King Fahd University of Petroleum & Minerals Department of Information and Computer Science SWE316: Project

Software design document for
Gatherity

11/8/2021

Phase 3

Team 2

201735070 - Abdullah Hekeem

201828120 – Osama Al-Fawaz (phase 3 leader)

201753590 - Abdulrahman Al-Qarawai

201641340 – Ziyad Al-Wagdani

201838060 – Abdulrahman Gharsa

Table of Contents

1. Introduction

Gatheriety is a mobile app that is designed to make social interactions more fun and easier to reach, especially when there are many who are looking for others who share similar interests with them, and who are looking for individuals to do activities with. The SDD is a document that shows the design of the system, Architecture of the system, diagrams of the system (class and package diagrams), assumptions, pseudo code, data design, and finally a requirements matrix of the system.

1.1 Purpose

Our goal with our project "Gatheriety" is to give individuals the ability to easily participate in a wide variety of social activities such as sports activities, board games, video games, topic discussions, and much more, with people who share similar interests. And to allow those who are looking for participants to their own activities to share and advertise their posts for others to find and join. The software is aimed to be built for smartphones. The SDD will help all stakeholders in identifying how the system will be implemented to satisfy the requirements.

1.2 Document Intended Audience

Software developers, to use this document during development phase to implement the design and structure each of the system's component.

Stakeholders, to be able to know how the system will be implemented to satisfy the requirements.

1.3 References

<Doc. 1.> SWE417 SRS Document

<Doc. 2.> Project Description
<Doc. 3.> SDD Document

1.4 Definitions, Acronyms and Abbreviations

Term / Abbreviation	***Explanation***
SRS	*Software Requirement Specifications*
SDD	*Software Design Document*
API	*Application Programming Interface*
UI	*User Interface*
DBMS	*Database Management System*
DSS	*Data Storage System*
SDK	*Software Development Kit*
AI	*Artificial Intelligence*
ML	*Machine Learning*

1.5 Overview

This document, which is the SDD document, will describe and highlight the implementation design an architecture of the main functionalities of our system, it is composed of 9 chapters.

The first chapter, the introduction, which introduces the content of the document including the purpose of the project.

The second chapter, which is the design considerations section, This section describes many of the issues which need to be addressed or resolved before developing a complete design solution. It Describes any assumptions or dependencies regarding the software and its use, then Describes any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact)..

The third chapter, which is the System Architecture section, is written primarily for the developers, it provides and discusses possible architecture styles, it also specifies and chooses the best architecture style based on an evaluation of each proposed architecture style.

The fourth chapter, which is the initial class diagram, in which we will draw a class diagram for the system while applying any necessary abstraction, encapsulation, inheritance, and polymorphism principles. In this section attributes and methods will be shown.

The fifth chapter, which is the final class diagram, will contain the refined version of our class diagram and will discus our thought process on how we came to choose specific design principles.

The sixth chapter, which is the package diagram. In this chapter we will utilize the class diagram to develop a package diagram and we will explain our thought process behind the coupling and cohesion principles applied.

The seventh chapter, which is the pseudo code, will contain pseudo code for non-trivial methods.

The eight chapter, which is the data design, will contain the database description which will identify potential database tables for the system, and the data dictionary which is an alphabetic list of names used by the system (entities, types, services, relations, attributes).

Then the ninth and final chapter is the requirements matrix (tractability matrix) which will provide a cross reference that traces packages and classes to the requirements outlined in the SRS document.

2.Design Consideration

2.1. Assumptions and Dependencies:

Assumptions:

- No male can join female activities and vice versa.
- System users are at least 18 years or older.
- Dart will be used as a programming language.
- Flutter will be used as a UI SDK.
- The user will have notifications on
- Microsoft word is to be used for tracking and managing the progress of documentation.
- The system shall support 1000 users at the same time.
- The user will be connected to the network
- All users will have an active account
- The application will access users locations

Dependencies:

- Google maps API.
- Firebase.

2.2. General Constraints:

- The product will be delivered on iterative bases. Each iteration is 5 weeks.
- The product is to be developed following Agile Scrum method.
- the scrum master will choose the feature from the product backlog and put it in the sprint backlog
- The product is to be developed with the budget of $500,000.
- The product is to be modeled using UML as a modeling language.
- The application requires no more than 1GB RAM.
- The application requires no more than 500MB space on storage.
- The application requires an internet connection.
- The application requires a GPS.
- The application will be designed using a hybrid publisher-subscriber and client-server architecture.
- The application will be developed using Flutter for front-end for both IOS and android.
- The application will be developed using Firebase for back-end for both IOS and android
- UI/UX will be developed using inVision
- The system shall be available 95% of the time per month.
- The system shall support 1000 users at the same time.
- The system must response in 1 second in the home page and render text and images for less than 4 seconds.
- The user validation process shall not take more than 5 seconds.

3 System Architecture

3.1 System Architecture

Client Server architecture

Components:

1. Client
2. Server
3. Databases

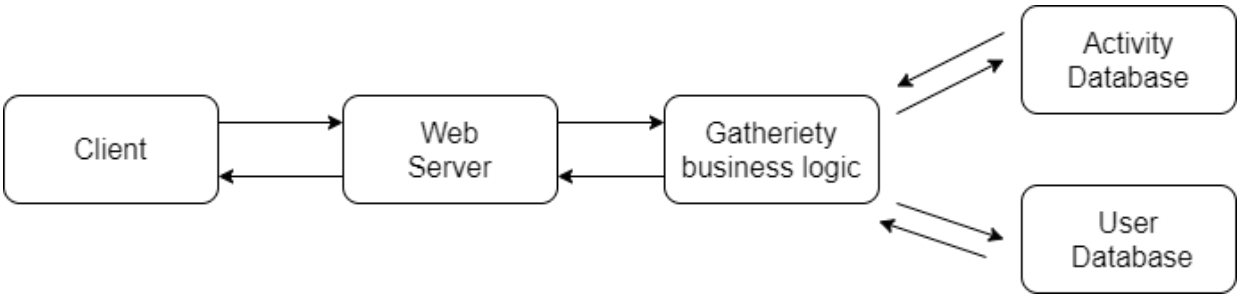
Connecters:

1. Remote procedure calls
2. Network protocols

Quality attributes:

1. Efficiency: Centralized data and the high processing ability increase efficiency.
2. Portability: the ability to use the software on different environments.
3. Security: The ability to secure the server and other layers.

Topology:



Object Oriented

Component:

1. Packages
2. Database
3. Google API

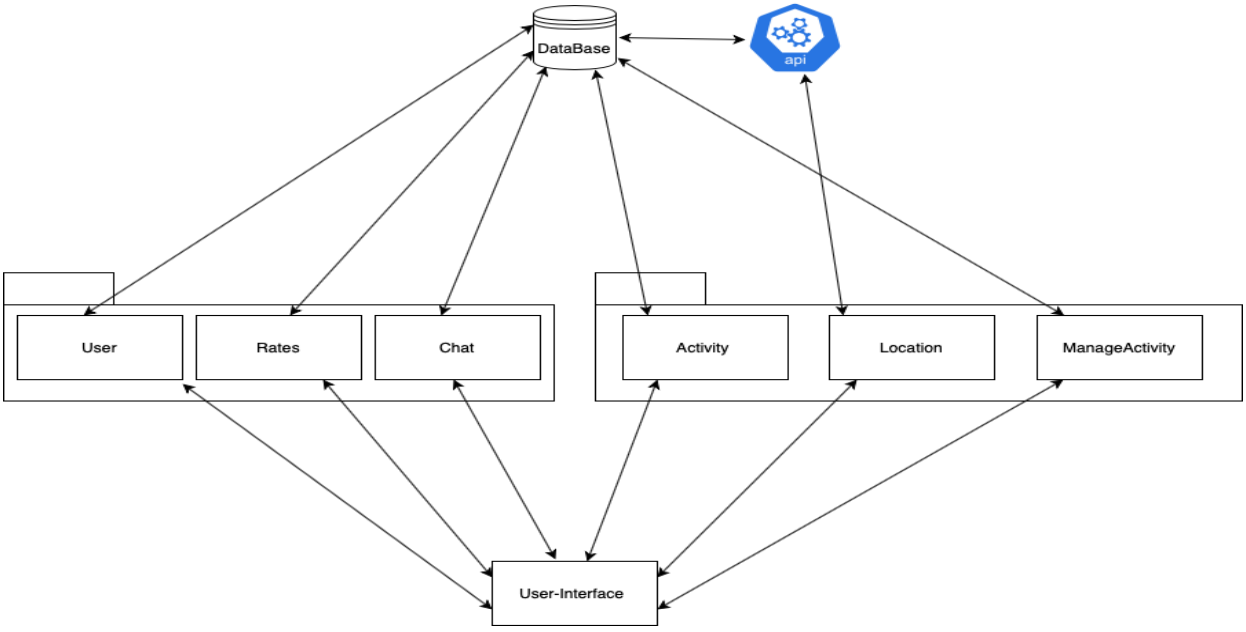
Connectors:

1. Procedure calls
2. Database query

Quality attributes:

1. Maintainability: because all the related classes are in the same package and the relation between them clearly defined.
2. Reusability: the nature of object-oriented approach and applying the Common Reuse Principle increase reusability the system.
3. Scalability: object-oriented approach utilizes the use of the inheritance which increase the scalability of the system

Topology:



Blackboard

Component:

1. User component
2. Activity component
3. Google API

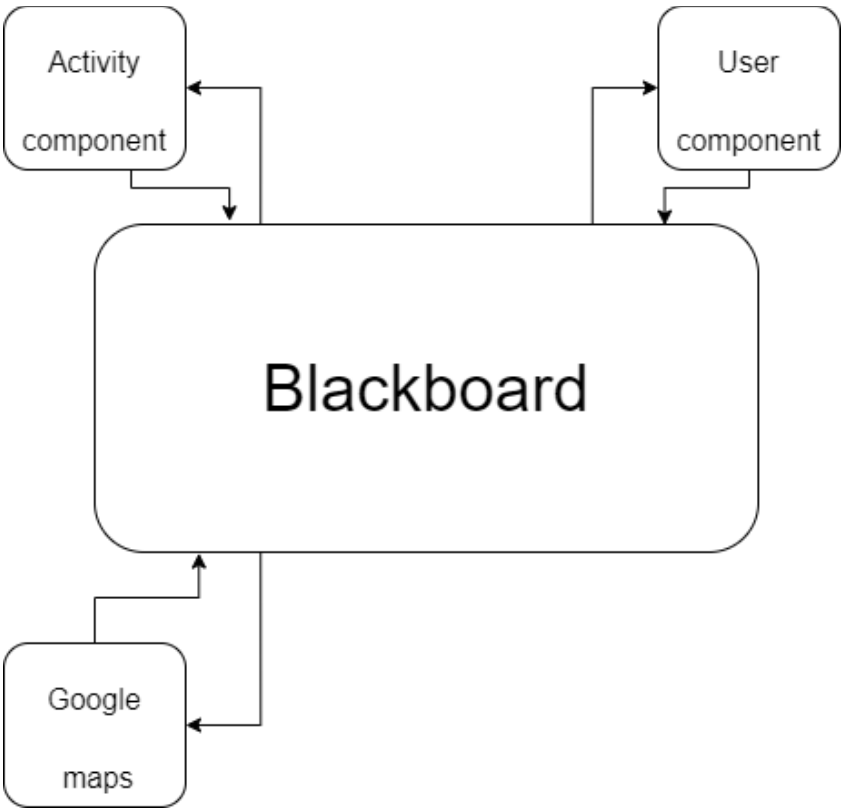
Connectors:

1. Database query
2. Procedure calls

Quality attributes:

1. Performance: all the information is available at the blackboard where all the components have access to it.
2. Scalability: the ability to add components directly to the blackboard.
3. Maintainability: the ability to shut down certain components without breaking the system.

Topology:



Hybrid Client-Server/P2P

Component:

1. Peer (User-Interface)
2. Server
3. Google API
4. Database

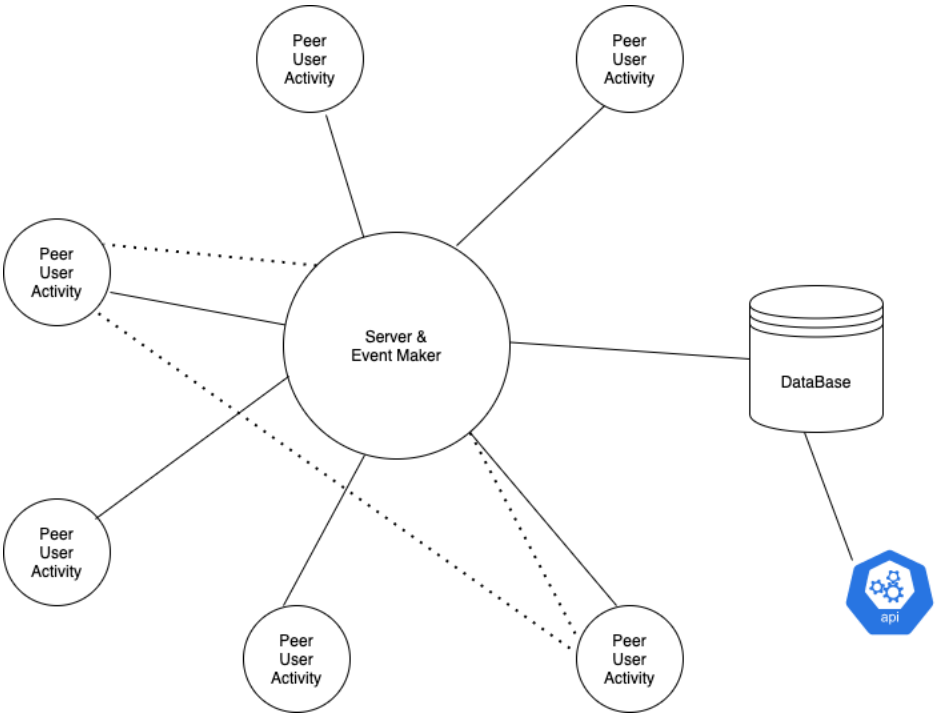
Connectors:

1. Database query
2. Procedure calls

Quality attributes:

1. Performance: all the information is available through the server where all the components have access to it.
2. Scalability: the ability to add components directly to the Server.

Topology:



Hybrid Client Server/Publish Subscribe

Components:

1. Client / Subscriber

2. Server / Publisher

3. Database

4. Google API

Connectors:

1. Remote procedure call

2. Queries

3. Network protocol

Quality attributes:

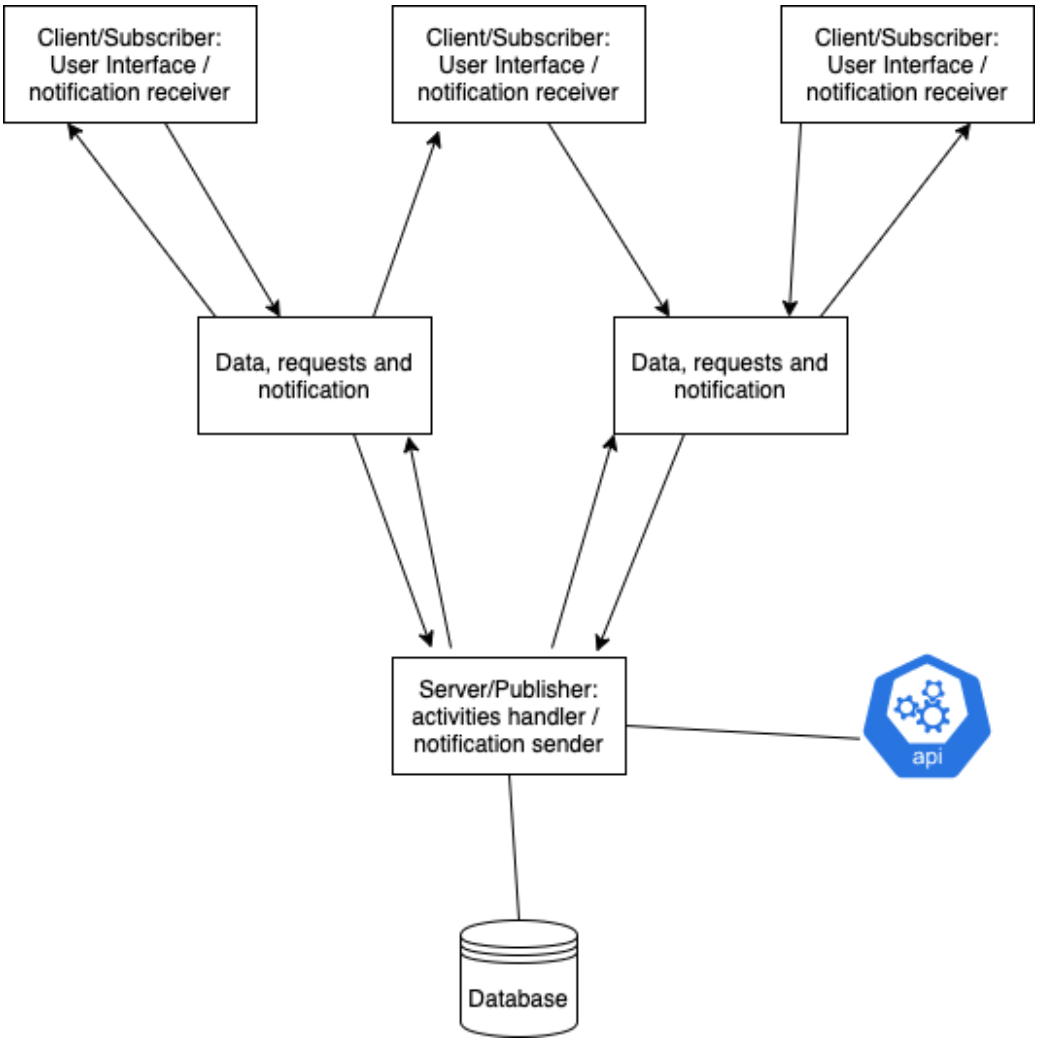
1. Scalability: the ability to add components directly to the Server.

2. Efficiency: Centralized data and high processing ability of the server increase efficiency

3. Portability: the ability to use the software in different environments

4. Security: the ability to secure the server and other layers

Topology:



3.2 Design Rationale

We evaluated the quality attributes based on which are most fitting for our requirements specifically.

Architecture	Client server	Blackboard	Object oriented	Hybrid p2p/client server	Hybrid client server/publish subscribe
Scalability	High	Moderate	High	Moderate	High
Performance	High	Low	Moderate	High	High
Portability	High	High	Low	Moderate	High
Security	High	High	High	Low	High

Upon discussing and evaluating each architecture, we concluded that Client server architecture was the most fitting for our requirements based on the table above however, upon analyzing client server architecture, we found out that merging Client server with public subscribe architecture would give us an advantage of sending notifications to users which make it more fitting for our requirements.

In our chosen style, Hybrid client server/publish subscribe:

The server will receive procedure calls from the user interface containing the activities and will communicate with the database to store the activities alongside the locations gathered from Google API and also will send notifications to end users (clients), based on their shared interest.

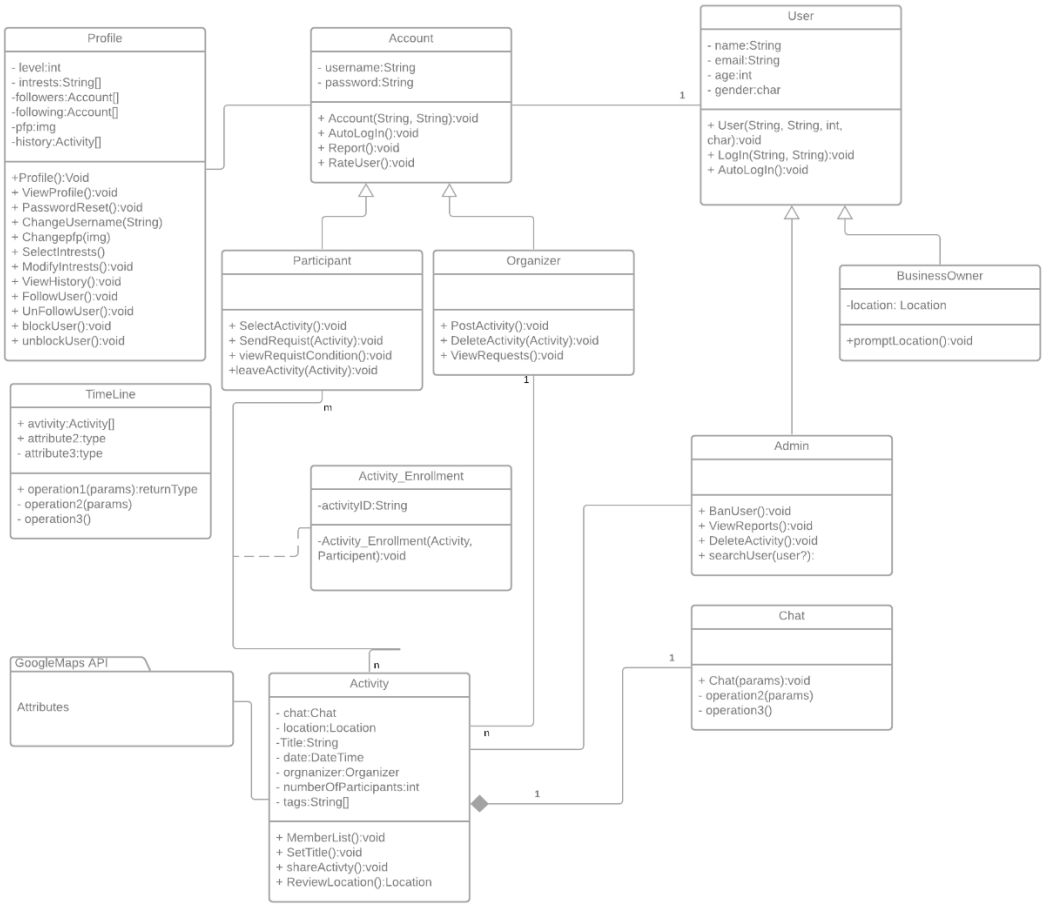
The clients will send procedure calls to the server to initiate activities and check for ongoing events/activities, it also will receive notifications whenever a new activity with shared interest is initiated.

Issues with our chosen style, is reliability, for example, if the server went down, the whole process would shut down, therefore a high runtime is required and shall be focused on.

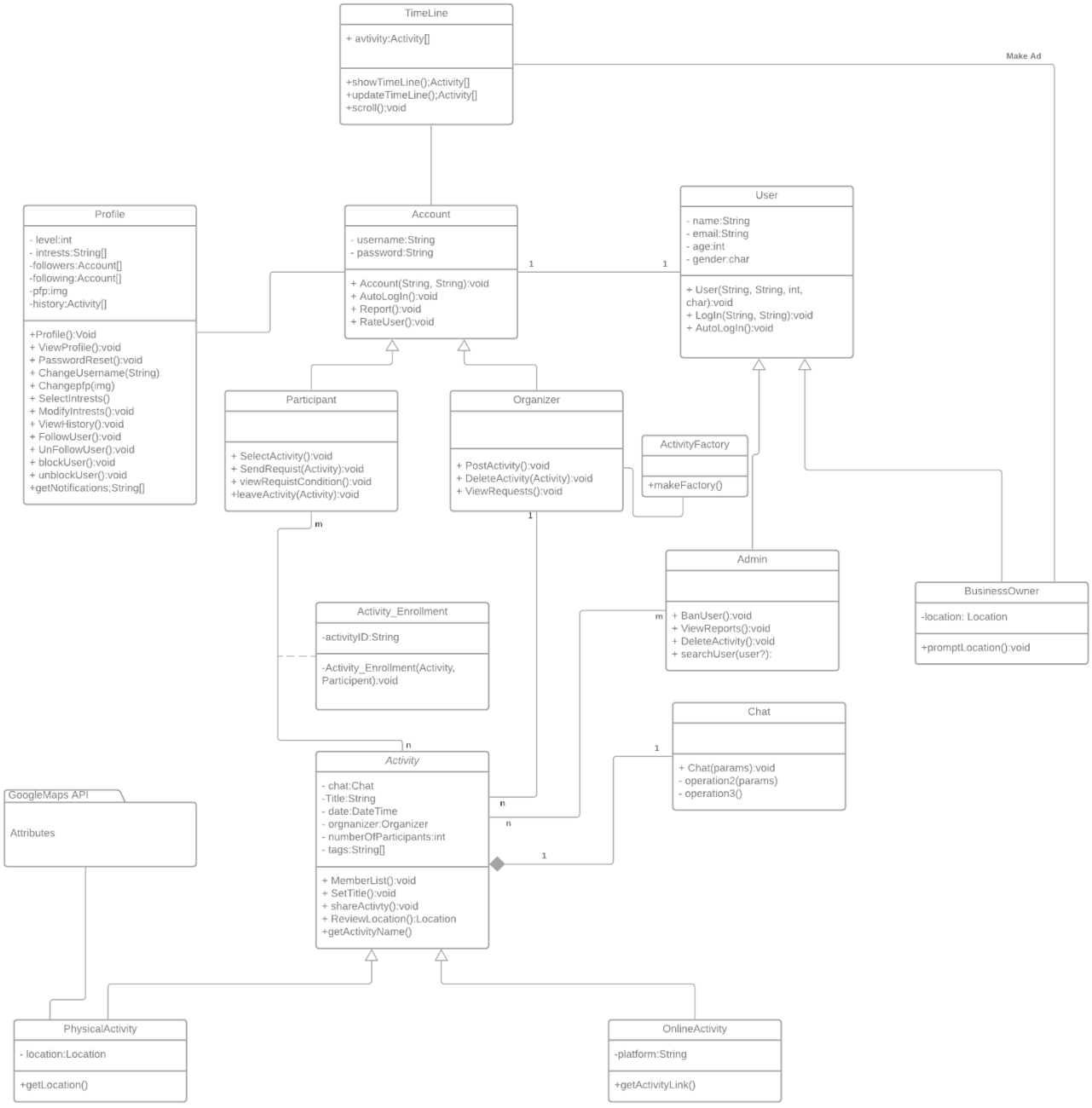
4 Initial Class Diagram

Link for the photos in case it is not clear in the document:

https://kfupmedusa-my.sharepoint.com/:f:/g/personal/s201838060_kfupm_edu_sa/ElhH6gIE2otNlfWgfqepe9YBcBxrjE0HUA5dpCSmtSr0XA?e=JMJwSG

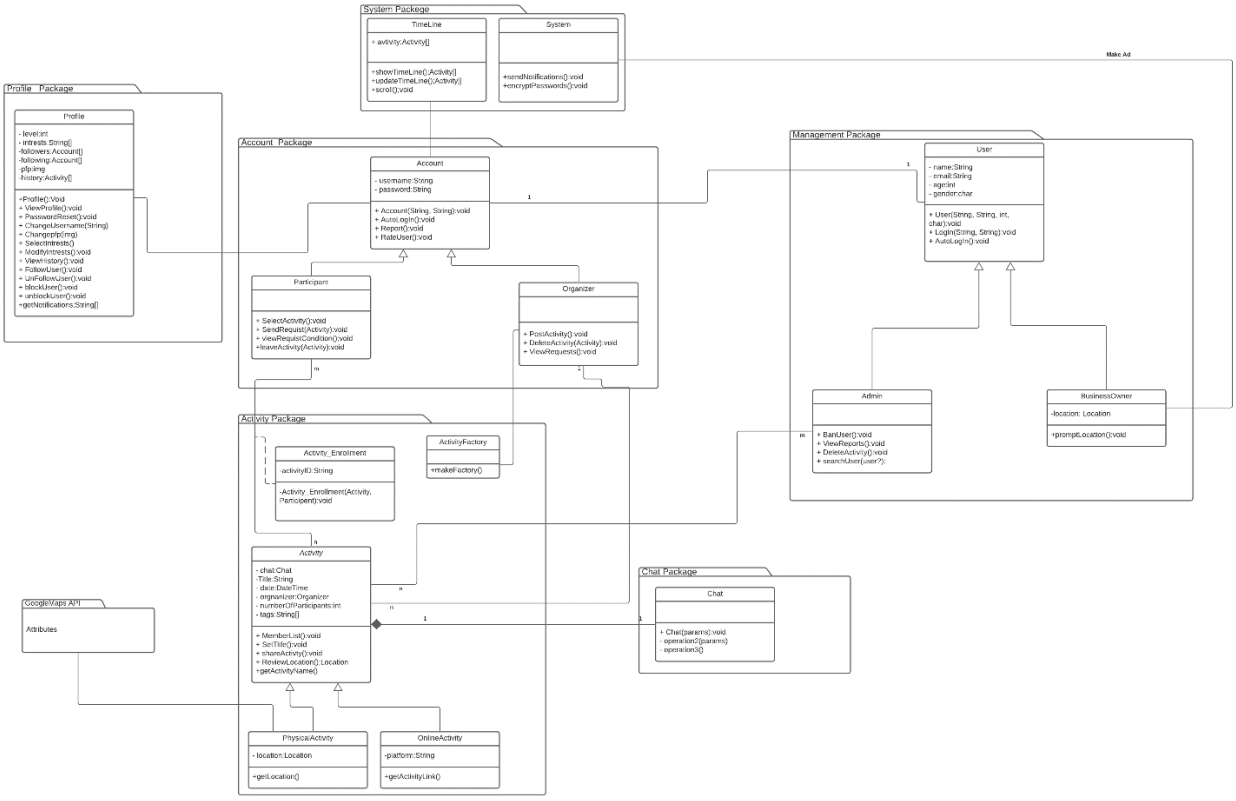


5 Final Class Diagram



We used a class called “ActivityFactory” the purpose of using this class to increase the maintainability and the scalability by creating many types of objects in one class. So, this will help us to add new activities and creating them in the factory without creating them in the main class, for the main class we only need to get the type of the activity, and the factory will create the object.

6 Package Diagram



The common closure principle is used in creating the package diagram, and each package contain classes that have the same functionality. Which will help in maintain the system in the future.

7 Pseudo Code

Sign In:

User fills the username textbox and password textbox

logIN(usernameTextbox.value, passwordTextbox.value)

AutoLogin{

if(account.exists)

userlogged in

else

redirects user to sign up page

user fills the text boxes

User(name, email, age, gender, password)

Database.add(user)

A new account is created

The User is signed in

}

Report:

The user goes to profile page and clicks on report

report()

user clicks on violations attempted by the profile account and clicks submit

database.add(report)

(to admins)

Select interests:

selectInterests()

user clicks on desired interests

return selected interests

database.add(return value)

Manage details:

Change username

User clicks on change username

The use fills the text box

changeUserName(textbox.value)

Database.replace(value)

username updated.

Change profile picture

The user click on change profile picture

changePFP()

An upload window pops up

The user clicks on the desired picture from his local machine

Database.replace(img)

The profile picture is updated.

Modify Interests

The user clicks on modify interests

modifyInterests()

The user clicks on the Interests to add and remove

Database.add(interests)

Follow user:

The user clicks on follow button

FollowUser()

Database.add(followedUser)

The user added to the followed users

Unfollow user:

The user clicks on unfollow button

unFollowUser()

Database.delete(followedUser)

The user removed from the followed users

View profile:

View 3 endorsement, view level

Block user:

View activity history:

The user clicks on view activity history

viewHistory()

the user is redirected to the view history page.

show timeline:

the user clicks on show timeline

updateTimeLine() (This method updates and filter timeline)

showTimeLine()

the user is redirected to the timeline page

timeline is displayed.

Notify user:

If(intresets.relevant(user)

notifyUser()

Send request:

The user click on join activity

sendRequest()

Database.add(request)

Select activity:

if(activity.ongoing)

selectActivity()

else

print(“activity is finished”)

Share Activity:

The user clicks on share activity

GenerateActivityLink()

If(link.exists)

copyLinkToUserClipBoard()

else

createActivityLink()

copyLinkToUserClipBoard()

Chat:

the user clicks on open chat

launchChat()

Notify for arrival:

If(new participant arrives)

notifyParticipants()

notifyOrganizers()

Rate user:

The user goes to a profile page of either participant or organizer

rateUser()

the user enters the rating

database.add(rating)

Post activity:

create.button

Method called postActivity {

String type= “” /initialize type

Prompt user to enter type of activity

type= input

call ActivityFactory.makeFactory(type){

case 1:

type.comateTo(“type1”)

case 2:

type.comateTo(“type2”)

…

Default :

Print(“no valid activity”)

Exit cases

If the type is not valid:

Print("activity successfully created”)

}

}

View request:

Organizer clicks view requests button

Request list displayed from database

If(organizer clicks accept button)

Database.add(participant)

notifyParticipant(“accepted”)

if(organizer clicks refuse button)

notifyParticipant(“denied”)

Delete activity:

Organizer/admin clicks delete button

Database.remove(Activity)

notifyParticipants(“activity deleted”)

View reports:

Admin clicks view reports button

If(admin clicks activities reports button)

Activities reported are pulled and displayed

If(admin clicks users reports button)

Users reported are pulled and displayed

Ban user:

Admin clicks on the ban button in the user profile

Print(“choose period”)

If (admin enters period)

period = input

banUser(period)

notifyUser(“banned from participating for:”+ period)

else

period= very big value

banUser(period)

notifyUser(“banned from participating forever”)

Search user:

Admin clicks search users button

print(“enter user name or email”)

users are pulled from the database using compareTo(input) and displayed

admin clicks on desired user

user.Account.Profile.viewProfile()

Unban user:

Admin clicks on the unban button in the user profile

period = 0

banUser(period)

notifyUser(“you are no longer banned”)

Promote location:

Business owner clicks the promote button

Print(“what is the promotion period”)

Period = input

Price= period*rate

Print(price)

Redirect the business owner to the payment service

For the ad period TimeLine.updateTimeLine()

Leave activity:

User clicks on leave activity button on the activity window

leaveActivity(activity)

print(“successfully left activity”)

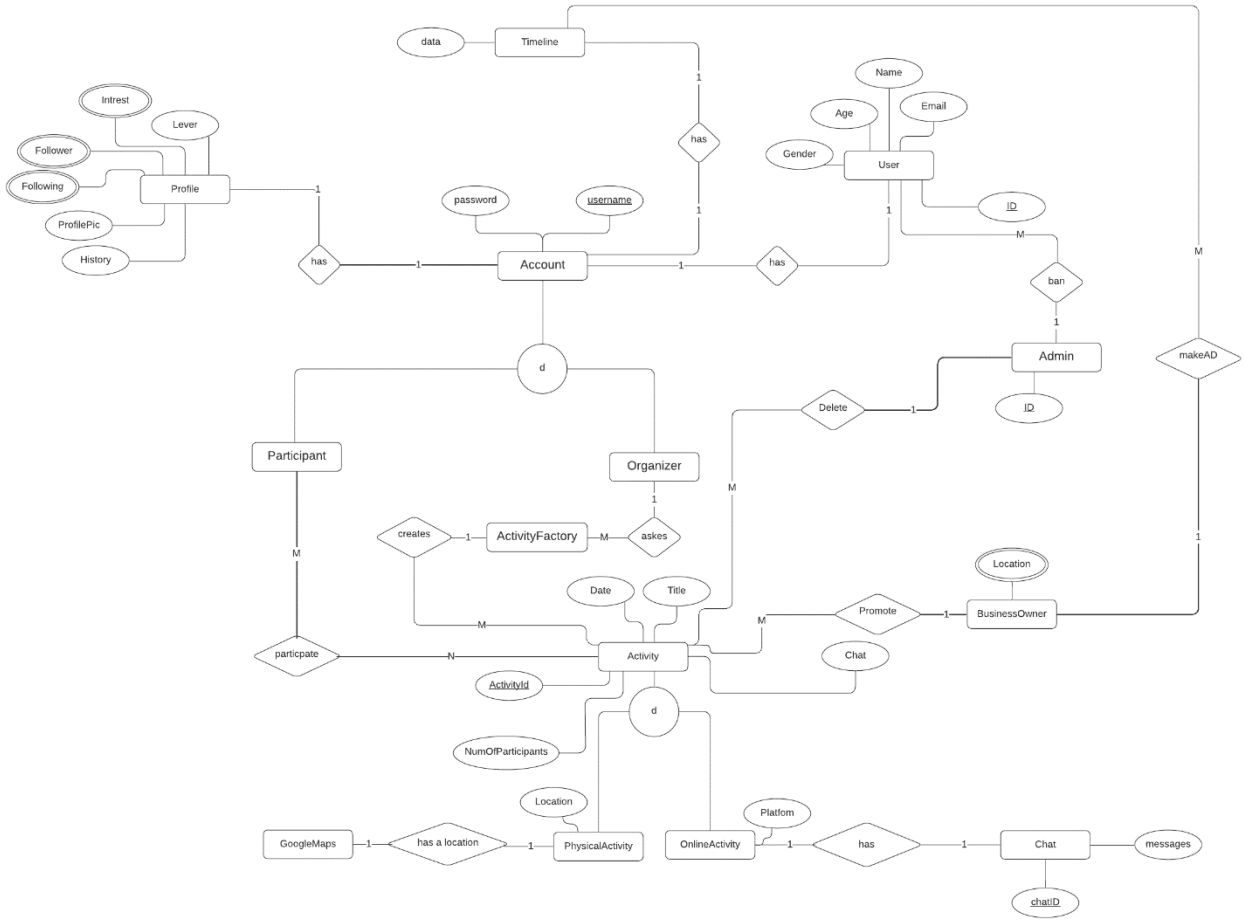
(any user can leave the activity←parameter)

8 Data Design

8.1. Database Description:

8.2. Data Dictionary:

For part 8.1 and 8.2 we made an EER diagram to show all entities, attributes, relations, types, and services. For the tables we assumed each entity with its attributes and many-to-many relations will be tables (relational schema).



9 Requirements Matrix

[tractability matrix]– provide a cross reference that traces packages and classes to the requirements outlined in the SRS document.

Req/Feature #	UC #	UC Description	Package(s)	Class(es)
R1, R2.2	UC-01	The email entered by the user will be checked if he already has an existing account or is making a new account	Account Package	Account
R2, R3, R3.1, R3.2, R3.3	UC-02 UC-03	The user can only access this page if he has an existing account / the user creates a new account	Management Package	User
R10.3	UC-32	The admin will be able to ban user	Management Package	Admin
R2.3, R4, R4.1, R4.2, R5, R5.1, R5.2, R5.3, R5.4, R5.5, R5.6, R5.7, R6, R7, R8, R9, R9.1, R9.2, R10, R10.1, R10.2, R11	UC-04 UC-05 UC-06 UC-07 UC-08 UC-09 UC-10 UC-11	Select interests / the user can manage his profile / the user can follow other users and unfollow / the user can see profiles / the user can block accounts and unblock / the user can see his history	Profile Package	Profile
			Account Package	Participant
			Account Package	Organizer
R13.1, 13.2	UC-14	The system display the activities based on interests ..etc.	System Package	TimeLine
R2.1, R2.4, R3.3, R3.4, R3.5, R3.6, R12	UC-12	The user can see the top 3 / the system sends notification	System Package	System
			Management Package	BusinessOwner

R23.1	UC-14	The system displays a sorted list of activities which is based on its relevance to the user	Activity Package	*Activity*
R23	UC-24	a notification alert needs to be able send all participants when any of participants arrives to the selected location	Activity Package	*Activity*
			Activity Package	PhysicalActivity
			Activity Package	OnlineActivity
R22, R22.1, R22.2, R22.3	UC-23	All activity participants would be able to contact each other using chat system	Chat Package	Chat
			Activity Package	ActivityFactory
			Activity Package	Activity_Enrollment
R13, R13.1, R13.2	UC-14	The system displays a sorted list of activities which is based on its relevance to the user	Activity Package	Activity class
R14	UC-15	The system notifies the users with recommended relevant activity posts.	Profile Package	Profile class
R15,R15.1	UC-16	When an activity post is selected, the	Account Package	Participant class

		system shall display the description, activity tags, number of participants and approximate location.		
R16, R16.1, 16.2, 16.3	UC-17	The activity participant sends a request to join an activity with a message to its organizer.	Account Package	Participant class
R17, R17.1, R17.2, R17.3, R17.3, R17.4, R17.5	UC-18	The user is able to search for an activity either by writing some words and the system displays the best results or by specifying the type of search used.	System Package	TimeLine class
R18	UC-19	The user can display similar activity of a selected activity.	Activity Package	Activity class
R19,R19.1	UC-20	The user can send to others though the main social media a link that will take them to shared activity.	Activity Package	Activity class
R20, R20.1, R20.2, R20.3, R20.4, R20.5	UC-21	The user can report a violation and specify its type and details.	Account Package	Account class
R21	UC-22	a user can endorse another user in a particular subject or a skill.	Account Package	Account class
R24	UC-26	After an activity ends the	Account Package	Account class

		participants can rate each by answering few simple question		
R25	UC-27	The user would be able to review any meeting location of an activity he has been part of	Activity Package	PhysicalActivity class
R26, R26.1, R26.2, R26.3, R26.4, R26.5, 26.6	UC-28	*The user can create and post a new activity for participants to join with specifying location, date, time, description, and related activity tag.*	Account package	Organizer class
R27	UC-29	The user receives join requests for an activity he creates, and either accepts the request or denies it.	Account package	Organizer class
R28	UC-30	*The activity organizer can delete an activity post he created*	Account Package	Organizer class
R29	UCS-31	The admin can view all violation reports and solve the issue	Management Package	Admin class
R30, R30.3, R30.4	UC-32	The admin is able to ban any user who did any illegal action in the application	Management Package	Admin class
R30.1	UC-32.1	The admin can view all reports and solve the issue	Management Package	Admin class
R30.2	UC-32.2	The Admin is able to search for a	Management Package	Admin class

		specific user directly		
R32	UC-33	The Admin is able to delete inappropriate activity post manually and directly after his command	Management Package	Admin class
R33	UCS	The location owner (representative) can promote the location of his place to be recommended for activity organizers when creating an activity related to this location	Management Package	BusinessOwner class

Meetings	Attendances	Duration - Date	Purpose
Meeting 1	Abdullah Hakeem, Ziyad Al-Wagdani	30 mins - 11/4/2021	Work distribution
Meeting 2	Abdullah Hakeem, Abdulrahman Gharsa, Ziyad Al-Wagdani, Osama Al Fawaz, Abdulrahman Alqarawi	1 hour - 11/8/2021	Design rationale Reviewing and combining work
Meeting 3	Abdullah Hakeem, Abdulrahman Gharsa, Ziyad Al-Wagdani, Osama Al Fawaz, Abdulrahman Alqarawi	2 hours – 24/11/2021	Creating the initial class diagram and brainstorming for the final class diagram and the start of creating the final class diagram
Meeting 4	Abdulrahman Gharsa, Ziyad Al-Wagdani, Abdulrahman Alqarawi	3 hours	Finishing the final class diagram and finishing the package diagram
Meeting 5	Abdulrahman Gharsa, Ziyad Al-Wagdani, Abdulrahman Alqarawi	2 hours	EER diagram
Meeting 6	Abdullah Hakeem, Osama Al Fawaz	2 hours	Pseudo code

Meeting 7	Abdulrahman Gharsa, Ziyad Al-Wagdani	2 hours	Tractability Matrix

Work distribution: All the work has been distributed equally between team members and have been made through collaborated effort through live chat.