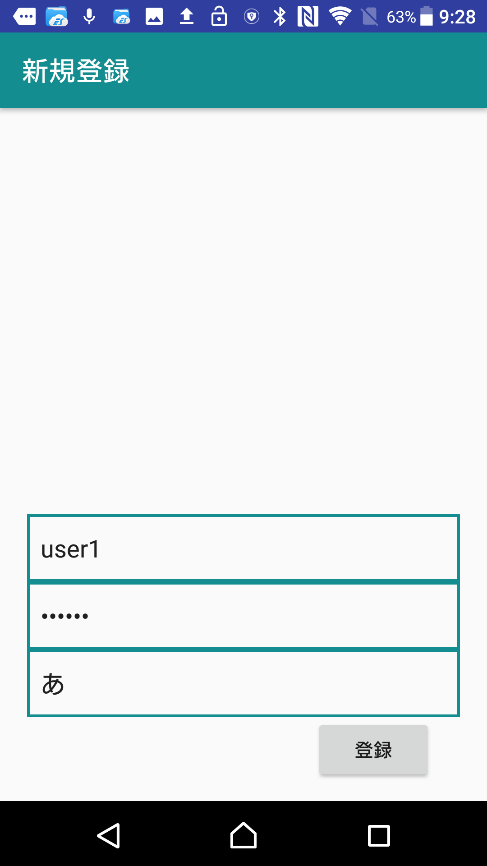




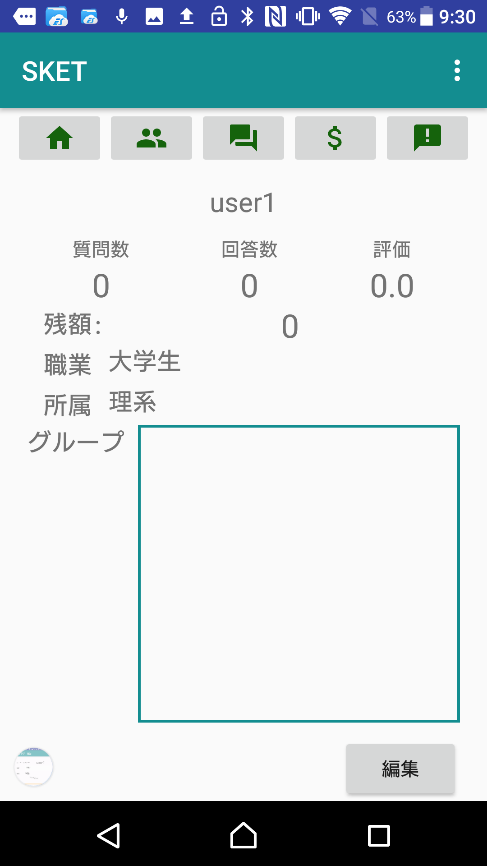
User1





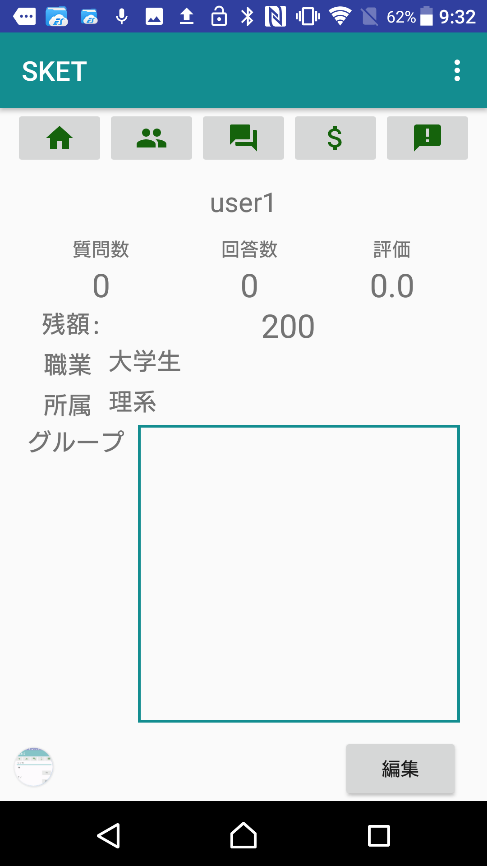


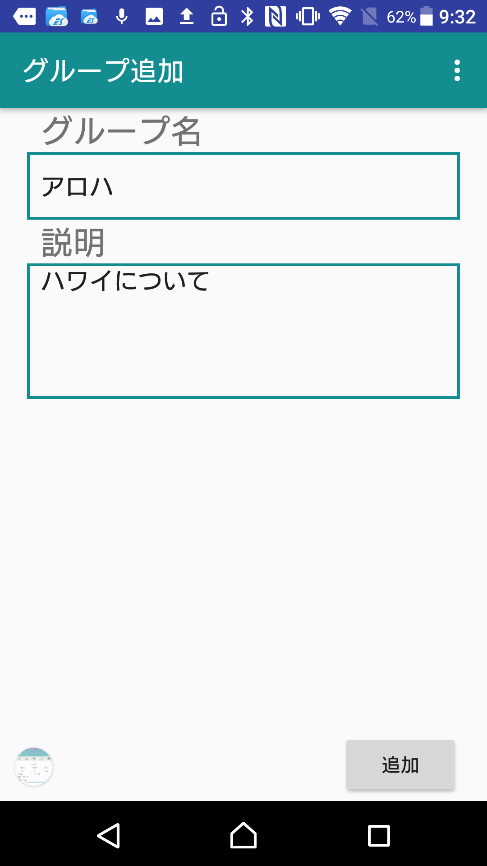


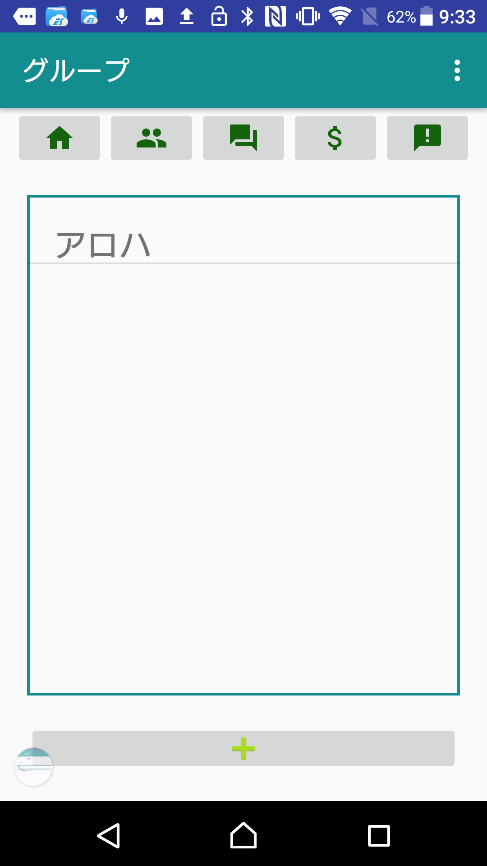




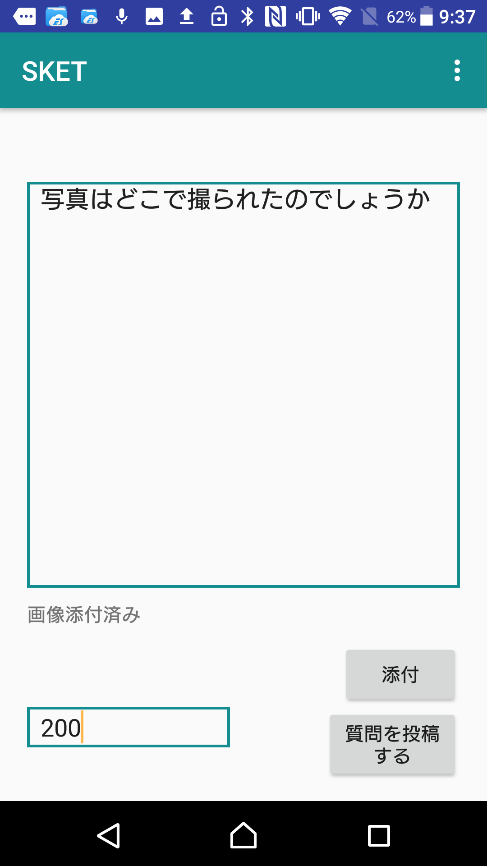




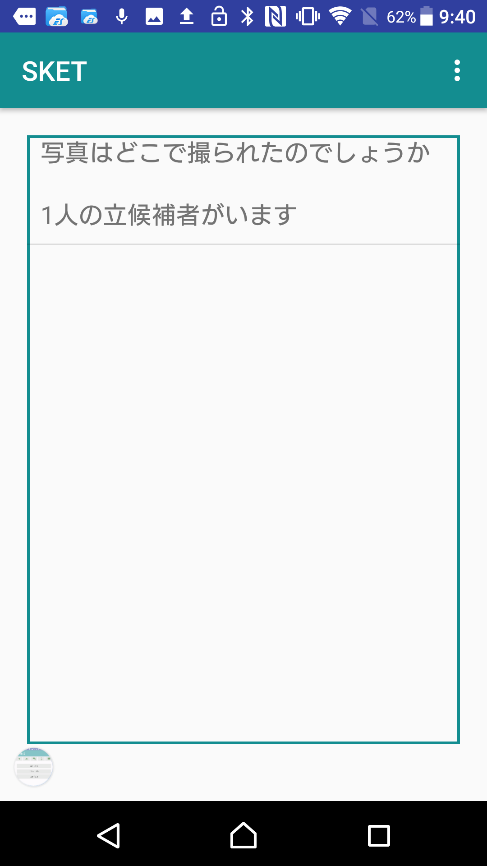


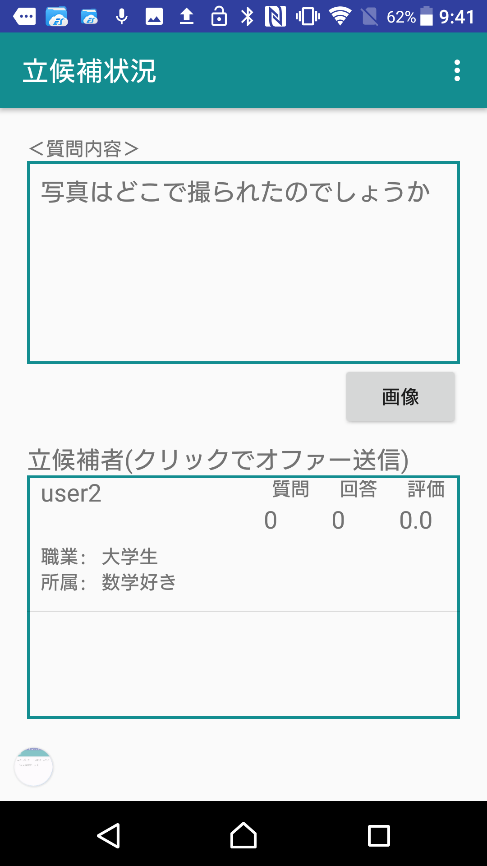






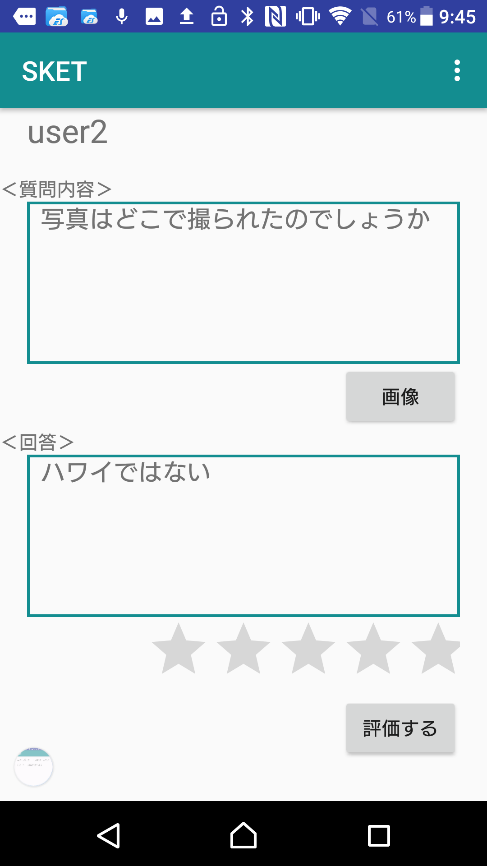


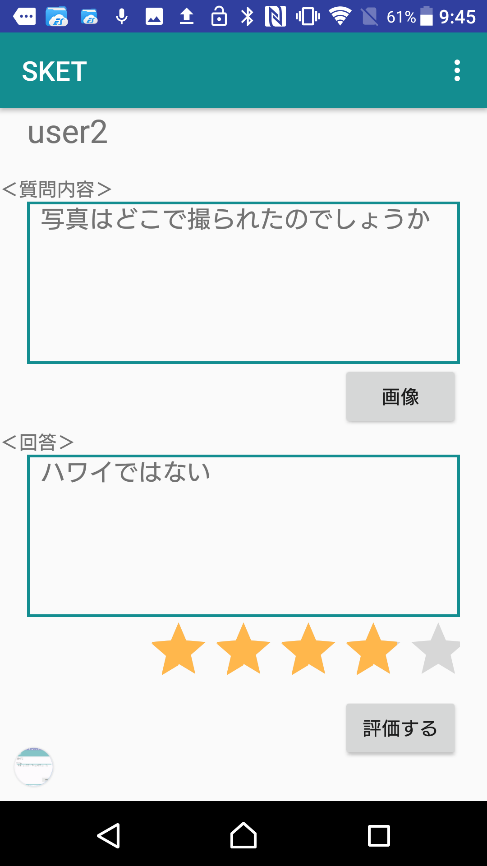




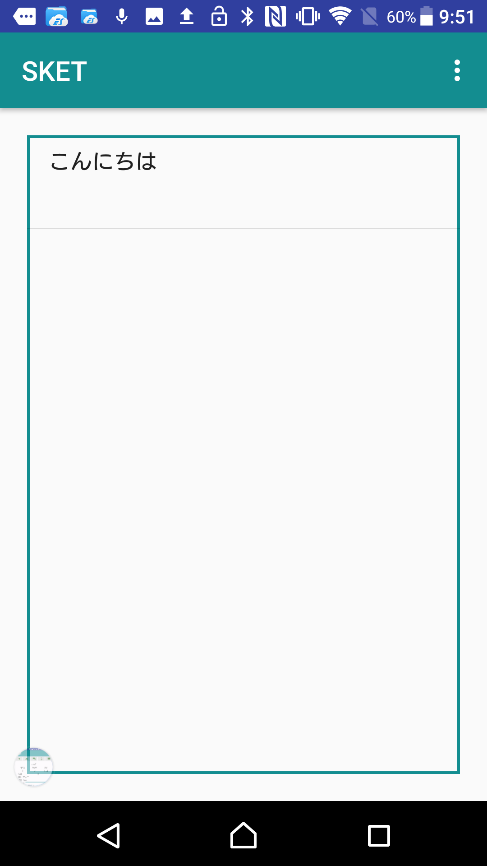




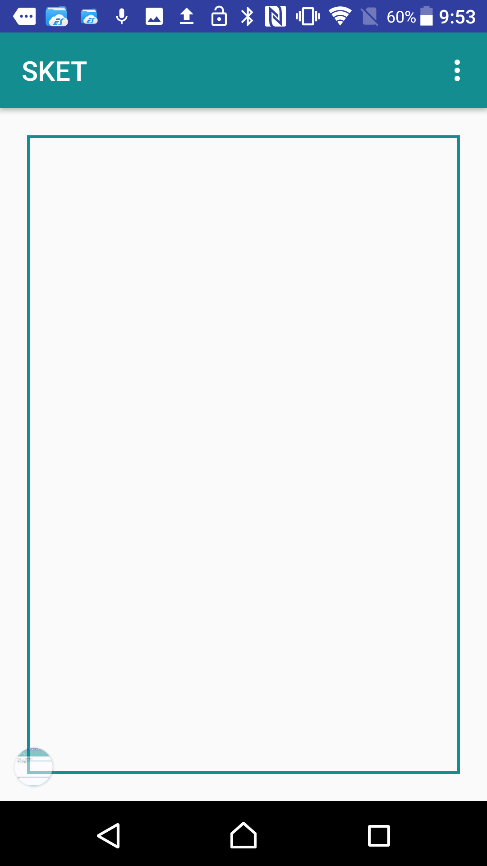














server

