Bilkent University

Department of Computer Engineering

Senior Design Project

SEPS

Low-Level Design Report

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

Supervisor: Shervin Rahimzadeh Arashloo
Jury Members:

Low Level Design

February 8, 2021

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 CS492.

1.Introduction

Social media users have been increasing gradually. Photo sharing is one of the main parts of it. For example Instagram had 90 million users in 2013 and now it has passed 1 billion users. [1] This data indicated that lots of people like photo sharing worldwide. However there was no photo sharing application for special events specifically. Our initial aim is providing people with a number of photos who attend a specific event like a wedding or coffee festival. Some people who take photos may not catch the moment so they can reach photos taken from different people who attend the same event by using our application called SEPS.

This application also prevents privacies of the users. For example people who are not authorized for selected events cannot join it. Therefore users have freedom to share their photos related to their events. Another innovative feature of our application is detecting the people who do not wear face masks in case of any pandemic. This feature can be developed and maintained with the help of medicine in the future for the possible contiguous diseases. SEPS also collects the photos which are not part of any event and label their tags like horse racing photos, music festival photos etc. After labeling the photos, SEPS will be able to create a new event for the photos which have close relationships by analyzing the attributes via machine learning algorithms such as deep learning network and Bayesian network. With understandable and easy user interface our customers will have an enjoyable experience hopefully.

This report aims to provide a good overview of the low level architecture and design of our project. Object design trade-offs will be explained firstly. After that interface documentation guidelines will appear with the engineering standards and acronyms. In the second part packages will be explained in detail. Therefore our class interfaces of the project will be shown in detailed explanation in the third part such as class names, class attributes and methods.

1.1 Object Design Trade-Offs

1.1.1 Memory Vs Time

SEPS needs lots of data such as personal information of users and photos. Many photos included in the one of the events and many photos need to be classified in terms of properties. In

order to do this SEPS stores branches of images in a training database. Classification of the photos that are uploaded by users requires processing time. In addition to this transferring data between data server to machine learning server also needs time. Our aim is to reduce the amount of time consumption by using the efficient deep learning methods and using the Microsoft Azure servers.

1.1.2 Functionality vs Usability

Our application is a user centric because it serves adequate features to users. In other words users will not see the disturbing buttons or pages. Despite SEPS containing robust functionalities such as detecting unmasked people in photos or classifying images in terms of genre, these functions work in the background of the software. They do not interact with any user directly, so SEPS favor usability over functionality.

1.1.3 Scalability vs Performance

One of the design issues of our application is providing a robust performance to each user from any part of the world. If many users use SEPS simultaneously it may decline the performance of server requests. Therefore our goal is to build a powerful system that can handle many requests at the same time from different users.

1.1.4 Availability vs Reliability

SEPS uses the remote servers in order to keep lots of data and process them easily. However if these servers shutdown or in repair schedule our software may not work efficiently. Therefore our system should check the availability of servers and should take precautions before possible data losses occur.

1.2 Interface Documentation Guidelines

This is an example class interface for our documentation. It includes the name of the class, instance variables of class and methods of it.

Class Name: SampleClass
Description Of SampleClass
Attributes: private int SCid private String SCname public SCsize
Methods: public int getSCid():Returns SCid public String getSCname():Returns the name of SampleClass instance

1.3 Engineering Standards

For the description of class interfaces, use case models and subsystem decompositions we used UML guidelines[2]. For the referencing and citations we followed IEEE standards [3].

1.4 Definitions, Acronyms and Abbreviations

GUI: Graphical User Interface. Type of user interface that allows users to interact with electronic devices through graphical icons.

SDK: Software Development Kit. Collection of software used for the development of applications

UI: User Interface. UI is the space where human - computer interactions exist.

UML: Unified Modeling Language. General language for modeling in the area of software engineering. It consists of diagrams to model various aspects of the software.

2. Packages

The low level subsystem of SEPS is composed of two main subsystems, Client and Server subsystems. Within these subsystems, lower scoped subsystems operate to maintain the application. Client system works via View and Controller subsystems within the Client subsystem. The Server architecture subsystem consists of a tiered hierarchy of its subsystems, Data tier and Application tier. In the conducted project, we used a structured approach to arrange the connections between Client and Server subsystems. Client system operates by arrangement of UI screens and interconnection between the Client and Server in an abstracted manner. On the other hand, the Server system has a tier based approach which separates data management and computation

2.1 Client

Designation of the application is focused on the Android operating system, therefore presentation of the client side consists of 2 subsystems, View subsystem and Controller subsystem. These subsystems work interoperably, to undergo the user interface operations and to create requests to interact with the server. User interface options will be handled via View subsystem, while request handling and server interactions of Client will be dealt via Controller subsystem.

2.1.1 View Subsystem

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

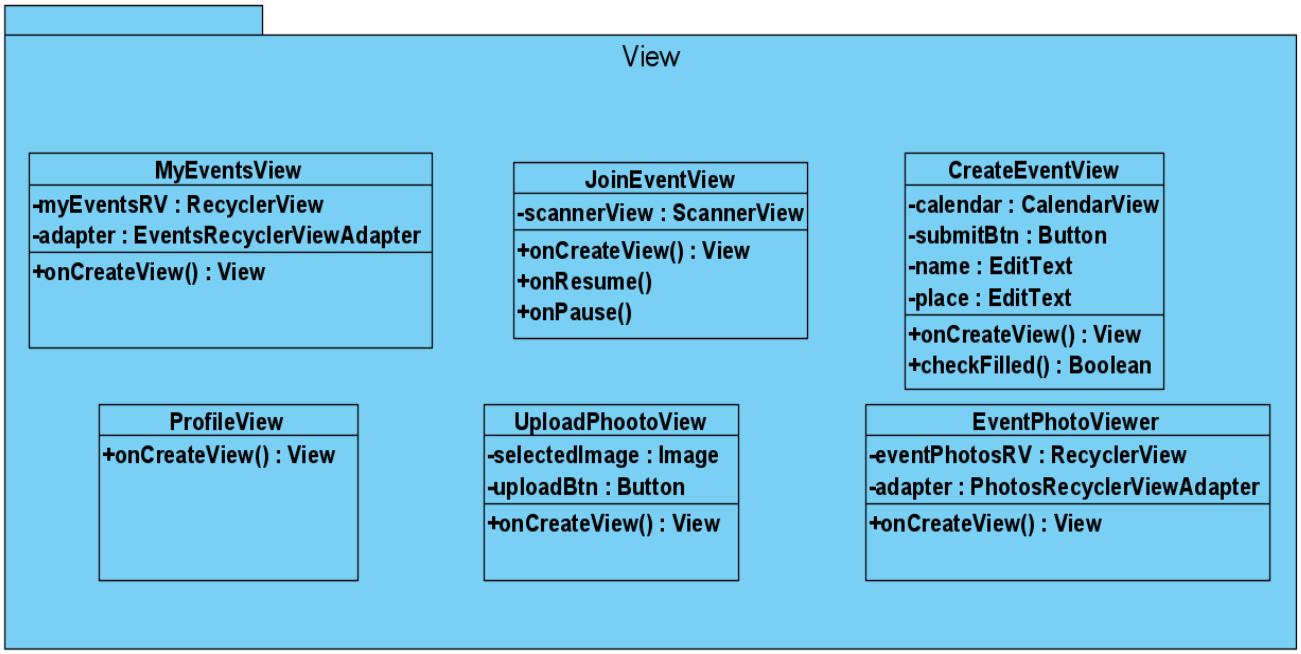

Figure 1: Subsystem Decomposition: View.

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, to retrieve event information of the specific client.

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. The class directly interacts with the server.

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. The view subsystem will transact image data to server subsystem via the s

EventPhotosView:

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

2.1.2 Controller Subsystem

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

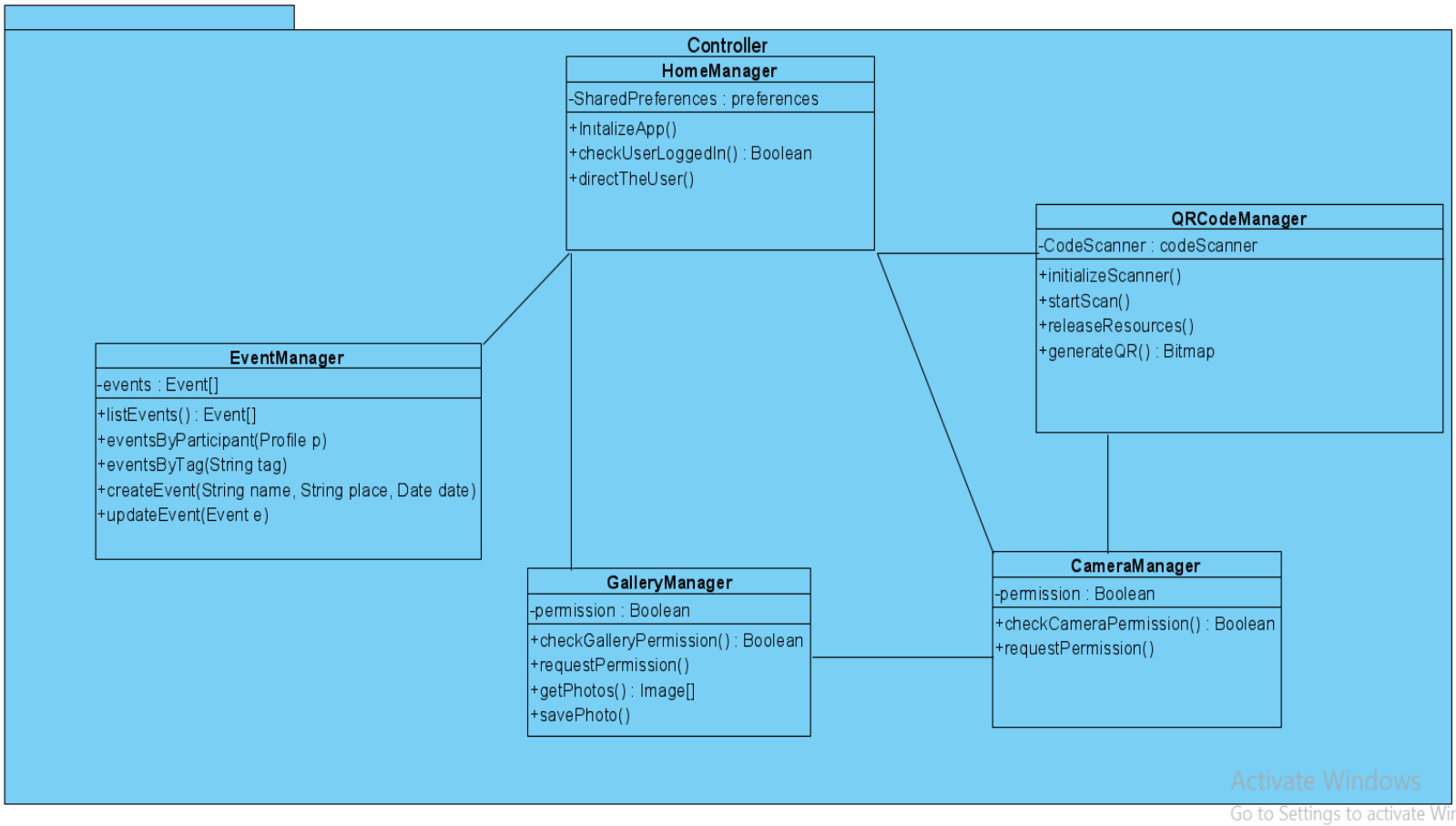

Figure 2: Subsystem Decomposition: Controller

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.

2.2 Server

Server has two layers, Data Tier and Application Tier. In the Data Tier, all objects are stored there as expected. Each time, user wants to access the system, Data Tier handles corresponding events, photos and profiles. In Application Tier, image classifications and mask recognition are handled in collaboration with the database manager.

2.2.1 Data Tier

Data Tier is responsible for managing interactions within the database. Data Tier keeps connected to Application Tier to get the requested data.

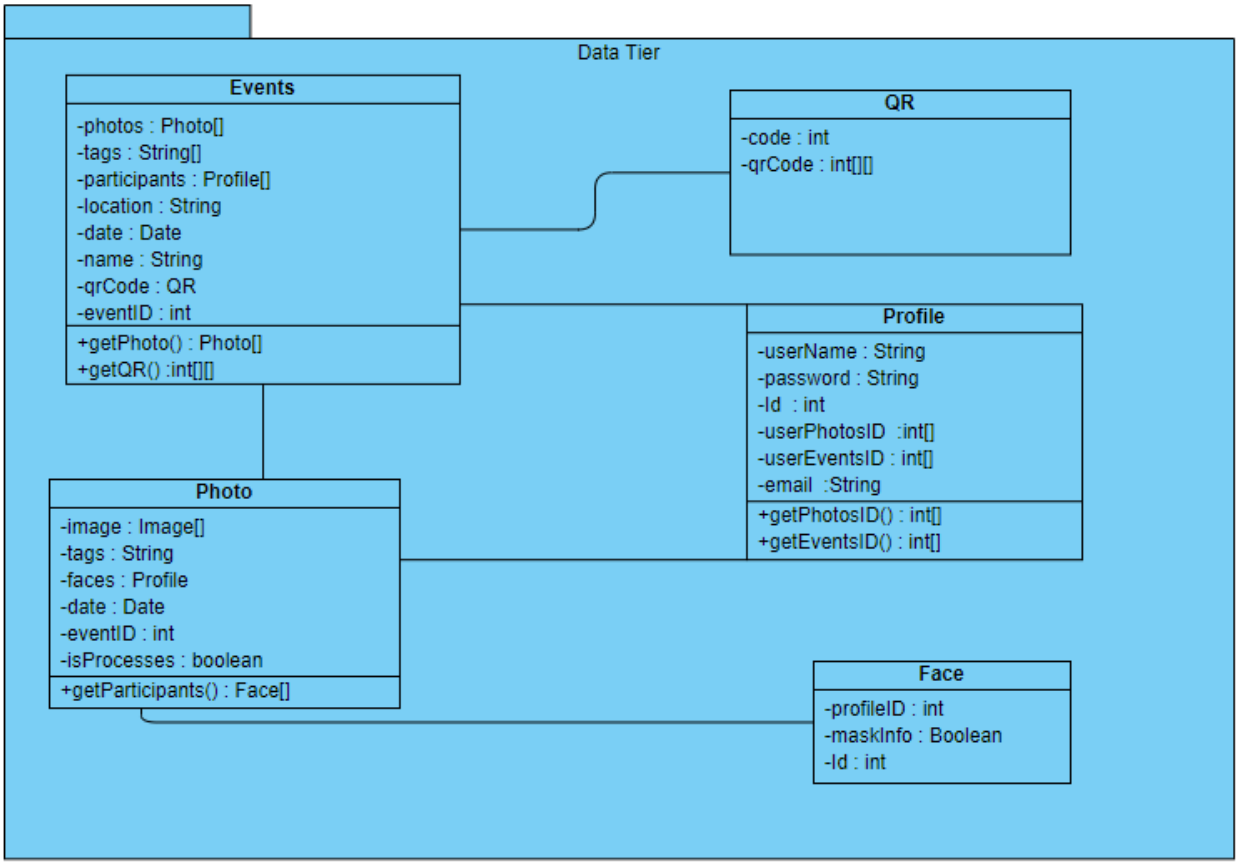

Figure 3: Subsystem Decomposition: Data Tier

Event:

Events with their IDs stored within the database. Events have relations with Photo, QR and Profile.

Photo:

Photo includes images stored in the database. Photos have relation with Face, Event and Profile.

Profile:

Profile of the Users with their specific IDs connected to the database. Profiles have relation with Event and Photo.

Face:

Faces from the corresponding images stored in the database. Faces only have relation with Photo, the relation means the face belongs to that particular photo.

QR:

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

2.2.2 Application Tier

Application Tier is responsible for the majority of operations in SEPS. This part of the server communicates the different APIs to keep the system ready to use. When the client sends a request, the request will be headed to the corresponding service.

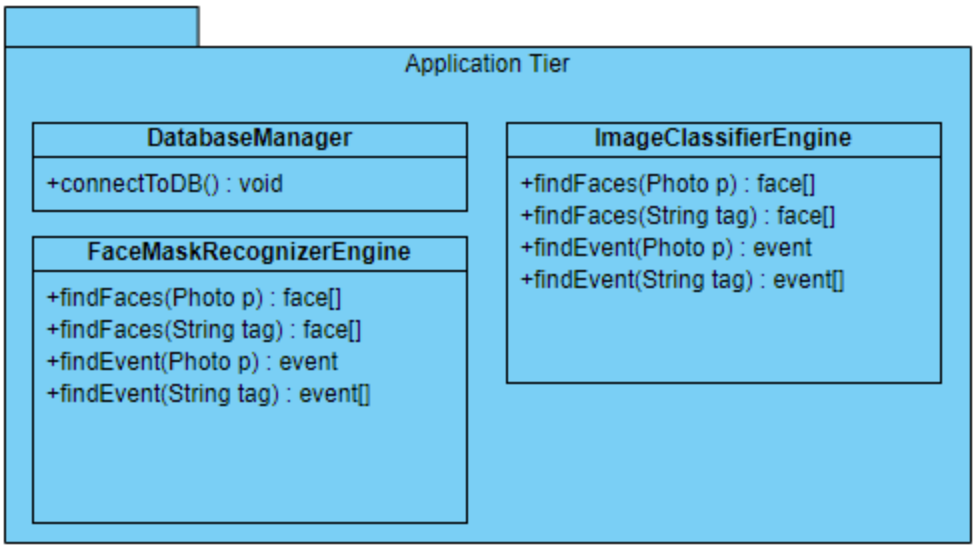

Figure 4: Subsystem Decomposition: ApplicationTier

DatabaseManager:

This component is responsible for managing models in the Data Tier and provides a connection between them. Events, Photos, Profiles, Faces and QR are tables in the database that are kept in control for their different purposes such as create, delete, open and close operations.

ImageClassifierEngine:

This component is responsible for determining the faces in the photo whether or not the owner of the face exists in the data tier. This 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 process determines the faces with masks from the image. The separate engines for each image recognizer will provide better performance with the use of different models for each recognizer.

3 Class Interfaces

The description, attributes and the methods of the classes are listed as separate tables in this section.

3.1 Client

The client includes view and controller subsystems which are responsible for user interface operations and creating requests to interact with the server, respectively.

3.1.1 View

The view subsystem contains user interface pages.

Class MyEventsView

This View class is responsible for displaying the my events page by getting the layout from xml file.

Attributes

private RecyclerView myEventsRV private EventsRecyclerViewAdapter adapter

Methods

public View onCreateView()	Creates a view and integrates it with the according layout file.

Class JoinEventsView

This View class is responsible for displaying the join events page by getting the layout from xml file.

Attributes

private ScannerView scannerView

Methods

public View onCreateView() public void onResume() public void onPause()	Creates a view and integrates it with the according layout file. Starts scanner preview when. Releases camera resources.

Class CreateEventsView

This View class is responsible for displaying the create events page by getting the layout from xml file.

Attributes

private CalendarView calendar private Button submitBtn private EditText name private EditText place

Methods

public View onCreateView() private boolean checkFilled()	Creates a view and integrates it with the according layout file. Checks and returns if all EditTexts are filled(name of the event and place of the event)

Class ProfileView

This View class is responsible for displaying the profile page by getting the layout from xml file.

Methods

public View onCreateView()	Creates a view and integrates it with the according layout file.

Class UploadPhotoView

This View class is responsible for displaying the upload photo page by getting the layout from xml file.

Attributes

private Image selectedImage private Button uploadBtn

Methods

public View onCreateView()	Creates a view and integrates it with the according layout file.

Class EventPhotosView

This View class is responsible for displaying the event photos page by getting the layout from xml file.

Attributes

private RecyclerView eventPhotosRV private PhotosRecyclerViewAdapter adapter

Methods

public View onCreateView()	Creates a view and integrates it with the according layout file..

3.1.2 Controller

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

Class EventManager

This Manager class is responsible for event related functionalities. Creating, joining, deleting, editing events are managed by EventManager class.

Attributes

private Event[] events

Methods

public Event[] listEvents()	Returns all events.
public Event[] eventsByParticipant(Profile p)	Returns events of the particular participant.
public Event[] eventsByTag(String tag)	Creates a new Event with the given name, place and date.
public Event createEvent(String name, String place, Date date)	Deletes selected event.
public void deleteEvent(Event e)	Updates the details of selected events.
public void updateEvent(Event e)	

Class QRCodeManager

This Manager class is responsible for creating new qr codes, decoding the scanned codes.

Attributes

private CodeScanner codeScanner

Methods

public void initializeScanner()	Initializes scanner by checking required permissions.
public void startScan()	Starts scanning by using the camera.
public void releaseResources()	Releases the resources of the camera while closing the activity.
public Bitmap generateQR()	Creates a new Bitmap of the QR code.

class CameraManager

This Manager class is responsible for handling all functions of the camera needed for the app.

Attributes

private Boolean permission

Methods

public Boolean checkCameraPermission()	Checks and returns whether camera permission is granted or not.
public void requestPermission()	Requests camera permission from the user.

class GalleryManager

This Manager class is responsible for all handling gallery functionalities.

Attributes

private Boolean permission

Methods

public Boolean checkGalleryPermission()	Checks and returns whether gallery permission is granted or not.
public void requestPermission()	Requests gallery permission from the user.
public Image[] getPhotos()	Returns the images selected by the user from the gallery.
public void savePhoto()	Saves the photo to the gallery.

class HomeManager

This Manager class is responsible for initializing the app and checking the user authentication.

Attributes

private SharedPreferences preferences

Methods

public Boolean checkUserLoggedIn() public void directTheUser() public void initializeApp()	Checks and returns whether the user signed in or not. Directs user to the LoginPage or MyEvents page according to the login info. Loads preferences of the user if the user is signed in.

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

class 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.

Attributes

private Photo[] photos

private String[] tags

private Profile[] participants

private String location

private Date date

private String name

private QR qrCode

Private int eventID

Methods

Public Photo[] getPhoto()	Returns photos of the event.
public int[][] getQR()	Returns QR code of the event.

class Face

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

Attributes

private int profileID

private Boolean maskInfo private int Id

class 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.

Attributes

private String userName private String password private int Id private int[] userPhotosID private int[] userEventsID private String email

Methods

Public int[] getPhotosID() Public int[] getEventsID()	Returns ID of the photos . Returns ID of the events.

class 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.

Attributes

private Image[] image

private String[] tags

private Profile uploader

private Face[] faces

private Date date

private int eventID

private boolean isProcessed

Methods

Public Face[] getParticipants()	Returns faces of the participants in the photo.

class QR

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

Attributes

private int code

private int[][] qrCode

3.2.2 Application Tier

class 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.

Methods

public face[] findFaces(Photo p)	Returns recognized faces in the photo.
public face[] findFaces(String tag)	Returns images related to the tag.
Public event findEvent(Photo p)	Returns recognized event photos belong to.
Public event[] findEvent(String tag)	Returns events related to the tag.

class DatabaseManager

DatabaseManager 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.

Methods

public void connectToDB()	Establishes connection to DB.

class 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.

Methods

public face[] findFaces(Photo p)	Returns recognized faces in the photo.
public face[] findFaces(String tag)	Returns images related to the tag.
Public event findEvent(Photo p)	Returns recognized event photos belong to.
Public event[] findEvent(String tag)	Returns events related to the tag.

4.Glossary

Activity: Activity is the part of software that is related with only users' actions like GUI in the Android operating system. [6]
Event: Social organization that oriented people can be a part of it in SEPS application.
Profile
QR: Symbolic code that is capable of handling all types of data such as numeric and alphabetic characters. [5]
Machine Learning: Branch of artificial intelligence used for building applications which learn from data and improve itself over time. [4]

5. References

[1] “Active Users of Instagram”. Instagram monthly active users | Statista . [Accessed 8- Feb- 2021].

[2] “Unified Modelling Language”. What is Unified Modeling Language (UML)? (visual-paradigm.com) . [Accessed 8- Feb- 2021].

[3] “IEEE REFERENCE GUIDE”. IEEE-Reference-Guide.pdf . [Accessed 8- Feb- 2021].

[4] “What is Machine Learning”. What is Machine Learning? | IBM . [Accessed 8- Feb- 2021].

[5] “Features of QR Code”. QR Code features | QR Code.com (archive.org) . [Accessed 8- Feb- 2021].

[6] “Activity”. Activity | Android Developers . [Accessed 8- Feb- 2021].