Bilkent University
Department of Computer Engineering

CS 491
Senior Design Project

High-Level Design Report

Project Name:

GrouPub

- A Location-based Chat Engine-

Group Members:

Arda Ekmekçi	21101065
Ayberk Aksoy	21100623
Ekin Karayalçın	21101919
Merve Tuncel	21102000
Seren Erdoğan	21100693

Supervisor: Fazlı Can
Jury Members: Selim Aksoy, Hakan Ferhatosmanoğlu
Expert: Mehmet Çakır

Project Website: http://groupub.github.io/

Table of Contents

Table of Figures

1. Introduction

1.1. Summary & Purpose of the System

GrouPub is a location based quiz application where users can sign up to an event, form or join a group and participate in a quiz event that is hosted in a specific pub or café.

Users need to create an account to be able to use GrouPub. After successfully creating an account and logging in, users will see a list of upcoming events. These events represent the actual quiz event that is taking place in a specific location. After selecting an event, users can see the groups that were already formed by other users that belong to the selected event. Users can either join a group that has an available slot or form a new group (maximum of five). Once the user successfully joins or forms a group, he/she can go the specified location at the specified time to join the quiz.

A quiz event will start at the specified time and users that have successfully signed up to that quiz (and present at the specified location) will start to receive questions every ten minutes. Users will have twenty seconds to answer a question. Once the quiz event is over, the winner group will be rewarded with drinks or food by the host and they will be recorded in GrouPub's public leaderboards. The winner group will also be prompted to take a photo of themselves and upload it GrouPub's leaderboards (this is an optional step for the group).

The purpose of GrouPub is to create an environment where even strangers can form groups, share a few drinks and socialize while trying to win a quiz event. It is an alternative activity that saves users from their daily routine, clear their heads, make some new friends and just have fun.

1.2. Design Goals

1.2.1. From the Perspective of Users

Usability: It should be easy to use this program since its main purpose is to provide fun. The users do not want to do so much work in leisure time activities. The application aims to be useful not to be boring.

User-friendly Interface: We will prefer a simple design because of our user target. We will try to apply Material Design features which are developed by Google to our user interface [1]. It will make our work much easier and our interface will be seen neat.

Low Cost: It will be a free application. Only cost is from ISP since it needs an internet connection to be played. The event place's WI-FI can be used to get rid of this cost, too.

Security: To join an event, a QR-code validation is required. Information stored in database (passwords and scores) should not be changed by others.

1.2.2. From the Perspective of Us

Performance: The game will be open for all users at the same time. The application uses web service database, hence the requests and responses should be shorten.

Memory: It must provide efficient queries to the database in order to use consume less RAM.

Updatability: Group information, leaderboards, personal records will be updated frequently.

1.3. Definitions Acronyms and Abbreviations

Event: An event represents a quiz event that is taking place in a specific location (pub, café etc.). It includes information about the quiz event including the start time and the location.

Firebase: An API that provides powerful services such as user authentication, static hosting and more.[2]

Group: A group (that is formed by a user) represents a group of maximum five users that belongs to a specific event.

Heat Level: Heat level of a user indicates how often that user uses words that are registered in the word blocker's database. Every time a restricted word is used, the heat level will increase by some amount. After reaching a certain threshold, that user will not be able to chat with anyone for a limited amount of time. Note that the heat level of a user will start to cool down if that user does not use a restricted word.

Host: Host is the location where an event will take place. A host can be a pub, a café or anywhere public with tables and chairs where people can sit together and solve quiz questions.

HTML: Hypertext Markup Language [3]

ISP: Internet Service Provider

JSON: JavaScript Object Notation [4]

Node.js: An API that provides services for scalable network applications. [5]

Phonegap: A framework that can run mobile applications. We will run/test our application using Phonegap. [6]

Rating: User A can rate user B based on how polite user B was to user A. The average rating of each user will be publicly displayed, giving a rough description of how polite each user is.

Spring: A framework that is used to create JVM-Based applications and systems. [7]

User: A user of GrouPub with a valid account.

Word Blocker: A class (will be written by us) that scans a conversation and censors each word that is registered in its database. The database of the word blocker will be created and maintained by us. The purpose of this class is to censor out insulting and inappropriate words and to help calculate the heat level of a user.

XML: Extensible Markup Language [8]

1.4. Overview

Our project is a mobile application both for Android and iOS operating systems which is uniquely designed for socializing and entertainment. It is especially for teenagers who want to compete and socialize. It is a quiz game which users can sign up to an event, form a group with their friends or others and participate in a quiz that is hosted in a pub or a café.

The purpose of GrouPub is to create an environment where even strangers can form groups, share a few drinks and socialize while trying to win a quiz event. GrouPub is a unique application since there is no application that covers the concept of chatting with others in a specified area and creating fun quiz environments in different locations with multiple users to compete with.

2. Current Software Architecture

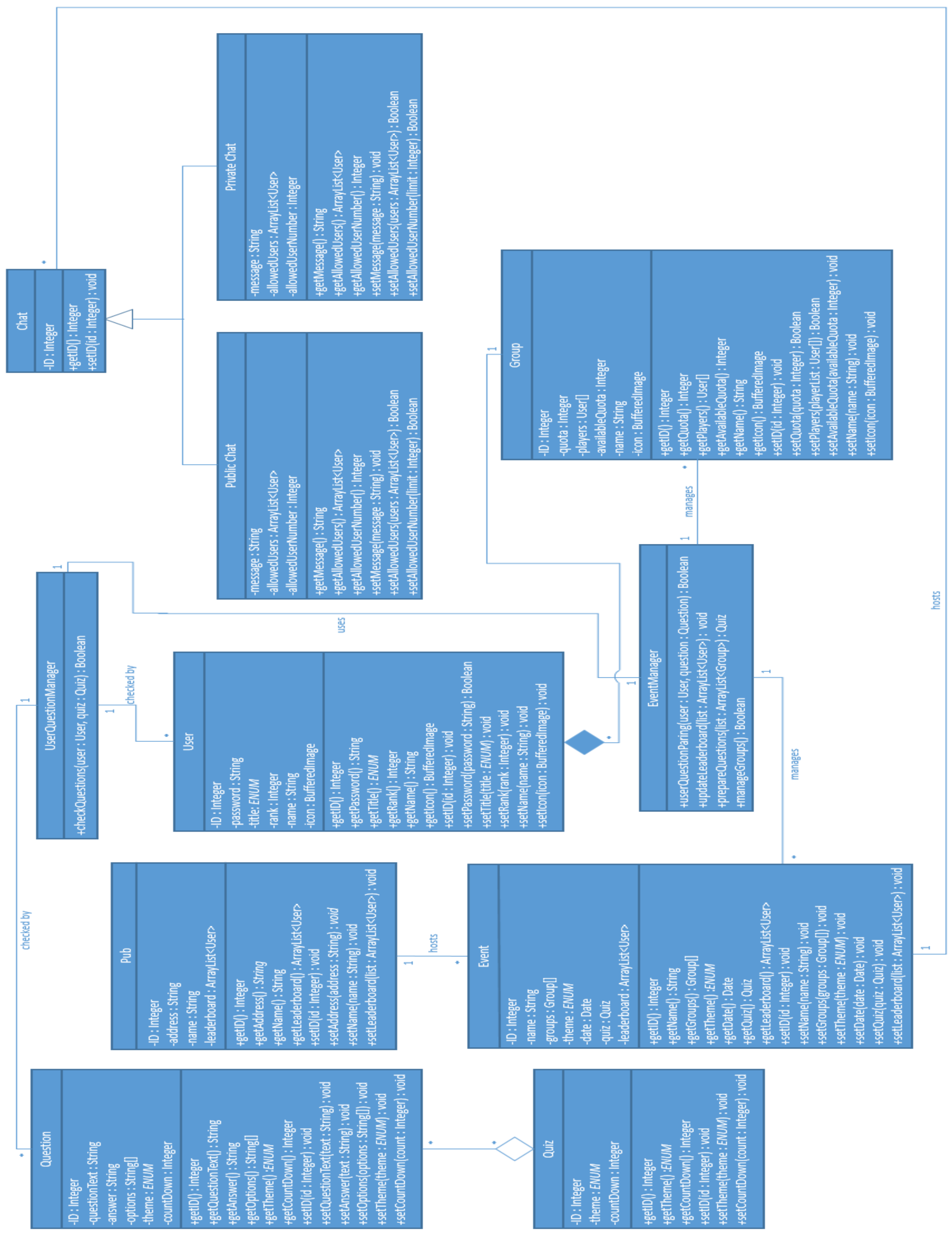
Chat
-ID : Integer
+getID() : Integer
+setID(id : Integer) : void
Private Chat
-message : String
-allowedUsers : ArrayList<User>
-allowedUserNumber : Integer
+getMessage() : String
+getAllowedUsers() : ArrayList<User>
+getAllowedUserNumber() : Integer
+setMessage(message : String) : void
+setAllowedUsers(users : ArrayList<User>) : Boolean
+setAllowedUserNumber(limit : Integer) : Boolean
Public Chat
-message : String
-allowedUsers : ArrayList<User>
-allowedUserNumber : Integer
+getMessage() : String
+getAllowedUsers() : ArrayList<User>
+getAllowedUserNumber() : Integer
+setMessage(message : String) : void
+setAllowedUsers(users : ArrayList<User>) : Boolean
+setAllowedUserNumber(limit : Integer) : Boolean
Group
-ID : Integer
-quota : Integer
-players : User[]
-availableQuota : Integer
-name : String
-icon : BufferedImage
+getID() : Integer
+getQuota() : Integer
+getPlayers() : User[]
+getAvailableQuota() : Integer
+getName() : String
+getIcon() : BufferedImage
+setID(id : Integer) : void
+setQuota(quota : Integer) : Boolean
+setPlayers(playerList : User[]) : Boolean
+setAvailableQuota(availableQuota : Integer) : void
+setName(name : String) : void
+setIcon(icon : BufferedImage) : void
UserQuestionManager
+checkQuestions(user : User, quiz : Quiz) : Boolean
User
-ID : Integer
-password : String
-title: ENUM
-rank : Integer
-name : String
-icon : BufferedImage
+getID() : Integer
+getPassword() : String
+getTitle() : ENUM
+getRank() : Integer
+getName() : String
+getIcon() : BufferedImage
+setID(id : Integer) : void
+setPassword(password : String) : Boolean
+setTitle(title : ENUM) : void
+setRank(rank : Integer) : void
+setName(name : String) : void
+setIcon(icon : BufferedImage) : void
EventManager
+userQuestionParing(user : User, question : Question) : Boolean
+updateLeaderboard(list : ArrayList<User>) : void
+prepareQuestions(list : ArrayList<Group>) : Quiz
+manageGroups() : Boolean
Pub
-ID : Integer
-address : String
-name : String
-leaderboard : ArrayList<User>
+getID() : Integer
+getAddress() : String
+getName() : String
+getLeaderboard() : ArrayList<User>
+setID(id : Integer) : void
+setAddress(address : String) : void
+setName(name : String) : void
+setLeaderboard(list : ArrayList<User>) : void
Event
-ID : Integer
-name : String
-groups : Group[]
-theme : ENUM
-date : Date
-quiz : Quiz
-leaderboard : ArrayList<User>
+getID() : Integer
+getName() : String
+getGroups() : Group[]
+getTheme() : ENUM
+getDate() : Date
+getQuiz() : Quiz
+getLeaderboard() : ArrayList<User>
+setID(id : Integer) : void
+setName(name : String) : void
+setGroups(groups : Group[]) : void
+setTheme(theme : ENUM) : void
+setDate(date : Date) : void
+setQuiz(quiz : Quiz) : void
+setLeaderboard(list : ArrayList<User>) : void
Question
-ID : Integer
-questionText : String
-answer : String
-options : String[]
-theme : ENUM
-countDown : Integer
+getID() : Integer
+getQuestionText() : String
+getAnswer() : String
+getOptions() : String[]
+getTheme() : ENUM
+getCountDown() : Integer
+setID(id : Integer) : void
+setQuestionText(text : String) : void
+setAnswer(text : String) : void
+setOptions(options : String[]) : void
+setTheme(theme : ENUM) : void
+setCountDown(count : Integer) : void
Quiz
-ID : Integer
-theme : ENUM
-countDown : Integer
+getID() : Integer
+getTheme() : ENUM
+getCountDown() : Integer
+setID(id : Integer) : void
+setTheme(theme : ENUM) : void
+setCountDown(count : Integer) : void
uses
checked by
checked by
manages
manages
hosts
hosts
1
*

Figure 1: Class Diagram for the server-side of GrouPub system

3. Proposed Software Architecture

3.1. Overview

In this section GrouPub system is going to be explained in more detail. As a start, subsystems will be covered to see which and what kind of subsystems the GrouPub system is using in order to be efficiently functional. Afterwards, hardware and software mapping of GrouPub is going to be visualized and explained. It is followed by the explanation of persistent data management of the system. In that part, the relation between the persistent data will be visualized and explained. Finally, access control and security, global software control and boundry conditions for GrouPub will be explained in detail.

3.2. Subsystem Decomposition

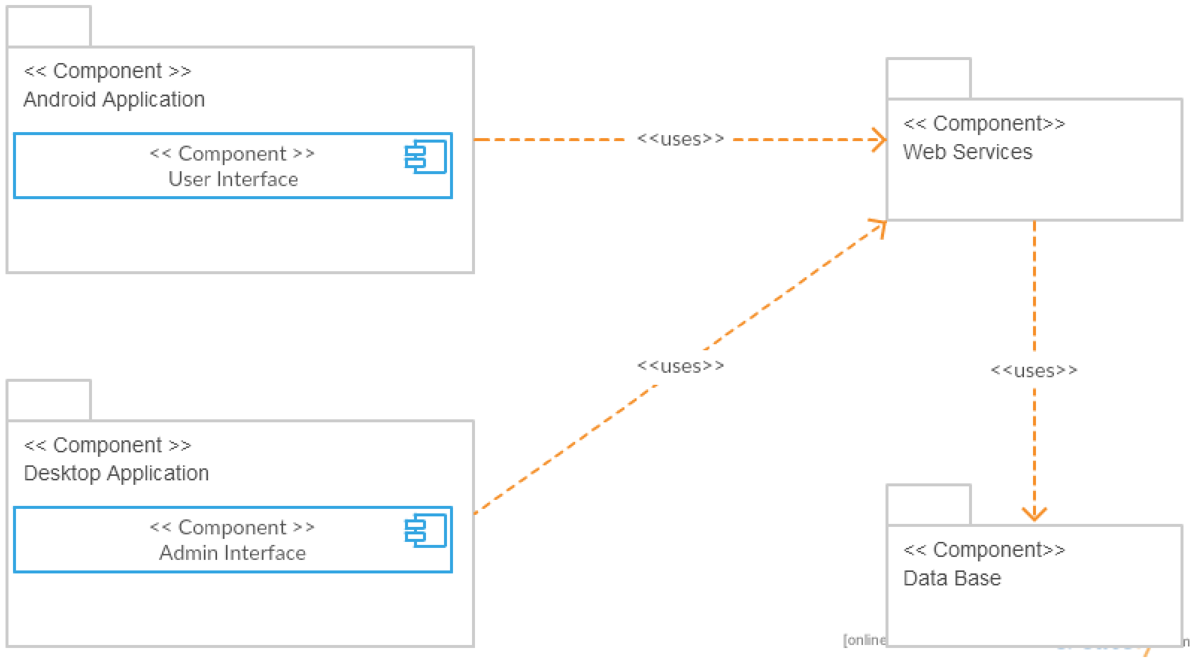

Figure 2: Subsystem Decomposition of GrouPub

Android Application

Android Application has two sub components, user and Admin interface. Components are implemented with JSON because of easy connection with WebService. Application will be implemented with phonegap.

Database Manager

Database Manager Database Manager is the controller of information that is collected in database. Database stores user information, questions for the quiz and cafe’s addresses and information.

Web Service

WebService provides the connection between application and server, database. When a user creates an account or when there is an event registered, will be done with the help of the web service.

3.3. Hardware/Software Mapping

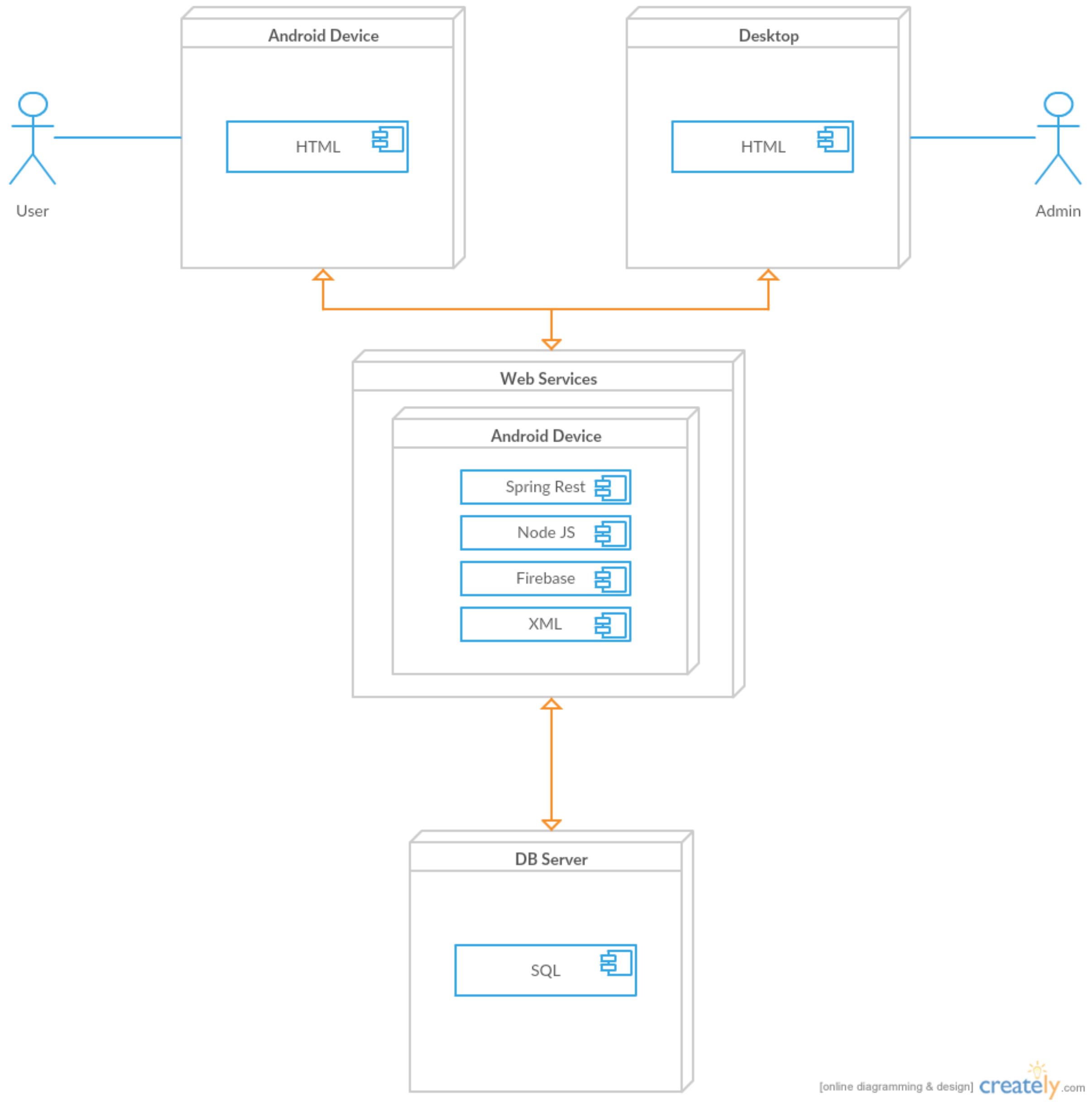

Figure 3: Hardware/Software Mapping of GrouPub

The user will connect to the event by the android device (smart phone). The admin can connect to the events by the desktop computer. Both user and admin use the web service with interacting the android device to add/delete information in database. In web service we use Spring Rest, Node JS, Firebase and XML to reach our database which is using SQL.

3.4. Persistent Data Management

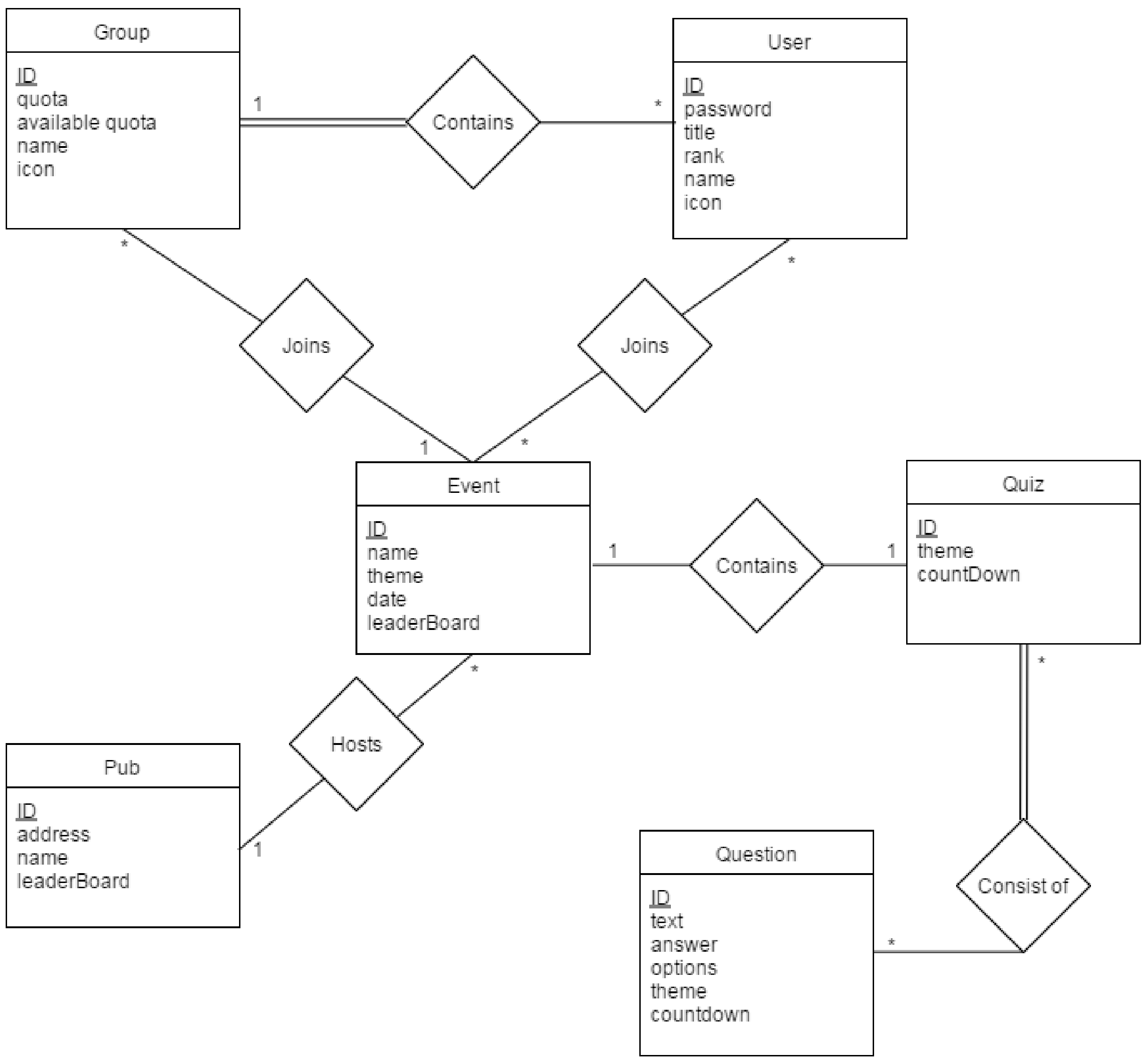

Figure 4: ER Diagram of our persistent data

All the data related to the questions such as the id of the question and its content stored in the database. These question's content will not change very often since there shouldn't be any reason to change the question's content if the question is valid and correct. All user accounts and its information such as user name, points etc. will be stored in the database and can be changed or deleted if needed. All pubs and their data such as their names, addresses and id's will be stored in the database and can be changed if a need occurs. Groups will be created in every event and will be deleted when the event is over which means groups will be constantly added and deleted with each event.

3.5. Access Control & Security

GrouPub has two protection and authentication controls within application. First one is login operation for registered users. When user sends a request, system checks the corresponding data in database system. System allows or denies the user to enter the system according to the information stored in database. If system allows user to attain the related page, otherwise system stays in the login page and asks login information again. Therefore, user accounts are protected by users' login information.

Because GrouPub is location-based chat application, we need to control users' location whether it is valid to attend the quiz or not. Users give the QR code information to the application provided by café/pub. System checks the QR code and if it is valid, gives the user access allowance.

3.6. Global Software Control

There are two different global software control services in GrouPub as web-based phone application and web-based desktop application. These two software systems have similar properties except their usage areas. Web-based phone application uses PhoneGap and used by application users. If the user has suitable software for his/her phone, application runs as mobile application. Web-based desktop application uses web-browser and used by admin to control system backend.

There will be server-client system for web platform. Users send requests to the database and system replies related requests. In addition to this, GrouPub application runs as event-driven. According to the requests, server replies it with corresponding services or continues listening the requests.

3.7. Boundary Conditions

3.7.1. Initialization

GrouPub has two types of users; the regular user of GrouPub and admin.

Users can access GrouPub from their smartphones. In order to login, users need to enter their username and password. If the combination matches an entry in the database, that user will be logged in and the home page of GrouPub will be displayed. If the combination does not match an entry, an error message will be displayed and that user will not be logged in.

Admins will use their PC to access GrouPub's administration page. To login, admins will enter their username and password. If it's a match, they will be logged in and the administration page will be displayed. Otherwise an error message will be displayed.

3.7.2. Termination

After their first successful login, users won't be prompted to log off. They will be kept logged in, in order to prevent the hassle of logging in every time they launch GrouPub. However, if users choose to log off, they can use the log off button to log off anytime they want.

Admin accounts won't be kept logged in once they login. They need to hit the log off button to exit the administration page.

3.7.3. Failure

In the administration page, admins need to save all the changes they made before logging off. If they fail to do so, those changes will not take place and the database will remain unchanged. The failure will occur if there is a connection error during an operation which involves editing the database.

4. Subsystem Services

4.1. Database Manager

Our system will be writing to and reading from its database frequently. Those transactions must be handled efficiently and securely for the sake of the system’s stability. For this purpose we are going to use a transaction manager. It is called Spring Transaction Manager a service that Spring Framework provides. This service allows us to create secure and stable connections to our database.

4.2. Event Manager

Events that need to send requests to our application server has to be handled by a manager. On the backend we will handle those requests using RESTful Web Services. It takes JSON and handle the request accordingly.

4.3. Login Manager

Every user is going to have its own credentials which are ‘username’ and ‘password’. When a user launches GrouPub, those credentials are required in order to login. This authentication process will handled by the login manager. For this purpose we are going to use Spring Security Session Management which allows us to create sessions per user and close those sessions when logout occur.

5. References

[1] Material Design for Android [Online]. Available:
http://developer.android.com/design/material/index.html

[2] Firebase Homepage [Online]. Available: https://www.firebase.com/

[3] HTML - Hypertext Markup Language [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[4] JSON – Homepage [Online]. Available: http://www.json.org/

[5] Node.js – Homepage [Online]. Available: https://nodejs.org/

[6] Phonegap – Homepage [Online]. Available: http://phonegap.com/

[7] Spring – Homepage [Online]. Available: https://spring.io/

[8] XML – W3C [Online]. Available: http://www.w3.org/XML/