GroupUp
Project Feasibility & Analysis Report

Author: Mehmet Alper Genç
June 2020

1. INTRODUCTION

GroupUp is a web application that lets its users form groups for activities they would like to do. The selection of groups is based on geographical location, so the users can pick groups that they can communicate easily. It also has a report system to control maladaptive behavior and a chat system to ensure that group members coordinate with each other successfully.

This report will first argue the feasibility of this project. Afterwards, the report will analyze the project by giving an overview of the system, listing its functional and non-functional requirements, and finally, it will provide the system models related to the analysis.

2. FEASIBILITY

2.1. Target Audience

The target audience of this project is the people looking to join, or form, small but coordinated groups for tasks, activities or events. The types of groups that this project mainly aims to form are groups that require experience or skill on a subject, as groups that do not require such an amount of experience or skill could be formed without the usage of such an application. However, in this application, there are no boundaries on whatever group a user wants to form, provided they obey the law and follow the rules of ethics, so any kind of group can be formed, but those seeking skill or experience based groups will benefit the most from this application. Furthermore, another group that can benefit from this application is those who are trying to adapt to a new city or country. These people can seek out social platforms and/or guides to help them settle into their new place of life.

2.2. Comparison to Other Applications

2.2.1. Meetup

Meetup (https://www.meetup.com/) is a web platform that offers a service similar to what my project aims to offer.

What I believe is a key advantage of my project, compared to Meetup, is that Meetup can handle both small and large groups, but my project focuses specifically on smaller groups that require more coordination. This is mainly because of the rating system and the trust points that I aim to implement, which are explained in the “Overview” section of my software analysis later on in this report.

A disadvantage that my project has compared to Meetup is that my project does not have good scalability, as there is only a finite amount of data only one database can hold.

2.2.2. Specific Applications

There are specific group finder applications in the market that cater to a specific group of people sharing the same goal or interest. One example to this could be an app named “Destiny 2 Companion,” an app built by Bungie so that the players of their video game “Destiny 2” can find groups to play together. Another such example is “Via,” a mobile application for ride-sharing (https://ridewithvia.com/). Users of Via can share rides with people who will be travelling the same route as them.

One advantage of my project, compared to specific applications like these, is that my project allows for a wider range of types of groups, as it lets users specify what kind of group they are looking to form, while the specific group finder applications let their users only join or form a distinct type of group, such as video game groups for Destiny 2 Companion and ride-sharing groups for Via.

One disadvantage of my project is in the terms of security. While the more specific group finder applications serve fewer types of groups, they provide more security. Continuing on the examples given above, “Destiny 2 Companion” lets its users see the skill level and experience of their group members, so that they can safely choose their group members. “Via” on the other hand, has its own employed and experienced drivers, so the users can trust the driving skills of their driver. To counteract this disadvantage in my project, I aim to provide a “trust points” system to show how well a user performed their previous duties, and an in-group chat system so that the users can communicate with potential members before setting out to do what they plan to do.

3. ANALYSIS

3.1. Overview

3.1.1. Users

The object representation of a user is the smallest building block in this system. A user has a username, a password, an email address, and contact information.

Furthermore, they have security levels and trust points. These work as follows: When a user initially signs up for this service, they are in Security Level 0. In this level, they can only see the group requests. They cannot apply to any group, or see the any user info other than their username. To pass to Security Level 1, they must prove that they are not a robot using captcha, and also provide email verification. In Security Level 1, they can apply to group requests and see the public information of other users. Contact information of a user is only shown to another user if they are currently in a group. Those in security level 1 can not create group requests. To pass to Security Level 2, they must have gained a bit of experience as

a good member of the group, meaning they must have accumulated enough trust points. In Security Level 2, they can create group requests.

Trust points work as follows: Once a group is closed, the users have to rate the other users in their group on how trustworthy they have been, and how successfully they fulfilled their role in the group. A positive rating will increase a user's trust points, and a negative rating will decrease them. These trust points will also be shown as public information in the users' profiles to indicate how well a group member they previously were.

3.1.2. Groups

As previously mentioned, the purpose of this system is to let people create groups. In the application domain, these groups have a title, a description, a maximum user capacity, location information, and a list of users currently in the group. A user can only view or join a group request if they are in the location that the group is created for. Those that are in the same group with another user will have whichever contact information they desire to make public shared with that user. Furthermore, the groups have a chat system for communication. A user that joins the group later will not be able to see previous messages.

3.1.3. Report System & Moderators

To reduce, and possibly eliminate maladaptive behavior, the application will have a report system. If a user finds that a group or another user is not behaving in a desirable manner, they can report this user or group so that action may be taken against them if necessary. The object representation of a report has its user, the user/group being reported, and the reasoning behind the report as a string, read from the user.

At this point, another type of user should be mentioned, which is the moderator. The moderator, while still being able to do everything another user can, also has the power to remove groups or punish individuals who are behaving undesirably. A moderator can also report as a normal user can, but for the purpose of fairness, the report concerning a moderator can only be viewed by another moderator.

3.2. Functional Requirements

The system should initially let the users:

- Sign up
- Log in
- View a group request
- Report group
- Attempt to proceed to the next Security Level

For users that have a security level of 1, the system should also let them:

- Join a group
- Leave a group
- See public information about a user in their group
- Report a user
- Chat with group members

For users that have a security level of 2, the system should also let them:

- Create a group
- Remove a user from a group they created
- View, edit or delete a group they previously created

If the user is a moderator, they have additional clearance to:

- View a report
- Block the user from accessing the system for a given amount of time
- Ban the user (i.e. block them indefinitely)
- Lift the punishment of a user
- Remove a group request
- Delete a report

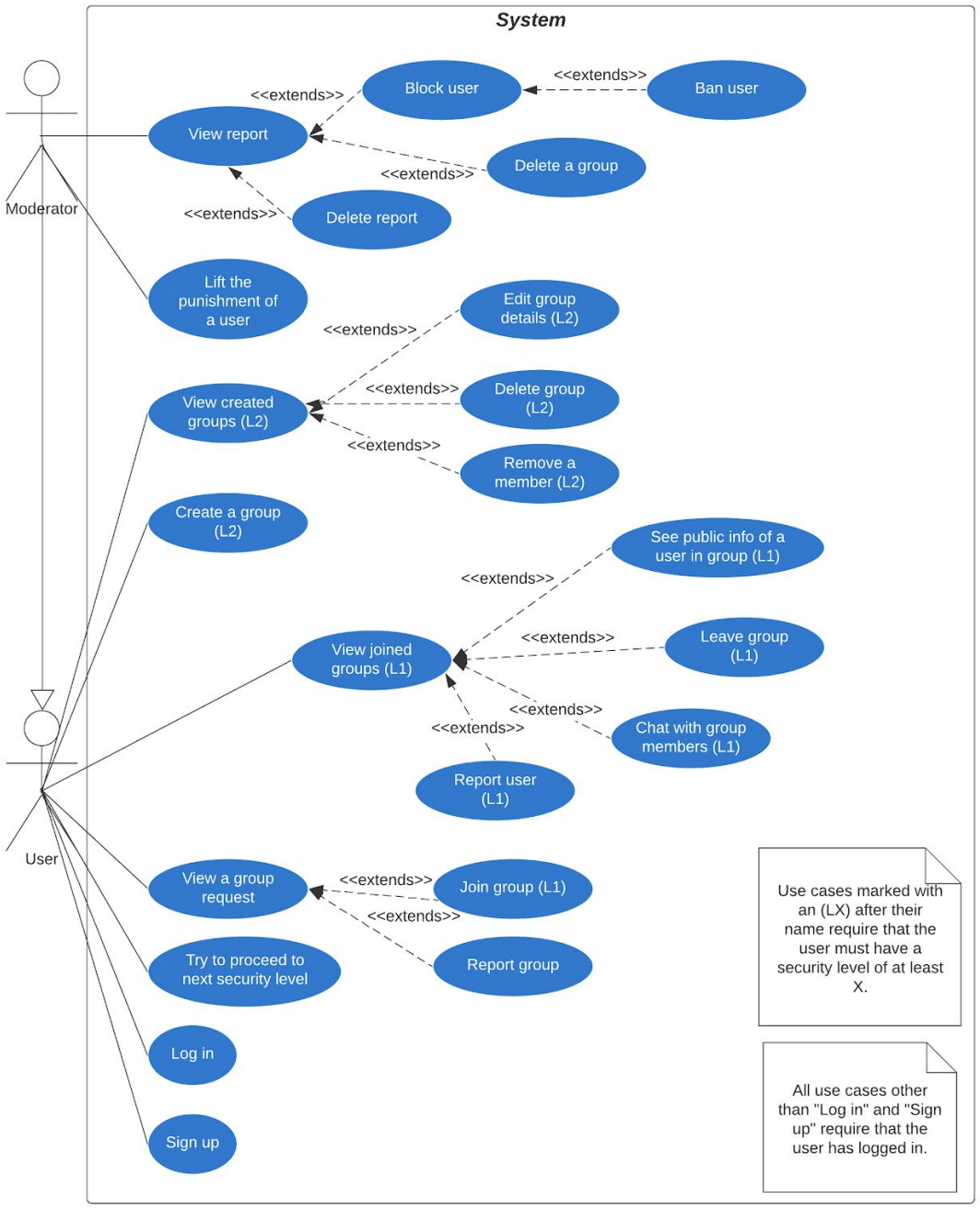
3.3. Non-functional Requirements

These are the non-functional requirements I have chosen for this project:

- **Security:** As mentioned in the Feasibility section above, this project lacks in security compared to other platforms. So, in order to make this project as safe as possible, I have decided to implement a “Security Level” system and a “Trust Points” system so that the users can have an understanding of what kind of a group member will another member be, and decide whether or not to keep them in their group.
- **Maintainability:** Since this application is a social platform, it should be maintainable to not disrupt any form of communication done through the application. To make the project as maintainable as possible, I will adhere to the object-oriented programming standards to make sure errors are caught and fixed easily, and future developers can add changes without having to change a large portion of the code.
- **Usability:** Again, since this application is a social platform, it should have a simple and easy-to-use UI, and the users should be able to form a general idea of the purpose of any clickable in the UI just by looking at the text on it.

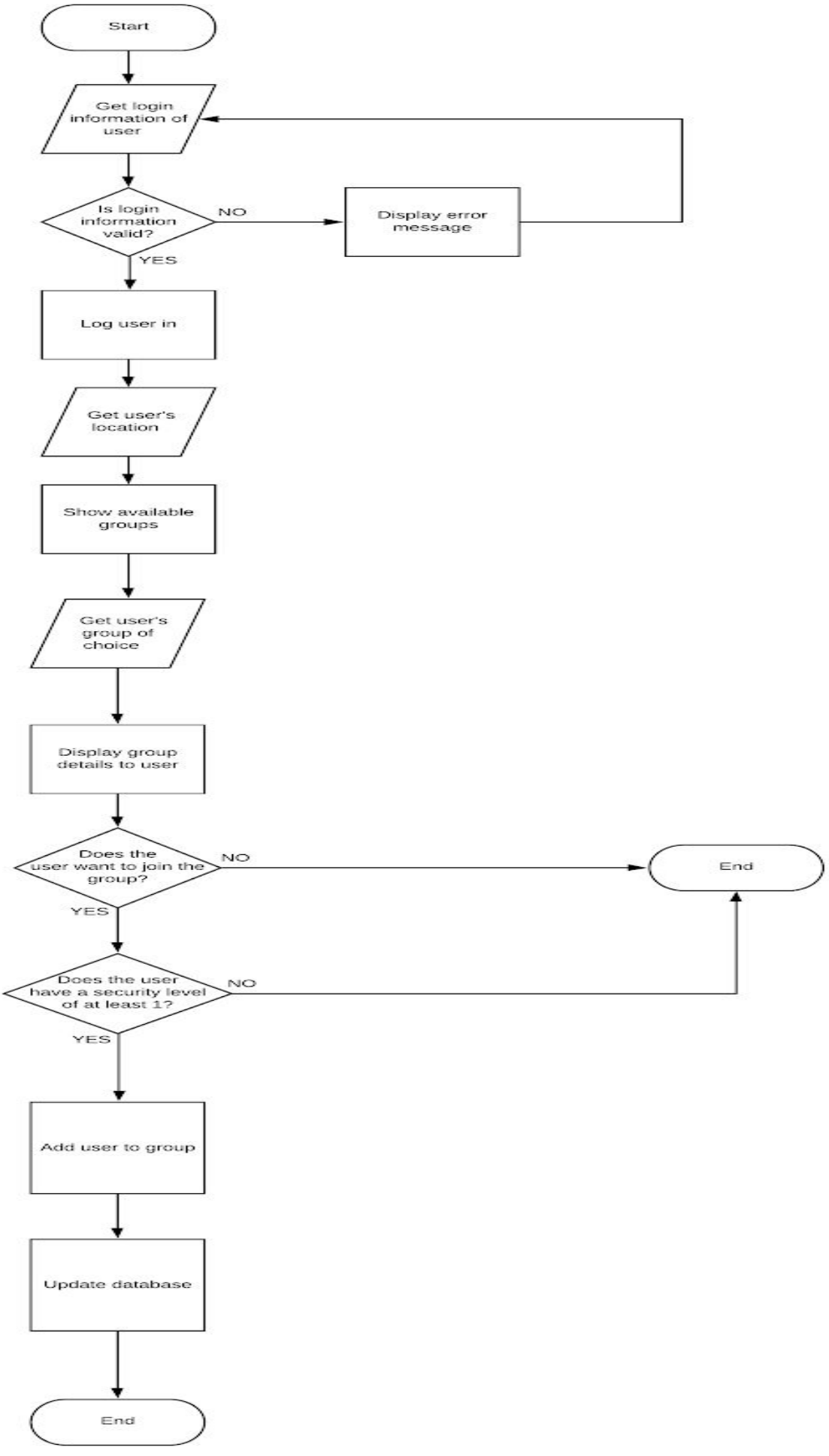
3.4. System Models

3.4.1. Use Case Diagram



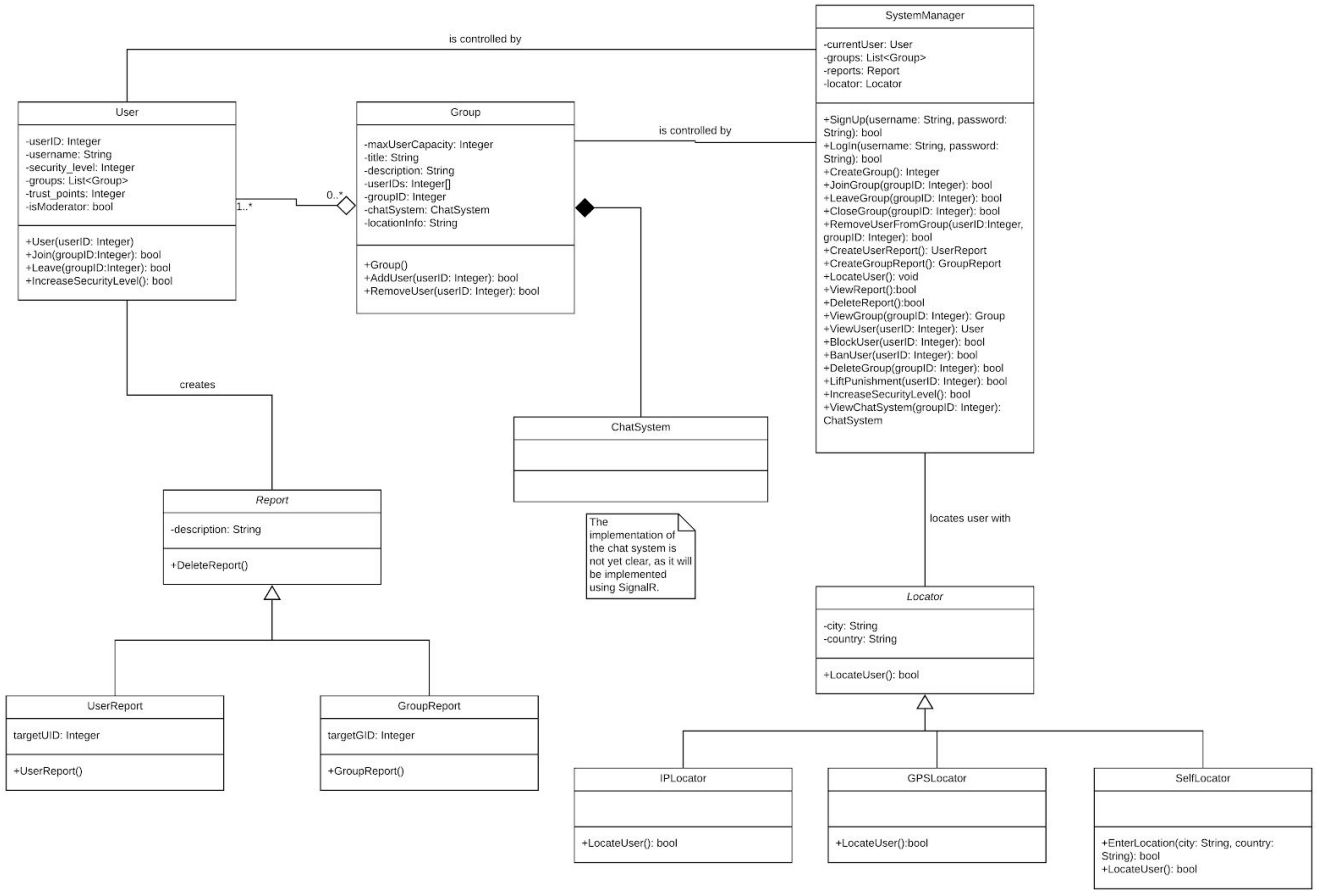
3.4.2. Flowchart

Since I could not put all the system's different functions into one flowchart, I decided to show its most common function: The sequence of logging in, receiving the user's location and showing matching group requests, and processing the join request of a user. The flowchart is in the next page.


Start
Get login information of user
Is login information valid?
NO
Display error message
YES
Log user in
Get user's location
Show available groups
Get user's group of choice
Display group details to user
Does the user want to join the group?
NO
End
YES
Does the user have a security level of at least 1?
NO
YES
Add user to group
Update database
End

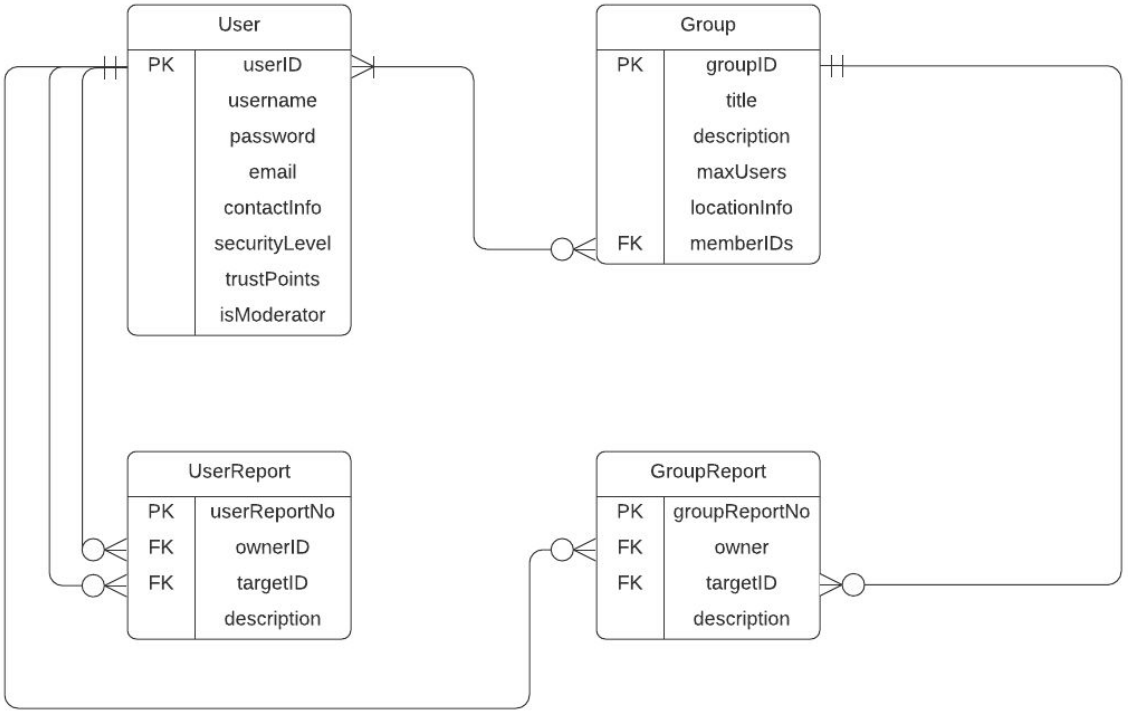
3.4.3. UML Class Diagram

The details described in this Class Diagram may change during the implementation phase.



If it is difficult to read the contents of this diagram, check Appendix A for a page-wide, horizontal version.

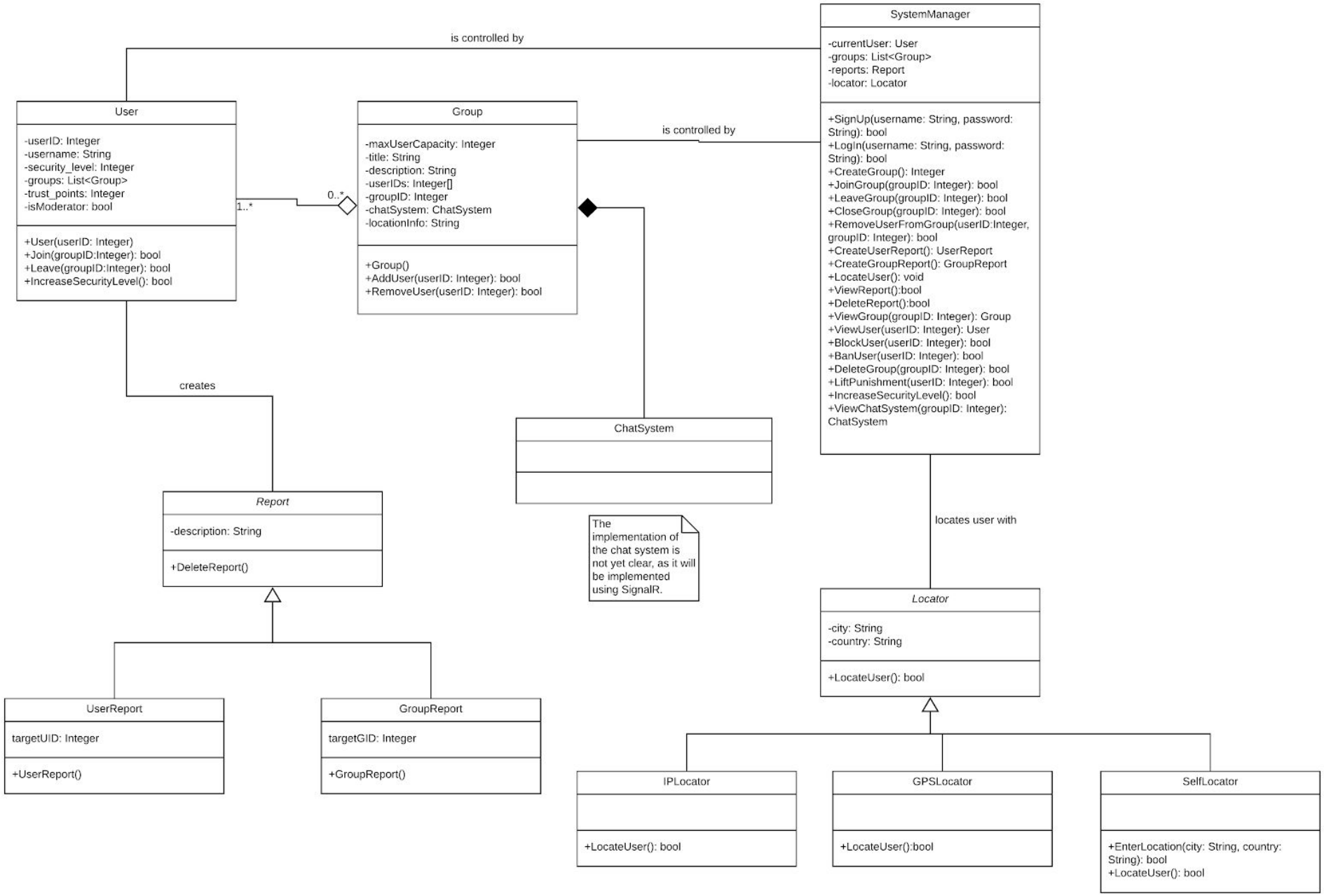
3.4.4. Entity Relationship Diagram



4. Appendix

Appendix A - Class Diagram

The enlarged version of the class diagram is in the next page.


SystemManager
-currentUser: User
-groups: List<Group>
-reports: Report
-locator: Locator
+SignUp(username: String, password: String): bool
+LogIn(username: String, password: String): bool
+CreateGroup(): Integer
+JoinGroup(groupID: Integer): bool
+LeaveGroup(groupID: Integer): bool
+CloseGroup(groupID: Integer): bool
+RemoveUserFromGroup(userID:Integer, groupID: Integer): bool
+CreateUserReport(): UserReport
+CreateGroupReport(): GroupReport
+LocateUser(): void
+ViewReport():bool
+DeleteReport():bool
+ViewGroup(groupID: Integer): Group
+ViewUser(userID: Integer): User
+BlockUser(userID: Integer): bool
+BanUser(userID: Integer): bool
+DeleteGroup(groupID: Integer): bool
+LiftPunishment(userID: Integer): bool
+IncreaseSecurityLevel(): bool
+ViewChatSystem(groupID: Integer): ChatSystem
is controlled by
User
-userID: Integer
-username: String
-security_level: Integer
-groups: List<Group>
-trust_points: Integer
-isModerator: bool
+User(userID: Integer)
+Join(groupID:Integer): bool
+Leave(groupID:Integer): bool
+IncreaseSecurityLevel(): bool
1..*
0..*
Group
-maxUserCapacity: Integer
-title: String
-description: String
-userIDs: Integer[]
-groupID: Integer
-chatSystem: ChatSystem
-locationInfo: String
+Group()
+AddUser(userID: Integer): bool
+RemoveUser(userID: Integer): bool
is controlled by
creates
ChatSystem
The implementation of the chat system is not yet clear, as it will be implemented using SignalR.
Report
-description: String
+DeleteReport()
locates user with
Locator
-city: String
-country: String
+LocateUser(): bool
UserReport
targetUID: Integer
+UserReport()
GroupReport
targetGID: Integer
+GroupReport()
IPLocator
+LocateUser(): bool
GPSLocator
+LocateUser():bool
SelfLocator
+EnterLocation(city: String, country: String): bool
+LocateUser(): bool