VIA University College

Project Report

Surfing-couch

Rosalie Peree (258377) Grégoire Cartier (258378)

Supervisor & Academic advisor: Kasper & Jakob

2017 (Autumn semester)

Contents

Abstract

1 Introduction

1.1 Background Description

1.2 Problem Description

1.3 Limitations and Delimitations

2 Methods

The following table presents our choice of model and methods for the problems that we listed in our project description (see Appendix #1):

What *Problem*	**Why** *Why study this problem*	**Which** *Which models/theories did you use to solve the problem?*
How can we develop a social application which is fun to use and rewards the user for helping travellers in their area?	How to make the application interesting for our potential users?	We have to pay attention to all the following problems to be able to develop an application that will address all of them as good as possible.
What is important for travellers?	As our application is mainly targeted to travellers, it is interesting to know what is important for them, so that we cant try to implement those things in the final application.	We conducted a survey via Google Forms to gather people's opinion on that subject.
What is important for hosts?	As we want hosts to offer services to travellers, we have to meet with what they want to encourage them to do so.	We conducted a survey via Google Forms to gather potential user's opinion on that subject.
How can we allow users to create and find trips in their area and offer a service to travellers?	This problem is the main problem for a couch-surfing-like application. We have to localize the user to be able to provide a service for them in a specific area.	We have to set up a way for the users to enter their city/location and look for hosts in cities they want to visit.
How can we make the service secure for users?	Users don't want their personal data to be forwarded to strangers.	No personal data is displayed on the application for users to see. We implemented an internal messaging system so people can communicate directly on the application.
Can we add reviews and comments about travellers and hosts?	If you have to host someone or sleep/shower at a stranger's place, safety is very important. Users need a way to see if someone behaved correctly or not before their stay.	We implemented a system of user reviews - with grades - so people can report how they experience was with that specific user - hosts and travellers.

What Problem	**Why** Why study this problem	**Which** Which models/theories did you use to solve the problem?
How can we define the point-value of the provided services?	As the system is based on a reward, all services must have a fixed value.	We defined a value for each service - sleep, shower and laundry. The points are given to the host once the booking is completed.
How can we exchange the points for gifts?	The system is based on rewarding hosts with gifts for providing a service.	We set up an online shop where users can exchange their points for goods.
How will the user interface be implemented?	The user interface has to be easy to get and use.	The interface will be an Android application.
What kind of tool will be used for this implementation?	The tools used to develop can make the process of developing the application more or less easy.	We decided to develop the application using Android Studio, as it is the platform we are both most familiar with.
How will the interface be user-friendly?	The interface has to be easy to use and understand for any user.	The interface is simple, with buttons and labels that are easy to understand.
How is the social interaction going to take place?	The point of the assignment is to enable a social interaction.	In our application, the social interaction will take place in the form of direct messaging between two users, a common chat for all the users and the possibility of reviewing experience with other users.
What kind of database will be implemented?	We need a database that is free and easy to integrate in Android.	The chosen database is Firebase for our project.

3 Requirements

The users of the application will be any person interested in traveling on a budget or having a different traveling experience, as well of any person that will be willing to host or provide a service -for now, shower and laundry - to a traveller visiting their city. They could be from everywhere in the world and have about any age, as long as they have access to a mobile phone running Android and a working Internet connection. They should not have an extensive knowledge of computer science to be able to use the application.

3.1 List of Requirements

Before starting any implementation, we conducted a survey to gather some data about potential users. Here are some of the results that could be interesting:

- **More than 70%** of the people that replied are using an Android phone. Developing a native Android application seems to be a good strategic choice.
- Again, **more than 70%** of the people would be interested to collect rewards by hosting someone. For **34 th** of those people, it would also depend of the rewards.
- For **more than 50%** of the travellers, safety is the most important.
- For **20%** of our potential hosts, getting a reward would encourage them to host people in their home.
- For more statistics, refers to graphics in Annex B.

3.1.1 Functional Requirements

Use cases:

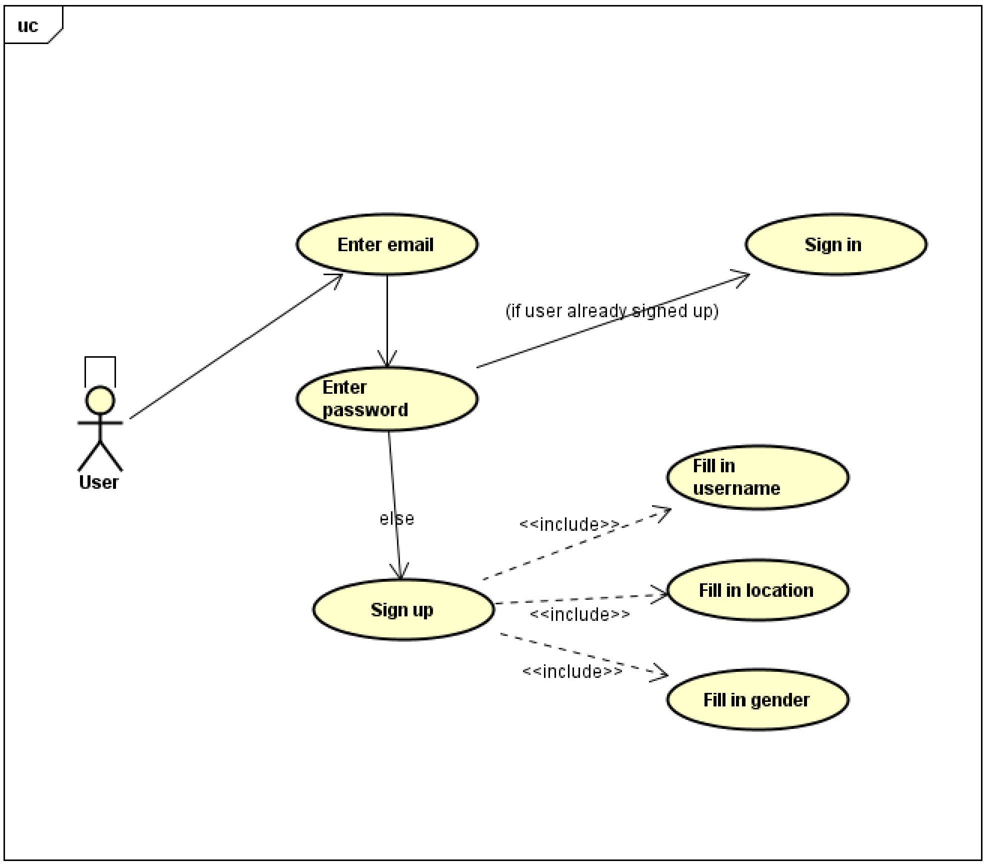


Figure 1: Use Case 1: how to access to application

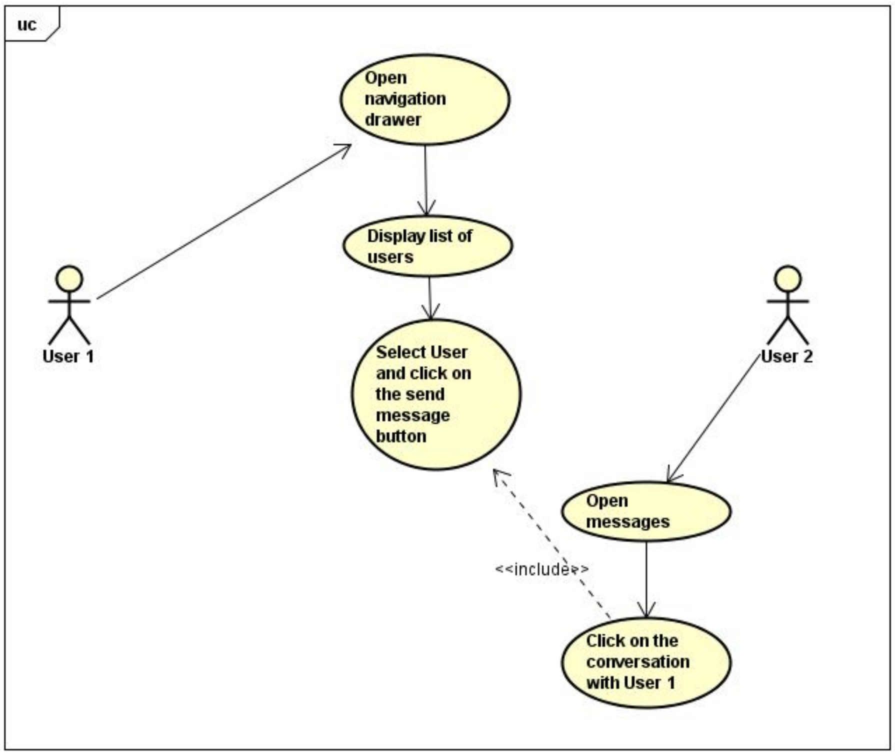

uc
Open navigation drawer
Display list of users
Select User and click on the send message button
User 1
User 2
Open messages
<<include>>
Click on the conversation with User 1

Figure 2: Use Case 2: how users can communicate together by messages

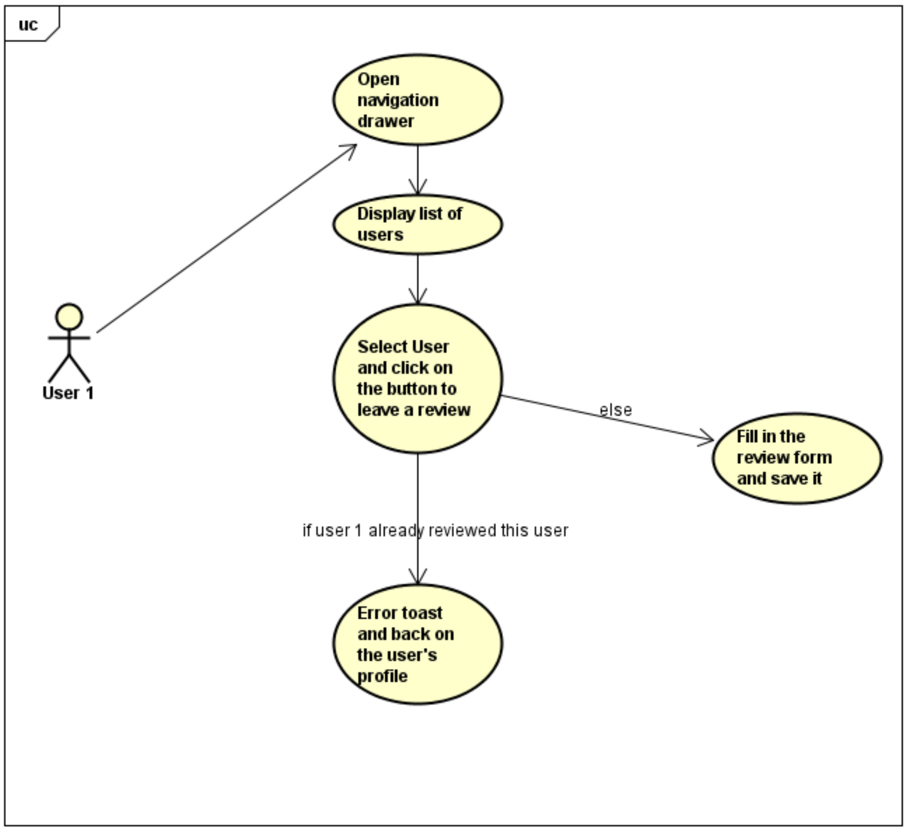

uc
Open navigation drawer
Display list of users
User 1
Select User and click on the button to leave a review
else
Fill in the review form and save it
if user 1 already reviewed this user
Error toast and back on the user's profile

Figure 3: Use Case 3: how a user can leave a review to another user

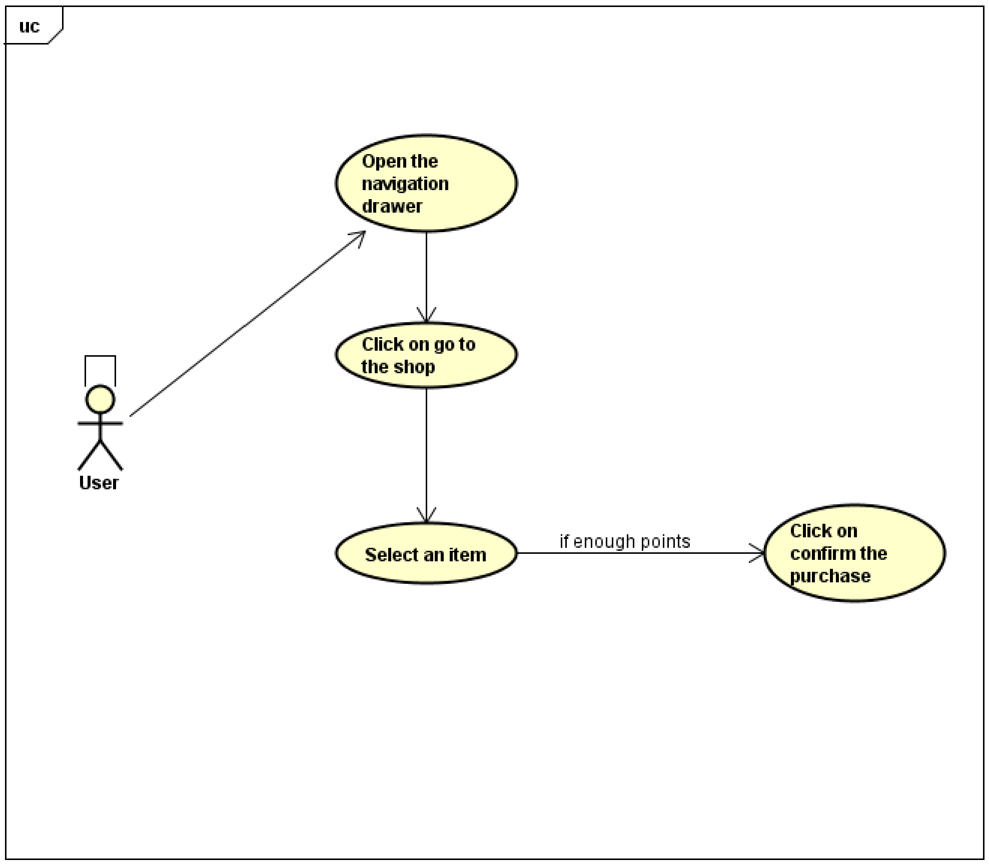


Figure 4: Use Case 4: how a user can cash his points for a reward

Activity Diagrams:

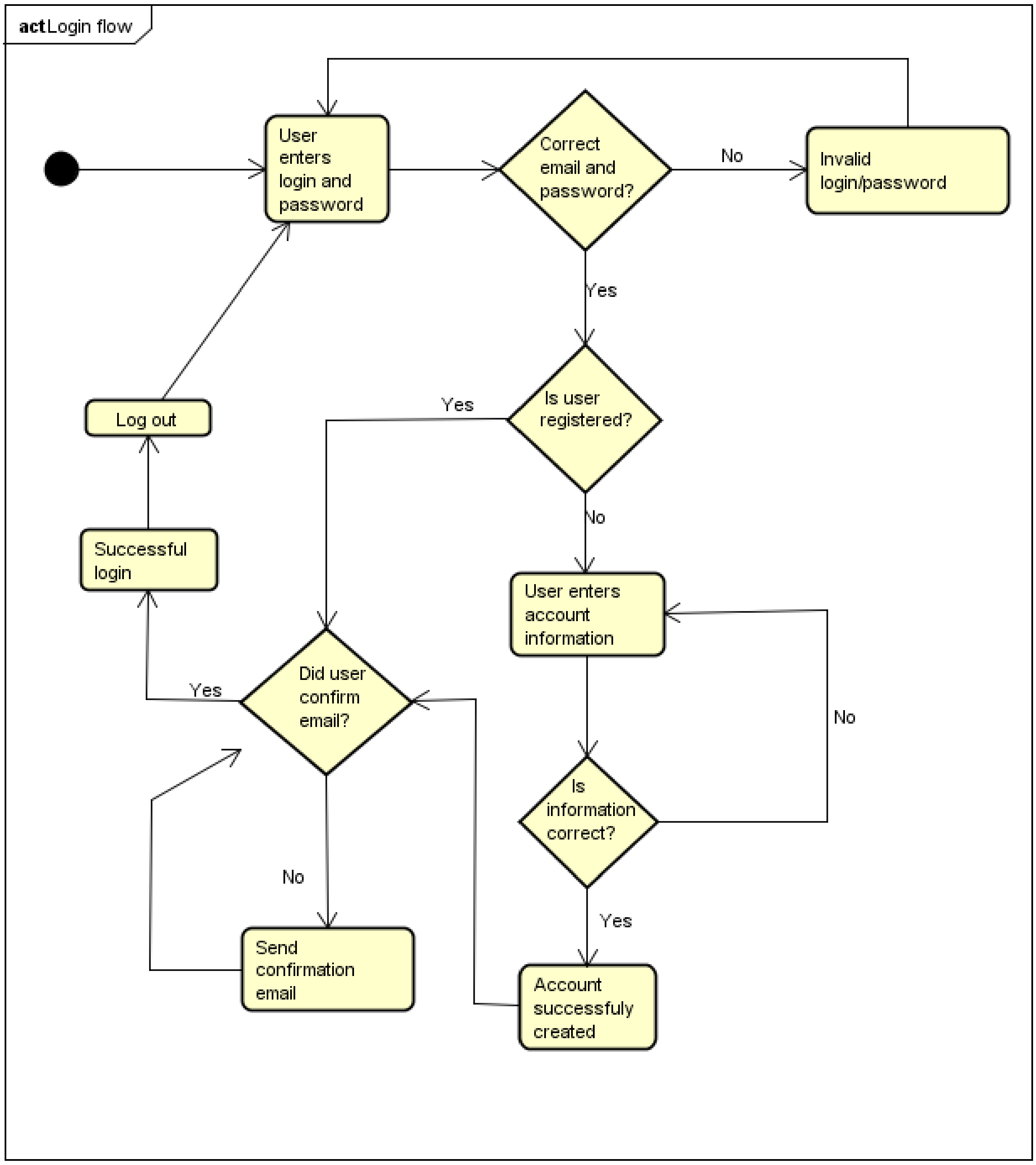


Figure 5: Activity Diagram 1: sign in on the application

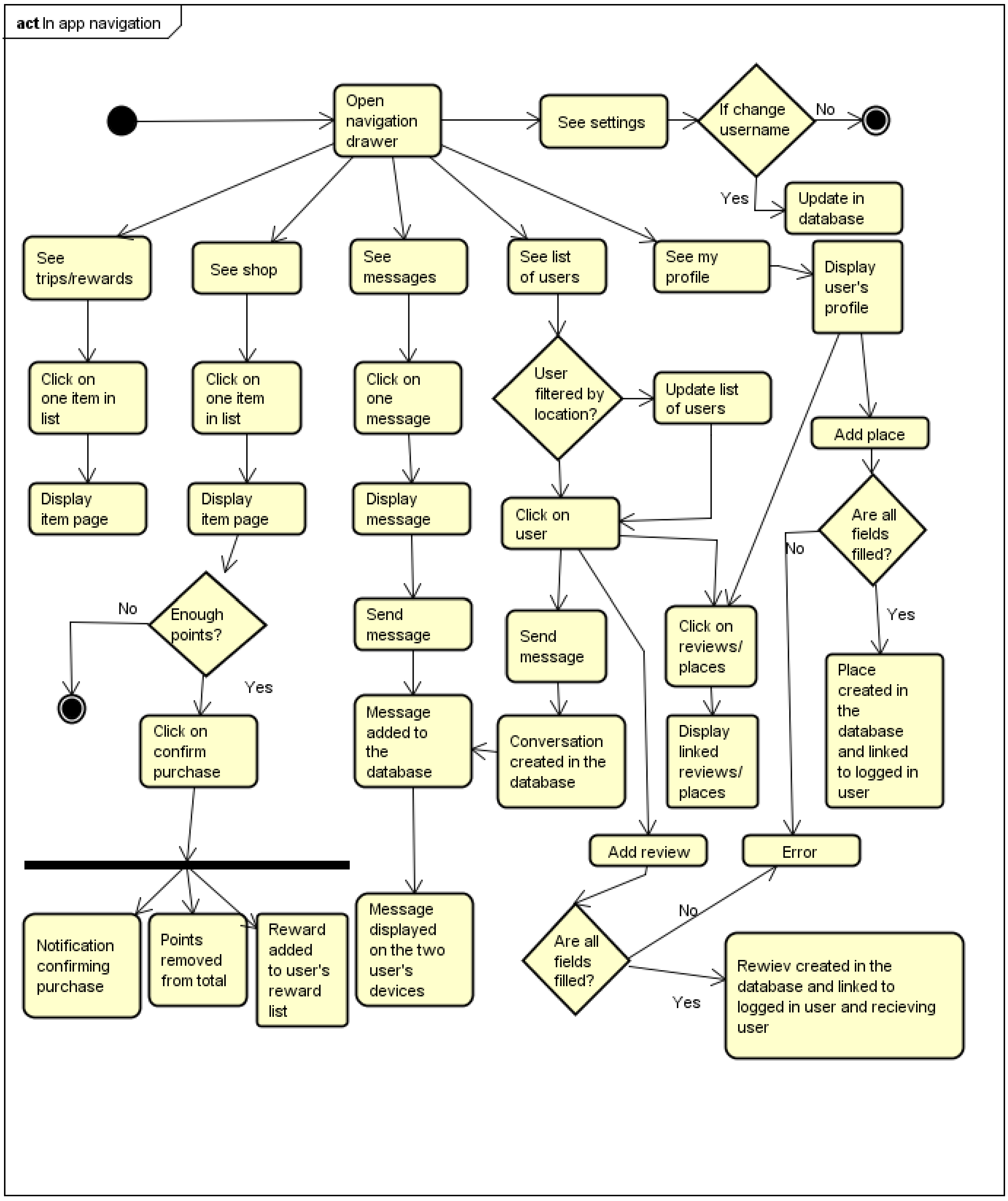

act In app navigation
Open navigation drawer
See settings
If change username
No
Yes
Update in database
See trips/rewards
See shop
See messages
See list of users
See my profile
Display user's profile
Click on one item in list
Click on one item in list
Click on one message
User filtered by location?
Update list of users
Add place
Display item page
Display item page
Display message
Click on user
Are all fields filled?
No
Yes
Enough points?
No
Yes
Send message
Send message
Click on reviews/ places
Place created in the database and linked to logged in user
Click on confirm purchase
Message added to the database
Conversation created in the database
Display linked reviews/ places
Add review
Error
Notification confirming purchase
Points removed from total
Reward added to user's reward list
Message displayed on the two user's devices
Are all fields filled?
No
Yes
Rewiev created in the database and linked to logged in user and recieving user

Figure 6: Activity Diagram 2: navigation from the navigation drawer

The second activity diagram groups some activities together. The activities that are merged together on the diagram are distinct activities, but have the same behavior and thus can be grouped together. This is done to avoid this diagram to become too complicated to read.

3.1.2 Non-Functional Requirements

Security

- Password requirements (set up by Firebase)
- Password security (hashed)
- Password not saved unencrypted
- If password lost, email to make a new one (handled by Firebase)

Usability

- User friendly
- Possible translation of the application due to avoiding hard-coded strings

4 Analysis

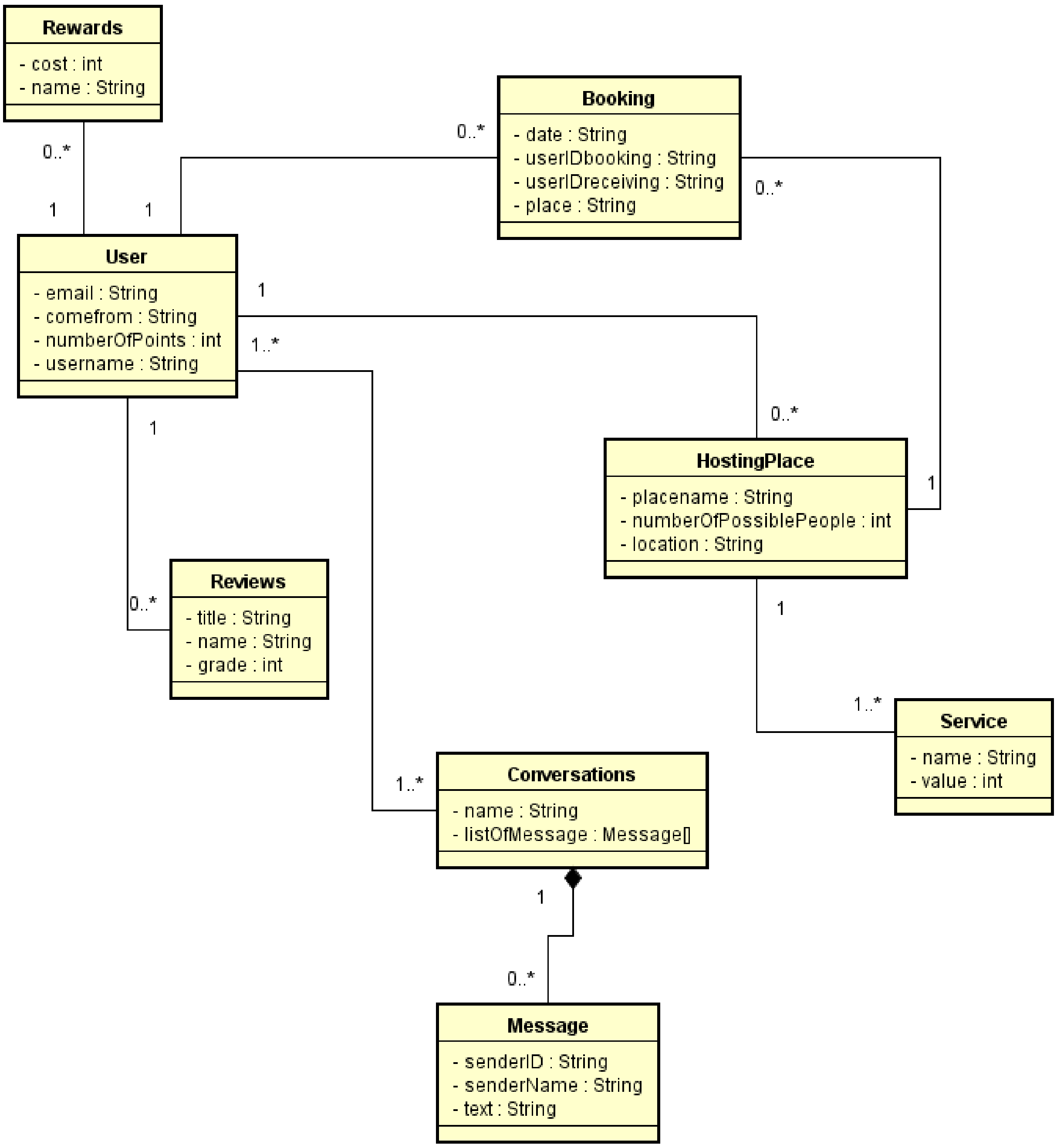


Figure 7: Domain model

The domain model represents the basic structure that we want our application to have and the requirements that we want it to be able to complete.

5 Design

5.1 Architecture

We used the Adapter in Android which is close to a Pattern Adapter.

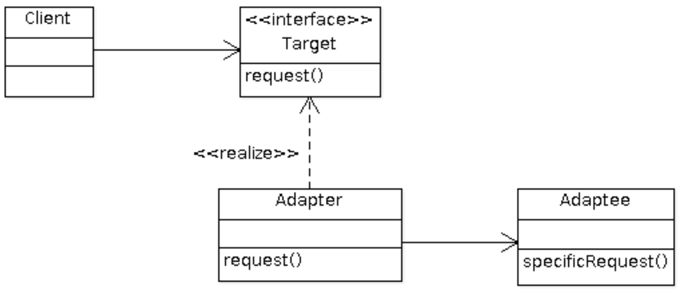


Figure 8: Adapter Pattern Template (*softwarepassion.com*)

We used those Adapters to populate the listViews with ArrayList and make clickable. Those Adapters are customizable, and help to display the content of the Objects that are kept in the database

We also wanted to use a MVC (Model View Controller), which is a MVP (Model View Presenter) but the scope of the project was too short to implement it correctly, and we didn't studied it during the Android course.

5.2 Technologies

Surfing Couch uses different technologies :

To do deal with all of the data manipulated by the users, the applications uses **Firebase** as database :

- Pros :
  - Firebase provides an easy implementation on an Android application, there is not that much of a setup to do, the read and write functions are really easy to implement
  - Firebase use JSON which permits an implementation of data in a fairly effortless way. That create a database that is flexible, well-structured and easy to read to some extent.
  - Firebase gives useful tools, including a user authentication that deals with the security of database thanks to its data hashing. Firebase is also loaded with email confirmation, data analytics, and stability tools, which helps to make the user experience better.
  - Made by Google
- Cons :
  - When the data structure starts to be a bit complex, it becomes quite hard to navigate through all of the data. Especially if you have a lot of data.
  - The querying and indexing is very limited due to the use of the JSON. It makes the search of data is very redundant.
  - Firebase is a bit high level, so you don't feel you have that much of control over it
  - Made by Google

One of the alternative to Firebase could be to use a more "classical" database management system like **SQLite** :

- Pros :
  - Classical way to store data, you can do SQL queries on it to retrieve data with more efficiency. Include also a good indexing of the data.
  - Is better than Firebase at storing huge load of data.
- Cons :
  - You don't have the tools proposed by Firebase, so you can really analyze your data usage and you must make all of the part of the security by yourself
  - Harder to use and implement than Firebase.

The application must be developed natively on Android, so **Android Studio** was a default choice to develop the application. But it's interesting to point out that there is alternatives to Android Studio such as **Xamarin** or **Cordova**.

5.3 Design Patterns, Class Diagram, Sequence Diagrams

Like said earlier, we are not using a specific design pattern.

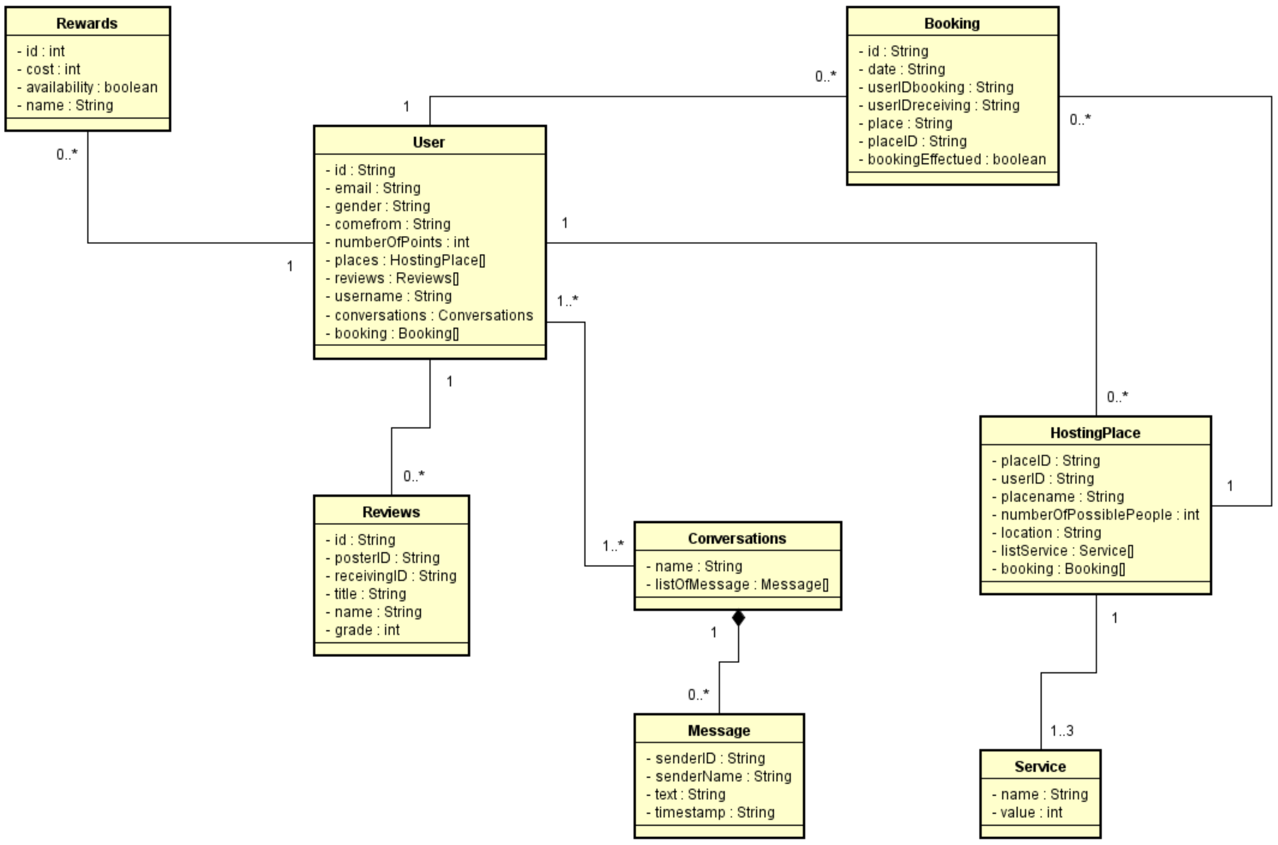
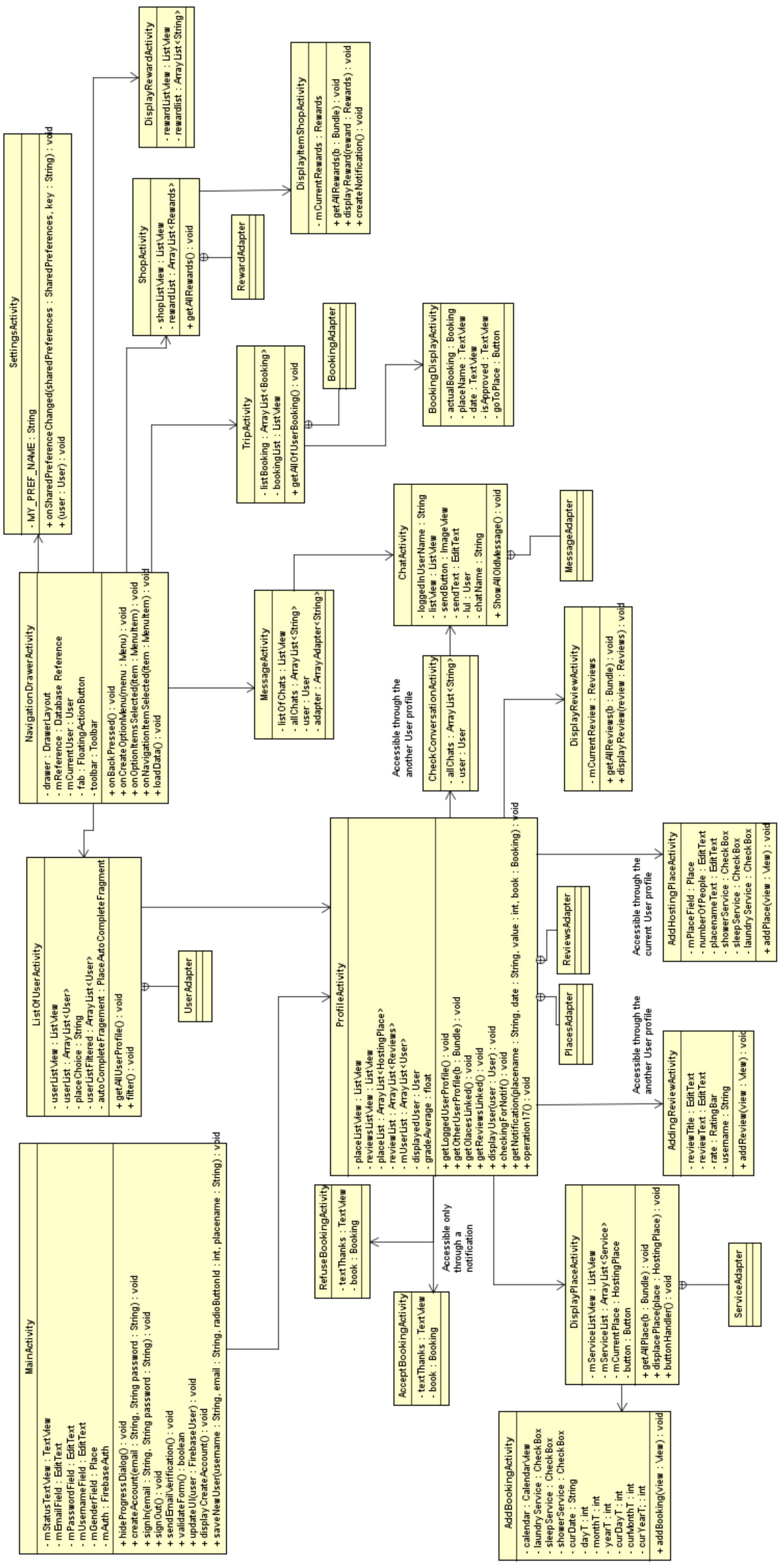


Figure 9: Class Diagram of the database

The database[1] is pretty straightforward, everything is based around the User. However, we must highlight some details. By default, when a user is created, the bookings are null, so not present in the database JSON. The list is created with the first booking he makes.

Also, a default chat is assigned to each user which is the "General Chat" so can talk all together. The other List are defined as "undefined" and taken as HashMaps inside of the application.

[1]We made a class diagram for the database because the data is kept in a JSON



SettingsActivity
- MY_PREF_NAME : String
+ onSharedPreferenceChanged(sharedPreferences : SharedPreferences, key : String) : void
+ (user : User) : void
DisplayRewardActivity
- rewardListView : ListView
- rewardlist : ArrayList<String>
DisplayItemShopActivity
- mCurrentRewards : Rewards
+ getAllRewards(b : Bundle) : void
+ displayReward(reward : Rewards) : void
+ createNotification() : void
ShopActivity
- shopListView : ListView
- rewardList : ArrayList<Rewards>
+ getAllRewards() : void
RewardAdapter
BookingAdapter
TripActivity
- listBooking : ArrayList<Booking>
- bookingList : ListView
+ getAllOfUserBooking() : void
BookingDisplayActivity
- actualBooking : Booking
- placeName : TextView
- date : TextView
- isApproved : TextView
- goToPlace : Button
NavigationDrawerActivity
- drawer : DrawerLayout
- mReference : Database Reference
- mCurrentUser : User
- fab : FloatingActionButton
- toolbar : Toolbar
+ onBackPressed() : void
+ onCreateOptionMenu(menu : Menu) : void
+ onOptionItemsSelected(item : MenuItem) : void
+ onNavigationItemSelected(item : MenuItem) : void
+ loadData() : void
MessageActivity
- listOfChats : ListView
- allChats : ArrayList<String>
- user : User
- adapter : ArrayAdapter<String>
ChatActivity
- loggedInUserName : String
- listView : ListView
- sendButton : ImageView
- sendText : EditText
- lul : User
- chatName : String
+ ShowAllOldMessage() : void
MessageAdapter
Accessible through the another User profile
CheckConversationActivity
- allChats : ArrayList<String>
- user : User
DisplayReviewActivity
- mCurrentReview : Reviews
+ getAllReviews(b : Bundle) : void
+ displayReview(review : Reviews) : void
ListOfUserActivity
- userListView : ListView
- userList : ArrayList<User>
- placeChoice : String
- userListFiltered : ArrayList<User>
- autoCompleteFragement : PlaceAutoCompleteFragment
+ getAllUserProfile() : void
+ filter() : void
UserAdapter
ProfileActivity
- placeListView : ListView
- reviewsListView : ListView
- placeList : ArrayList<HostingPlace>
- reviewList : ArrayList<Reviews>
- mUserList : ArrayList<User>
- displayedUser : User
- gradeAverage : float
+ getLoggedUserProfile() : void
+ getOtherUserProfile(b : Bundle) : void
+ getOlacesLinked() : void
+ getReviewsLinked() : void
+ displayUser(user : User) : void
+ checkingForNotif() : void
+ getNotification(placename : String, date : String, value : int, book : Booking) : void
+ operation17() : void
ReviewsAdapter
PlacesAdapter
Accessible through the current User profile
AddHostingPlaceActivity
- mPlaceField : Place
- numberOfPeople : EditText
- placenameText : EditText
- showerService : CheckBox
- sleepService : CheckBox
- laundryService : CheckBox
+ addPlace(view : View) : void
Accessible through the another User profile
AddingReviewActivity
- reviewTitle : EditText
- reviewText : EditText
- rate : RatingBar
- username : String
+ addReview(view : View) : void
MainActivity
- mStatusTextView : TextView
- mEmailField : EditText
- mPasswordField : EditText
- mUsernameField : EditText
- mGenderField : Place
- mAuth : FirebaseAuth
+ hideProgressDialog() : void
+ createAccount(email : String, String password : String) : void
+ signIn(email : String, String password : String) : void
+ signOut() : void
+ sendEmailVerification() : void
+ validateForm() : boolean
+ updateUI(user : FirebaseUser) : void
+ displayCreateAccount() : void
+ saveNewUser(username : String, email : String, radioButtonId : int, placename : String) : void
RefuseBookingActivity
- textThanks : TextView
- book : Booking
AcceptBookingActivity
- textThanks : TextView
- book : Booking
Accessible only through a notification
DisplayPlaceActivity
- mServiceListView : ListView
- mServiceList : ArrayList<Service>
- mCurrentPlace : HostingPlace
- button : Button
+ getAllPlace(b : Bundle) : void
+ displacePlace(place : HostingPlace) : void
+ buttonHandler() : void
ServiceAdapter
AddBookingActivity
- calendar : CalendarView
- laundryService : CheckBox
- sleepService : CheckBox
- showerService : CheckBox
- curDate : String
- dayT : int
- monthT : int
- yearT : int
- curDayT : int
- curMonthT : int
- curYearT : int
+ addBooking(view : View) : void

Figure 10: Class Diagram of the application Activities

All of the activities apart from the MainActivity and the Adapter are extending the NavigationDrawerActivity which is itself extending AppCompatActivity.

A user can add review only once per User, and some Display activities are accessible according if you are on your profile or someone else profile

Each Activities have an OnCreate() function and other (OnStart(), OnStop()), since those methods are part of the life cycle there aren't displayed.

Each Adapter have overridable methods, that are also not displayed on the diagram

6 Implementation

```
        sendButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (sendText.getText().toString().trim().length() == 0) {
            Toast.makeText(ChatActivity.this, R.string.chat_empty_text,
                Toast.LENGTH_SHORT).show();
        } else {
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy - HH:mm",
                Locale.FRANCE);
            String currentDateAndTime = sdf.format(new Date());
            FirebaseDatabase.getInstance().getReference().child("Conversation/" +
                chatName + "/listOfMessages").push().setValue(new Message
                    (
                    FirebaseAuth.getInstance().getCurrentUser().getUid()
                    , lul.getUsername()
                    , sendText.getText().toString()
                    , currentDateAndTime
                    )
            );
            sendText.setText("");
        }
    }
});
```

This is an example of how the writing a message is made.

At first, we are checking if the input field of the user is empty or not. We have to trim it in case of the User just put a blank space.

If the input field is not empty, we can fetch the date and the time at the moment the send button was pressed, and we write all of the relevant data into the database and then we empty the input field.

To write in the database, we have to get a reference on the database and then look for where we want to write the data which here is: ***Ćonversation¿chat_name¿listOfMessages´***

The push value will add a new instance of the object **Message** that contains a UserID, a username, the user text and a timestamp.

```
public class Message {
    protected String senderID;
    private String senderName;
    private String text;
    private String timestamp;
    }
```

7 Test

7.1 Test Specifications

Because of the specification of our application and the fact that we did not go through testing during class, we decided to go with black box testing and a user testing, because our application is supposed to have a focus on an intuitive user interface.

7.2 Black Box Testing

Test	**Expected Result**	**Actual Result**	**Status**
Login	Success	Success	Passed
Login with wrong password	Password not matching the password in the database	Password not matching error	Passed
Login with password too short	Password less than 6 characters: error	Password too short error	Passed
Email address not correct	Email address not correctly formatted	Email address formatted badly error	Passed
Email address not already registered	Can't sign in with given email address	No user corresponding to this email address	Passed
Create account not allowed if address is already registered	Can't create account if already have an account	Email address already in use with another account	Passed
Create an account without filling all the fields	Not allowed to create an account	Error because all the fields are not filled	Passed
Go back to sign in activity after signing in by using the back button and not the logout button	Application closing	Application closing	Passed
Confirm email address	The user has to confirm their email address before having access to the application	It is not possible for the user to access the application without having confirmed their email address	Passed
Display User's profile	Display profile of user with data saved in the database	Display profile of user with data saved in the database	Passed
Have only button "add place" on logged user's profile	Button "write message" not displayed on logged user's profile	Button "write message" not displayed on logged user's profile	Passed

Test	Expected Result	Actual Result	Status
Add a place without filling all the fields should not be allowed	Not saving into database and output an error if the user attempts to create a place without all the fields	Error outputs as a toast and the data is not saved until all the fields are completed	Passed
Display all of my trips	Display list of my bookings, if list is empty, display nothing. Avoid crashing the application	The application is not crashing even if there is no booking	Passed
Display all of my rewards	Display list of my rewards, if list is empty, display nothing. Avoid crashing the application	The application is not crashing even if there is no rewards	Passed
Display the items in the shop with Grey background and not clickable if item is not available	The non-available items in the shop should not be activated	The non-available items in the shop are with Grey background and can't be clicked	Passed
Allowing buy of items in the shop only if the amount of points the user has is enough	If the user does not have enough points, the button for the buy is not activated	Button not activated if the user does not have enough points	Passed
Allowing the purchase if item available and user has enough points	The item can be purchased	The item ca be purchased	Passed
Removing the points from the total amount of points the user has when a purchase is made	The points are deduced from the amount of points the user has	The point are correctly deduced	Passed
Notifying the user that the purchase has been made	Give an alert to the user to tell them that the purchase was successful	Send a notification once a purchase has successfully been made to inform the user	Passed
Check the user's messages	The logged in user should be able to get a list of the messages he got	The list of messages is accessible to the user via the navigation drawer	Passed
Send a message to another user via an existing conversation	The conversation should update on both phones and display correctly	The conversation displays on both phones and refreshes once a message is sent	Passed
List all the users that are using the application	Being able to see all the users in a list	The users are all displayed in a list	Passed

Test	Expected Result	Actual Result	Status
Filter users in a specific area	We have to be able to filter the user list to keep only users in one area - to be able to look for a host for your next trip	The filter allows you to filter using a Google Map location	Passed
Send a message to another user	The users have to be able to send a message to another user by navigating to the user's profile	The button "send a message" is displayed and allows to send a message to the user - if a conversation exists already, this one is opened.	Passed
Leave a review to an user	Being able to review another user	It is possible to review an user using the button on their profile	Passed
Review another user multiple times	It should be possible to review someone just once to avoid fake reviews	If a review has already been added for that user, the system does not allow the user to send a new review	Passed
Send an empty review	It should not be possible to send a review without filling out all the fields	All the fields have to be filled for the review to be sent	Passed
Display the user's data in the setting activity	The settings page has to display the correct data	The settings page displays the data that is in the database	Passed
Change the settings	The user has to be able to change their username in the settings	When the user changes their username in the settings, it changes it in the database and refresh the data displayed on the settings page	Passed
Log out of the application	The user has to be able to log out successful from the application and be redirected to the login screen	The user is logged out and redirected to the login screen	Passed
Pressing the back button after logging in	The user should not be able to go back to the inside of the application without logging in again	If the user tries to go back after logging out, the application stops	Passed

7.3 User test

8 Results & Discussion

9 Conclusion