Bilkent University

Department of Computer Engineering

Senior Design Project

SEPS

High-Level Design Report

Taner Durmaz, Samir Ibrahimzade, Mehmet Erkin Sahsuvaroglu, Alperen Koca, Burak Yeni

Supervisor: Shervin Rahimzadeh Arashloo
Jury Members: Dr. Can Alkan and Dr. Cigdem Gunduz-Demir

High Level Design

December 26, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

1.1 Purpose of the system

SEPS (Social event photo sharing) is a mobile application which creates a comfortable environment for the event organizers and attendants to share and access the photos of certain events while securing their privacy. SEPS aims to enable users to share their photos in a social media environment, while taking advantage of cutting-edge security technologies, such as QR protocols and image processing solutions.

SEPS will operate on the Android operating system as an online application. Within the Android OS, users will be able to share photographs from events to the participants of that event. Moreover, the security requirements will also be guaranteed by QR scanned codes and photograph contents, which will be imputed by Android machines and processed at backend. SEPS will also inform users about security issues from their Android OS machines.

1.2 Design goals

- **Availability**

SEPS will be available for the usage of every person possessing Android OS machines.

- **Accessibility**

SEPS will be accessible through the Android store, “Google Play”. The users, who have mobile Android machines, can download the application from that “Google Play”. Upon further developments, users might also use SEPS from their PC.

The users will be able to access the server of SEPS, while the event security will be prioritized. Only participants of an event will be able to see event photographs. QR scan code and security status of an event will be accessible for the participants of that event.

- **Efficiency**

Mobile devices with Android OS should be able to run the offered system SEPS with ease. The photos from events will be held in the database of SEPS, such that the users should be able to access them efficiently. Event crowdedness would have an effect on QR scan and image processing due to the possible amount of content, which issue should be dealt in SEPS.

- **Flexibility**

    The system of SEPS should be open for future modifications, add-ons and updates. An update on the system due to user issues might naturally be needed. Moreover, updates on GUI and color theme would also be beneficial for attracting user applause. SEPS would also be offered for IOS systems, hence the system should not depend on merely one platform during designation.

- **Manageability**

    System will be able for remote administration operations by the developer team of SEPS, under privacy protocols. Azure will be used for the sake of manageability. For the management side of users, they will be able to create accounts and modify its status.

- **Integrity**

    The system of SEPS will be integrated with the camera of the mobile device within the Android OS. Moreover, the server and client devices will be integrated in order to offer a QRcode feature for the user events.

- **Performance**

    The systems planned performance metrics will depend on user device, user operating system, network speed and back-end algorithm speed. Hence, the designation should prioritize using fast algorithms for image processing operations. Compatibility with the mobile hardwares of users is also a variable in terms of performance.

- **Portability**

    The system aims to be used in a portable manner via mobile phones.

- **Reliability**

    The database of SEPS will potentially include private photographs and other contents among users. It is essential to ensure the security of these private datas, so that no actors would have an access to the database contents directly, including the developers. The QR reliability is also essential for the functionality of SEPS.

- **Robustness**

    The application of SEPS will record user states for a planned event, such as “Invited”, “Entered”, “At break”, “Not Invited”. Upon a failure and a need for reboot, the system should be able to stay consistent.

- **Simplicity**

    The GUI of SEPS is planned to be simple such that it consists of a minimal amount of buttons. The usage instructions of QR codes should also be simple and brief.

- **Scalability**

    The database of SEPS will be scalable such that crowded events or huge user numbers would not entail problems. The server side should not lose performance upon scaling states.

1.3 Definitions, acronyms, and abbreviations

API:	Application Programming Interface
HTTP:	Hypertext Transfer Protocol
GDPR:	General Data Privacy Regulation
GUI:	Guided User Interface
ID:	Identification
IOS:	iPhone Operating System
OS:	Operating System
PC:	Personal Computer
QR:	Quick Response
SDK:	Software Development Kit
SEPS:	Social event photo sharing
TBD:	To be declined

1.4 Overview

In the following sections, firstly the current and proposed architecture of software will be elaborated, in which diagrams and planned phases can be seen. Afterwards, the subsystems and components of the SEPS system will be clarified. Finally, the internal affairs of the developers of SEPS during the designation and reporting phases will be given.

2. Proposed software architecture

2.1 Overview

Our system decomposition represents the structures of systems and subsystems of our project with details. First of all, subsystem structure is very important and includes the fundamentals of the project. Subsystem decomposition clearly shows how subsystems interact with each other. If diagrams are looked at, the responsibilities of each subsystem are clearly viewed. They can be information received or sent, processed information and information transportation between subsystems. These predefined subsystems also make the coding phase easier and make the program less error prone.

To ensure our program works properly and is less error prone, spending lots of time on defining and identifying subsystems is important. In this way, the program will be easier to code and less errors will be faced when we deeply code our program.

2.2 Subsystem decomposition

We have used the subsystem decomposition of client-server architecture. Why this architecture is selected is because our program needs the client to get inputs and needs a server for the outputs and connection.

Our program has a Client tier including Controllers and Views components. This tier turns on client devices and will be able to take photos, upload them into the server and other functionalities about input.

Controller component will take the photos, edit them and will do anything needed to be processed. View component will include the general properties about user interface and representation of content.

Our program also has a Server tier consisting of Application tier and Data tier. In the Application tier, there are DatabaseManager component, ImageClassifierEngine component and FaceMaskRecognizerEngine component. Data tier has models component.

DatabaseManager component will handle the information about datas including photos, edits, personal information, accounts. Nearly all of the information needed to be kept on the server is handled by the DatabaseManager component.

ImageClassifierEngine component will handle the processes about images. Scanning images, editing, comparing them and future properties will be handled by ImageClassifierEngine component.

FaceMaskRecognizerEngine component will handle the recognition of face masks on faces, whether a person wearing masks or not, and future properties about masks will be handled by this component.

In the Data tier, Models component will be used for pre built files, pre built images and to store new models to compare them for upcoming ones.

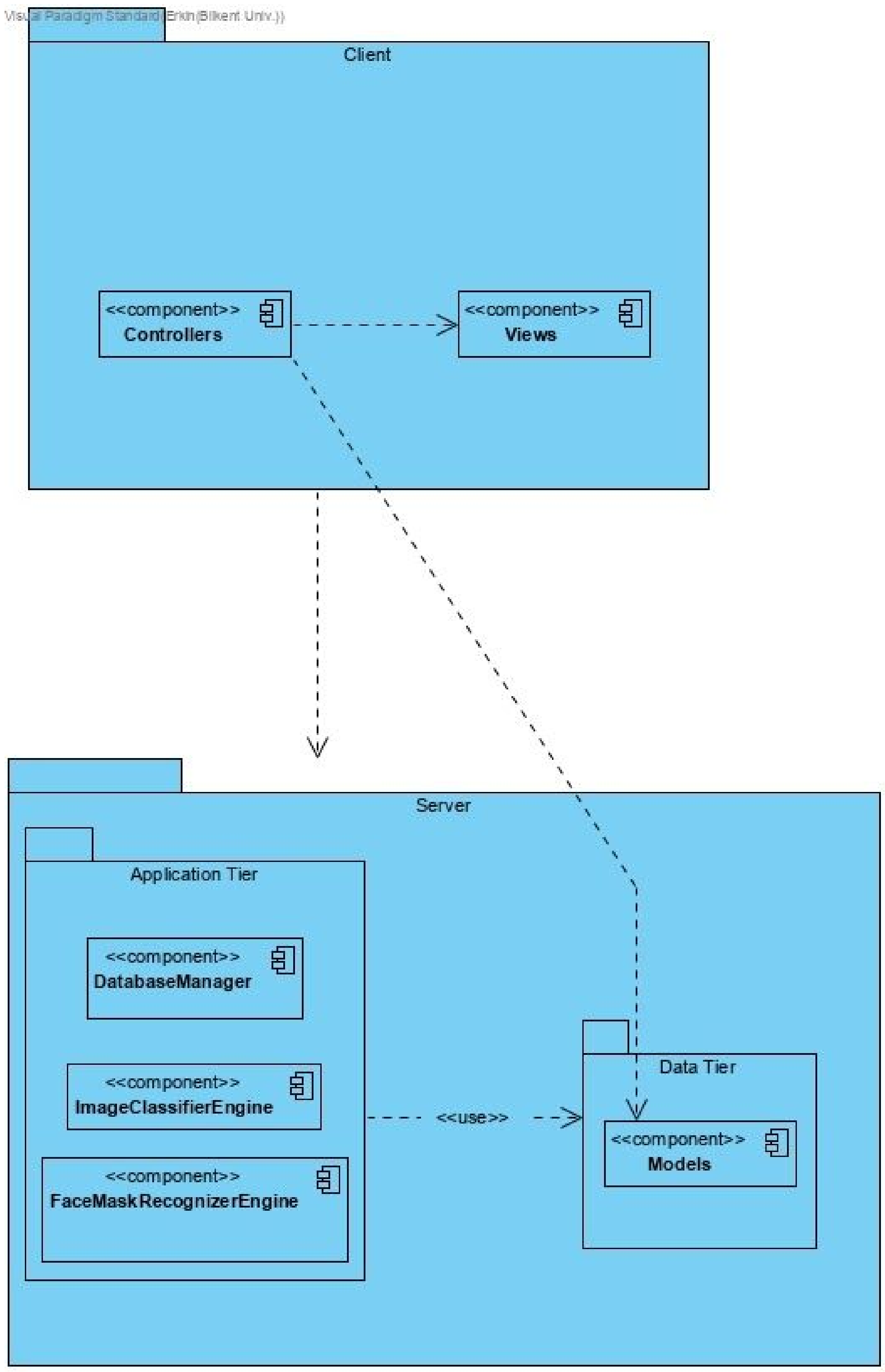

Figure 1: Subsystem decomposition UML component diagram

2.3 Hardware/software mapping

Our program will run on Android smartphones. So, softwares that works within the client side must be compatible for various models. Program will use the resources such as camera, connection purpose hardware, memory, screen etc. Since program use the phone memory, most of the photos taken will be stored in a Azure cloud system.

For our backend, server code and API integration will be implemented in Python language and Android app will be coded with Java and Android SDK.

HTTP requests/responses will be used to communicate with the server. The server side is planned to work on Azure services.

Our database stores essential information, and will have connections with the server. Database also provides and retrieves images for users.

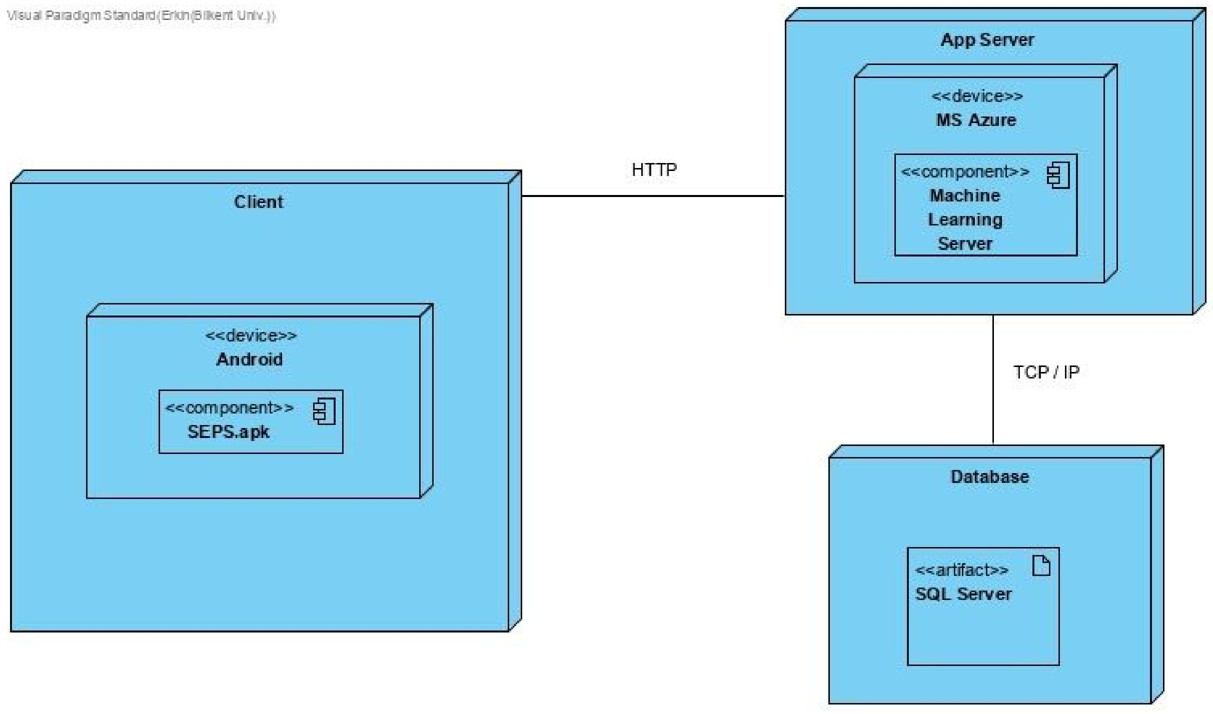

Figure 2: Hardware/Software Mapping deployment diagram

2.4 Persistent data management

Persistent data means a bottleneck in the system in many different ways: most of the functionalities can be concerned with creating or manipulating persistent data. So, access to the data should be fast and reliable. If retrieving data is slow, the whole system will be slower. If data corruption occurs, the whole system would collapse.

Nearly all of the features of our application are made for users' comfort such as fast and reliable image scanning and face detection. So we must take these features into consideration when designing our database. To provide reliable user experience, we will use a relational database to store persistent information.

All the data is saved in the Server Database if the user accepts it. If a user allows us to keep personal private information such as ID, phone numbers, addresses, we will make sure that the data remains safe. This will be done via a backup database.

2.5 Access control and security

Since our system includes multi users, different users must have different access rights over the images and information. Thus, access control is essential to ensure that only the users who have the necessary permissions access the restricted parts of the system.

Access control is significantly important for us because we keep the personal images and these files must be kept safe. In other words, users can edit the images they can access and these images are kept privately for each user.

For deciding who should access what information, we set some rules for different perspectives. Administrations such as developers and maintainers cannot access and modify any of assets, input and output images. However, users can only access and modify their assets, input and output images.

For example, user George does not have any access over another user's data, William etc. and William does not have any access over another user's data, George etc.

Data privacy is a fundamental issue that should be strictly kept safe. Our program is designed to conform to the General Data Privacy Regulation (GDPR).

Additionally, our program will never keep user data in the remote servers without explicitly asking for permission. The photo that the user takes on the phone will be transferred to the server, then the program will delete the photo after the processing is done if the user denied the permission. In case of the failure of the server communication, the server returns the initial state where there is no input image, by this way, it is ensured that no data is kept without permission.

2.6 Global software control

In this section, how the system is controlled on a global scale is described. There are two fundamental subsystems, Client and Server. Program requests information from the Server and receives it from the Client. Requests are processed and response is sent back to the client. Whole procedure is divided into three Phases.

1- Request Phase: This phase begins with the initiation of the requests on the client. A request is initiated when the user takes the photo and uploads it. This user-action triggers a request which sends the photo to the server to be kept safe. Problem here might be that the server may be overtrigged by a lot of requests, overwhelmingly increased number of users may cause this or a potential Denial of Service (DoS) attack. For precaution, rate-limiting will be used in our API, which will keep the client sleeping until the server is available.

2- Processing Phase: After a request arrives to the server, a script should run and find proper faces in the photo. This is the phase that takes almost all time. Hence, it requires proper work to be synchronized. Since the use of the servers are limited, a large number of servers will not be allocated. However, if the number of users increases, new servers must be allocated so that inputs can be processed at the same time.

3- Response Phase: Server sends the scanned images to the same user who sent the input image. To send the image properly, optimal port configuration must be found to minimize the delay.

2.7 Boundary conditions

There are three boundary conditions. Starting application, terminating application and facing failure while using the application.

Initialization: Users must have the program on their phones. For uploading the photo, connection with the server must be established so internet connection is needed. If there are previous saved photos taken, they will be called from disk.

Termination: When the program is terminated, any running processes on the server by the user will also be terminated. The photos taken will be saved to the local disk. All unsaved photos and information will not be saved on disk.

Failure: There are many cases for this part. If the camera can not be used due to battery percentage, there will be failure to take photos. If internet connection is dropped while uploading the photo into the server, the program will wait for the connection and the user will be informed. If one of the features in the program stopped working like a ui error, the user will be informed. If users do not have enough disk space to save new photos, users will not be able to save photos and will be informed. But if the user cleans the disk space while the program is working in the background, the program starts saving photos to disk. If the program has a new software update, the user must be informed and upload the program to use it properly.

3. Subsystem Services

In this section, components and subsystems of our system are described in detail. The main two subsystems of the SEPS application are client and server, therefore the app will be formed by the interaction of these two subsystems.

3.1 Client

As our application is designed for the devices with Android OS, client can be used through the phones with respective operating systems. The client includes view and controller subsystems which are responsible for user interface operations and creating requests to interact with the server, respectively.

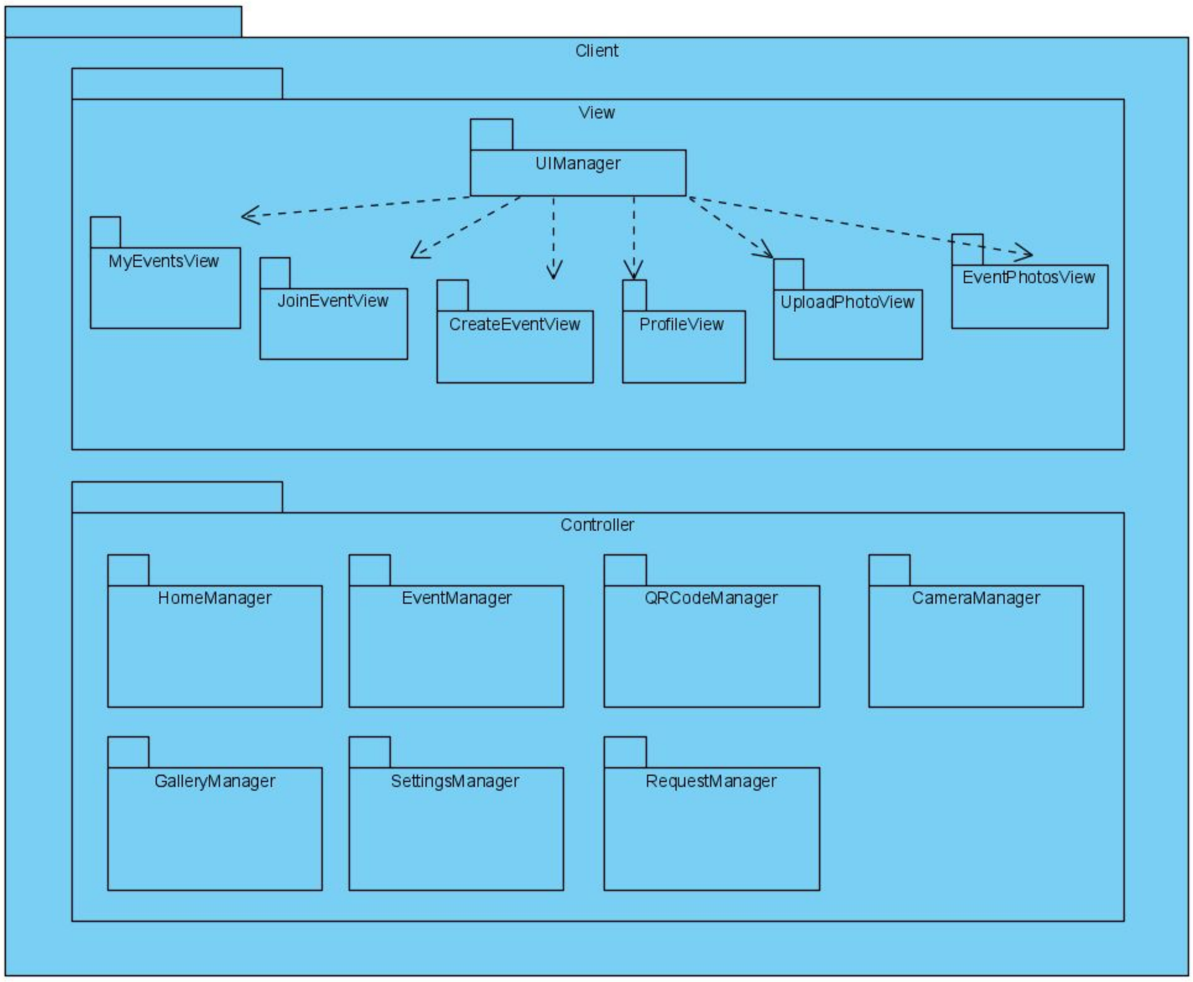

Figure 3: Subsystem Decomposition: Client

3.1.1 View

The view subsystem contains user interface pages and the manager class to handle them.

UIManager:

The class for managing the transitions among all view classes.

MyEventsView:

This view class displays the event selection screen, where all the events of the user (both created by and attended) are listed. The view uses EventManager for interacting with the server and getting the needed data from the database.

JoinEventView:

This view class displays the event joining screen, where the user needs to scan the QR code to be able to attend the event. The view uses CameraManager to take photos of the QR and QRCodeManager to decode QR code.

CreateEventView:

This view class displays the event creation screen, where the user is asked to fill details of the event. The view uses QRCodeManager to create new QR for the new event and EventManager to create the event with the given data.

ProfileView:

This view class displays the profile page screen, where the user can change some of the app settings according to the preference. The view uses SettingsManager to change and save the new preferences.

UploadPhotoView:

This view class displays the screen for uploading the photo for the selected event. The view uses CameraManager and GalleryManager to get the photo from the user and create the according request.

EventPhotosView:

This view class displays the screen for event photos. The view uses EventManager to get the list of Images for the selected event.

3.1.2 Controller

The controller subsystem contains manager classes that handle the creation of requests to interact with the server.

HomeManager:

This controller class manages the initial start of the application by creating the first interaction with the server (for user authentication).

EventManager:

This controller class manages the all event related communication of the client and the server. The situations where the EventManager module is called by one of the view modules are: the new event is created, all events of the user listed, the user requests to join the event, new photos are added to the event.

QRCodeManager:

This controller class manages the new QR code creation for the newly created events and decoding the QR codes that are scanned.

CameraManager:

This controller class manages the configuration of the device camera for taking photos.

GalleryManager:

This controller class manages the image selection from the gallery process.

SettingsManager:

This controller class manages the changing and storing process of the application settings by the user.

RequestManager:

This controller class manages the all requests; communicates with the server after creating the according request, notifies back the views after getting the response from the server.

3.2 Server

Server handles the computation and data management as expected. In our design we have a tier based approach which separates data management and computation..

3.2.1 Data Tier

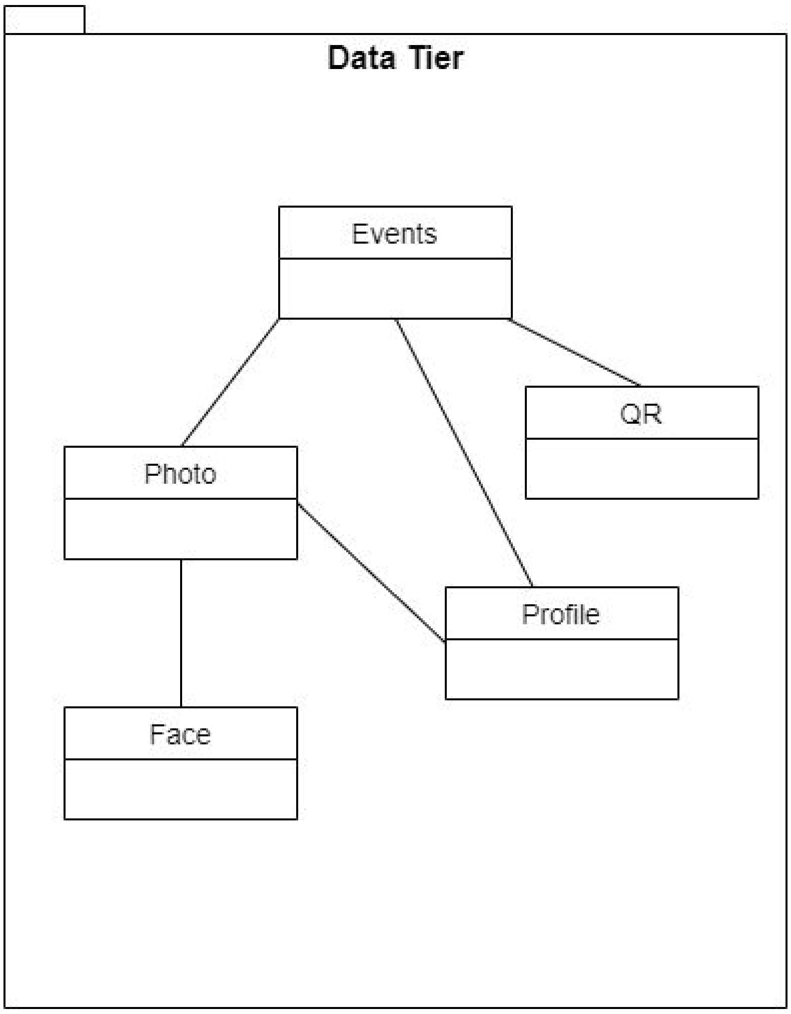

Figure 4: Subsystem Decomposition: Data Tier

Event:

Events have relations with Photo, QR and Profile. Those relations are defined in corresponding order, photos taken in event , QR code of event and participants of event.

Photo:

Photos have relation with Face, Event and Profile. Those relations are defined in corresponding order, faces in the photo, event photo belongs to and profile of owner of the photo.

Profile:

Profiles have relation with Event and Photo. Those relations are defined in corresponding order, events the profile participated in and photos the profile uploaded.

Face:

Faces only have relation with Photo, the relation means the face belongs to that particular photo.

QR:

QR only has a relation with Event, the relation means the QR belongs to that particular event.

3.2.2 Application Tier

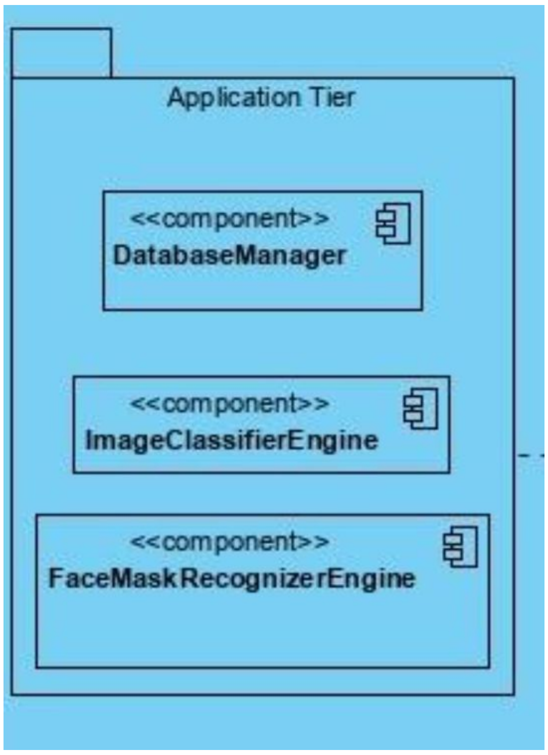

Figure 5: Subsystem Decomposition: ApplicationTier

DatabaseManager:

This component is responsible for managing models in the Data tier. Events, Photos, Profiles, Faces and QR are tables in the database which will be open for create, read, update and delete functions. Aforementioned functions will be performed by using this component.

ImageClassifierEngine:

This component is responsible for labeling unknown faces in the photos even though the owner of the face does not exist in the data tier. This labeling process will enable both image search from photo and image search from text or tag options.

FaceMaskRecognizerEngine:

This component is similar to **ImageClassifierEngine** component except it will be only used for recognizing the faces with a mask. The separate engines for each image recognizer will provide better performance with the use of different models for each recognizer.

4. Consideration of Various Factors in Engineering Design

Public Health

The application's face recognition model should take COVID-19 mask regulations into consideration.

Public safety

The application should guarantee security of the event zone by the usage of QR code.

Global Factors

The application's implementation will take global trends on software solutions into account. The most trending photograph sharing applications will be analyzed in order to ensure the quality desired by users.

Cultural Factors

The application will provide users with the opportunity to easily find photographs that they do not even know but they are in. Moreover, the users will not search for their photos from other attendants, they will be able to access them directly via the application. The face recognition algorithm should be unbiased for any kind of race, religion and other cultural values.

Social Factors

The application should use ethical constraints for determining uploadable photographs. Any abusive usage should be reported and deleted from the database.

Environmental Factors

The devices being used in the design process should be efficient in terms of electric consumption.

Economic Factors

The application should be free of charge.

Factors	Priority Point(out of 10)
Public Health	4
Public safety	10
Global Factors	6
Cultural Factors	7
Social Factors	5
Environmental Factors	3
Economic Factors	2

Figure 6:Factors and Priority Points

5. Teamwork Details

5.1 Contributing and functioning effectively on the team

The developer team had distributed design roles to each member of SEPS. Each of the members had contributed to every section of the design, although a leader for a specific role had always been assigned.

Briefly, Alperen contributed to specification of design goals and purposes and of the SEPS system, while considering other factors for the design. Burak planned the software/hardware mapping and subsystem decomposition. Erkin elaborated the data management and control scheme of the SEPS's system. Taner prepared the design plan for the server side of the subsystems. Samir designed the SEPS's client side subsystems.

5.2 Helping creating a collaborative and inclusive environment

We will be using several tools for proper communication among us, the number of such communication tools might also increase as time goes on throughout the semester. We will be using Whatsapp messages, e-mail messages, Zoom calling and phone calling for basic communications. However, since file sharing and editing is too hard for teamwork use on these communication tools, we will be using shared **Google Drive** folders and **Google Docs** for this purpose. We also divide projects into parts and distribute them among us and set due dates in the future so everyone in the group has certain work and time to do their work. For the, mainly implementation parts, we will use **GitHub** to share and synchronise project's separate code segments, which is constructed already.

5.3 Taking lead role and sharing leadership on the team

Our team utilizes delegated leadership, in which a different leader is elected for every stage. The testing roles will also be shifted during design progress, where at the end we will undergo testing as the whole team. The planned workload share is as follows.

5.3.1. Work package 1(Current Work package)

Leader: Samir Ibrahimzade
Major milestones and deliverables:Front-End Implementation, High-Level Design Report
Start Date: November, 21, 2020
End Date: Jan 21, 2021

5.3.2. Work package 2

Leader: Alperen Koca
Major milestones and deliverables: Back-End Implementation, Low-Level Design Report
Start Date: December, 21, 2020
End Date: Feb 8, 2021

5.3.3. Work package 3

Leader: Mehmet Erkin Şahsuvaroğlu

Major milestones and deliverables: Database & Server Installation, Testing

Start Date: Feb 9, 2021

End Date: TBD

5.3.4. Work package 4

Leader: Burak Yeni

Major milestones and deliverables: Final Reporting, Presentation & Demo

Start Date: March 15, 2021

End Date: April 30, 2021

6. Glossary

Abbreviation list can be seen from the section 1.3..

7. References

[1] The Code affirms an obligation of computing professionals to use their skills for the benefit of society. (n.d.). Retrieved November 1, 2020, from https://www.acm.org/code-of-ethics

[2]Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010