CS 492

Senior Design Project

Low Level Design Report

Project Name: LikedIt

Students

Zeynep Hande Arpakuş - 21501525

Zeynep Ayça Çam - 21502269

Muhammet Said Demir - 21602021

Zeynep Nur Öztürk - 21501472

Elif Beril Şayli - 21502795

Supervisor

Hamdi Dibeklioğlu

Jury Members

Özcan Öztürk

Selim Aksoy

1. Introduction **3**

1.1 Object design trade-offs 3

1.1.1. Space vs Time 3

1.1.2 Usability vs Privacy 4

1.1.3 Performance vs Privacy 4

1.1.4 Performance vs Functionality 4

1.2 Interface documentation guidelines 4

1.3 Engineering standards (e.g., UML and IEEE) 5

1.4 Definitions, acronyms, and abbreviations 5

2. Packages **5**

2.1. Presentation Package 5

2.1.1. Controller Package 5

2.1.2. Model Package 6

2.1.3. Web View Package 7

2.2 Application Package 8

2.2.1. Storage Manager Package 8

2.2.2. Analysis Package 9

2.2.3. App Manager Package 9

2.3 Data Package 10

3. Class Interfaces **11**

3.1 Model 11

3.2 View 13

3.3 Controller 15

4. Glossary **19**

5. References **20**

1. Introduction

Nowadays, human-computer interactions have been increasing rapidly. This interaction creates new social environments for people to share their lives. “With the advent of mobile applications and social websites such as YouTube, Vine, and Vimeo, we have observed an increase in the number of online videos shared by people expressing. To give you a better idea of how popular these websites are, more than 300 hours of video is uploaded to YouTube every minute.” [1]. The amount of users in these platforms increases attention from companies, clients, and researchers. Creating and influencing people’s experiences has become a valuable differentiation strategy for the owner of videos. Therefore, research data from these platforms are necessary and attractive in today’s world.

The purpose of our Senior Design Project in Bilkent University Department of Computer Engineering is to contribute research about human-computer interaction when people watch online videos. We will capture physiological reactions in real-time to accurately research how humans appreciate. One of the strongest indicators for an understanding of appreciation is the human’s face. The facial expression represents one of the most important non-verbal means of communication.

This application provides a feedback report to the video owner about how much the other users liked or disliked the video scene by scene, where do the users look at in the video mostly, which groups liked their video mostly. The application can be used by every age, gender, and ethnic group who watches or records videos. Furthermore, in our application, we recommend videos that users might like in the future based on their reactions to the videos they watched.

1.1 Object design trade-offs

1.1.1. Space vs Time

Our system analyzes the videos which are taking from users when they watch videos. The system ends up having lots of frames to be analyzed at the end because each video would be at most 5-10 minutes in length and 30-60 frames per second. Open Face requires time to process user videos. During development, we tried to use optimal algorithms and tools such as we prefer GPU processing instead of CPU processing to reduce time consumption. Therefore, time optimization is our most fundamental goal. To minimize the processing time of videos, on the server-side, we allocate lots of space in terms of memory and disk.

1.1.2 Usability vs Privacy

Our system provides two options for using our application. The first option is using the cloud as a backend of our website, and the other one is using the local machine as a backend. The first option provides easy use of the application since it does not require any additional set up for application. However, if the user wants to use the cloud, he/she has to accept to send his/her facial data(image) to other machines. The second option is using the local machine as a backend of our website. In this way, facial data of users stay at a personal computer and do not share with any other party. However, this option requires additional setup before using the application, which can decrease the usability of our application.

1.1.3 Performance vs Privacy

As we mention in the 1.1.2, the system provides two option. If the user wants to give priority to its privacy, it affects performance because operations are done by the user's own computer. Our installer downloads necessary programs to the user's computer. There is a difference between doing operations in the cloud and personal computers for performance view. The cloud version is faster than the computer itself. So we allow users to choose its trade-off.

1.1.4 Performance vs Functionality

Our system has a significant amount of backend. Therefore, we processed our website with the MVC principle which directs our web page to the multi-page app rather than a single-page app. The performance of the single-page app is faster than the multi-page app and only the data is transmitted back and forth. However, the multi-page app can have unlimited documentation as a backend. Therefore, because our site has many features, the functionality of our program is better than a single-page app.

1.2 Interface documentation guidelines

In the documentation, all class names are singular and named with standard class name form like 'ClassName'. Also variable and method names follow the same form like 'variableName' and 'methodName()'. The class' descriptions follow the hierarchy as the class name and its description, its properties, and the explanations, and finally, its methods and explanations are listed. The detailed outline is provided below:

class RecommendedVideos
This class is created for recommending videos to the user according to the videos they liked
Properties
public string userToken

Methods
public int[] findLikedVideosDB(String userToken): This method finds videos the user watched and liked from the database and return their video ids as an array. *public int[] findRecommendation(int[ ] likedArr):* In this method videos to recommend are found according to videos in the likedArr list.

1.3 Engineering standards (e.g., UML and IEEE)

We followed the IEEE citation format for referencing our resources and UML design principles for class descriptions and diagrams. [2]

1.4 Definitions, acronyms, and abbreviations

Term	**Definitions, acronyms, and abbreviations**
Open Face	OpenFace is an open-source toolkit for facial analysis. OpenFace is a useful tool for computer vision, machine learning, and affective computing communities and will stimulate research in facial behavior analysis an understanding[3].
MVC	The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects. [4]
GPU	Graphics Processing Unit
CPU	Central Processing Unit
UML	Unified Modeling Language

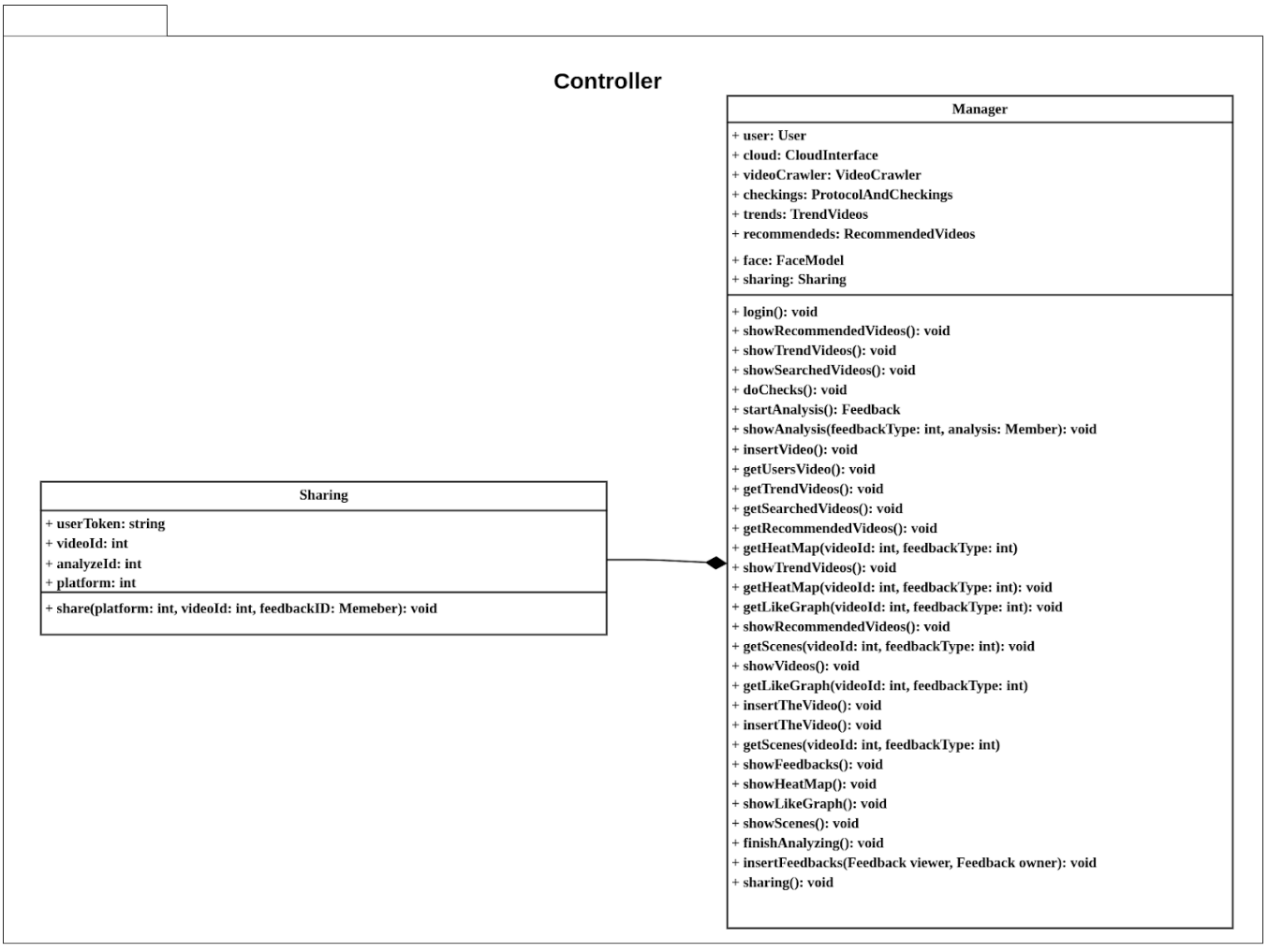
2. Packages

Our system is composed of 3 tier as packages. These packages consist of many sub-packages.

2.1. Presentation Package

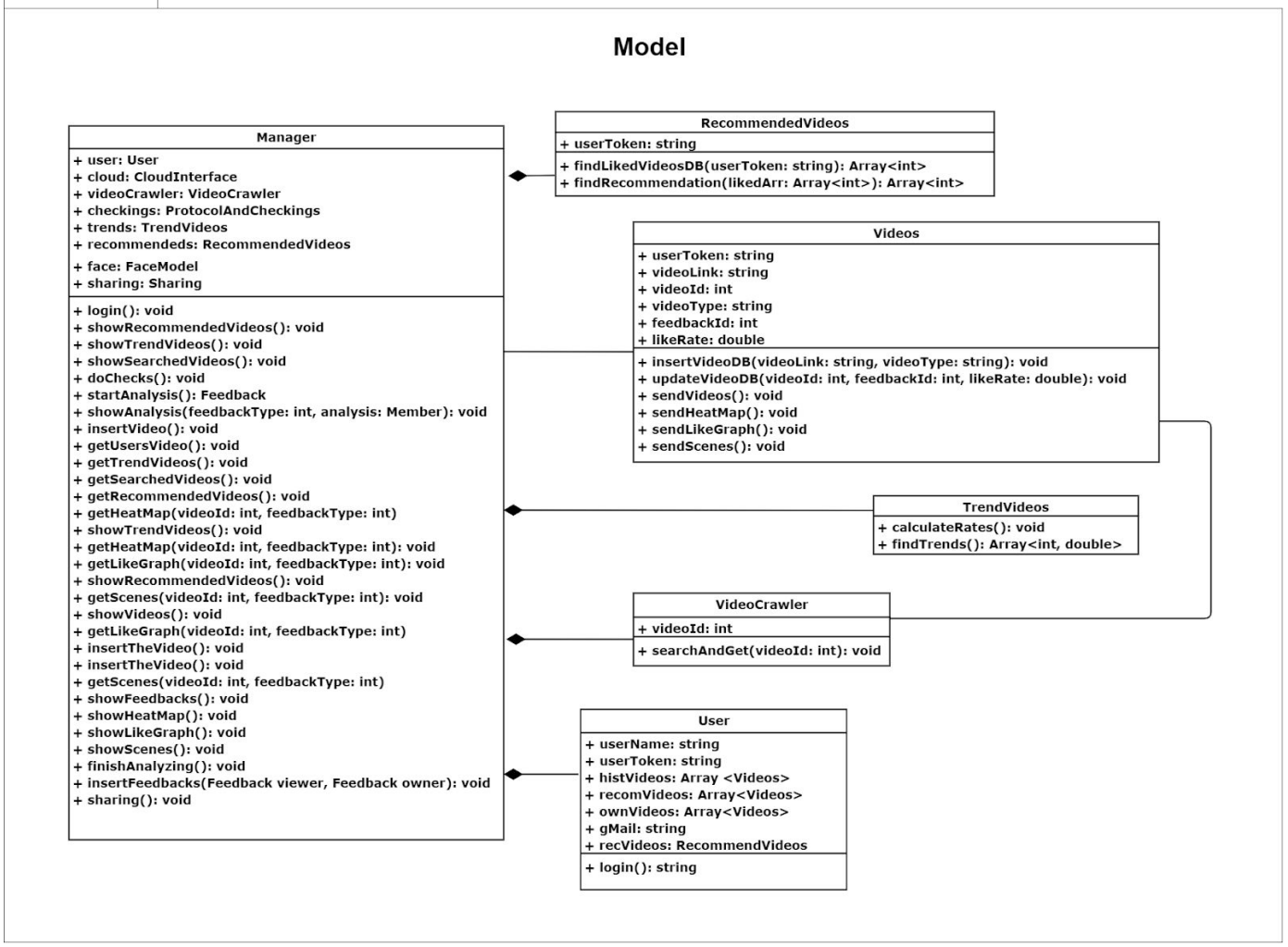
2.1.1. Controller Package

The controller package manages the client operations and the interaction between the client and the server.



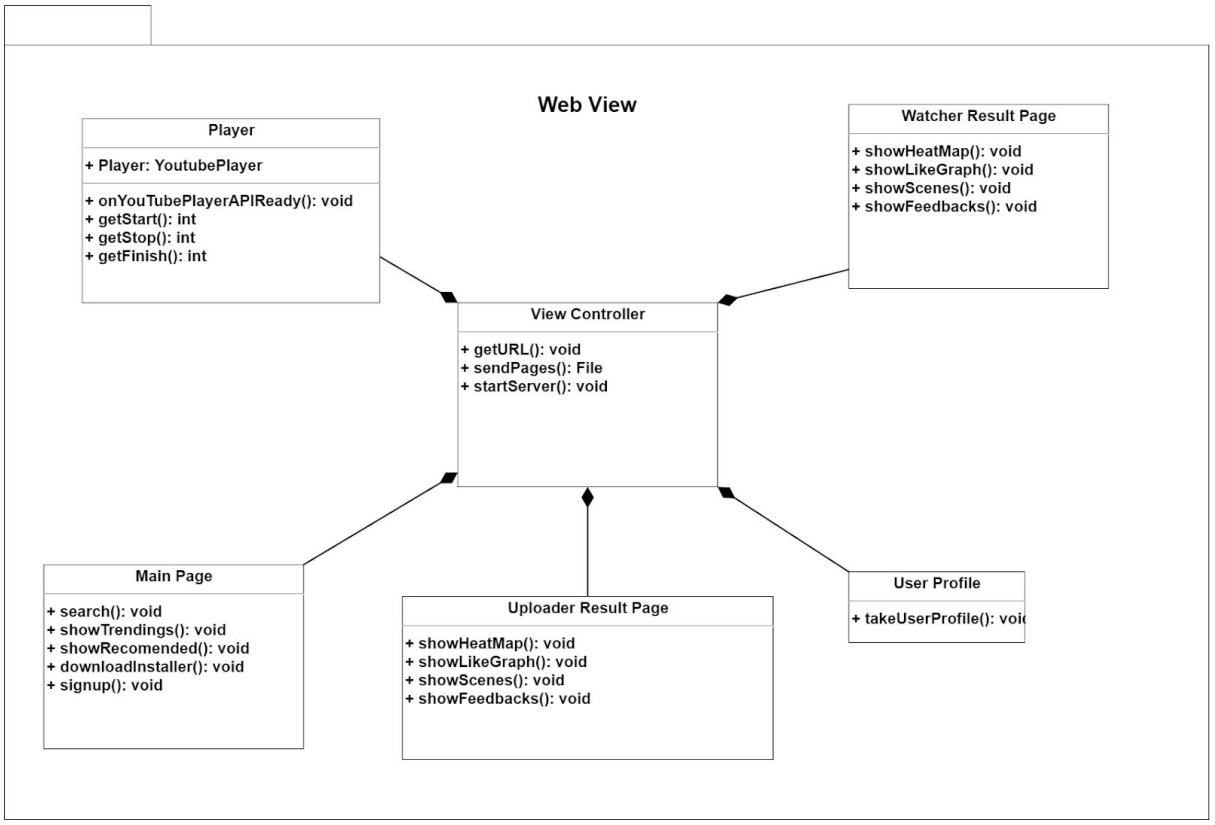
2.1.2. Model Package

The model package is the layer responsible for the main operations of our system. First of all, it is responsible for getting the video from the client-side. Then it generates feedbacks by analyzing the video. Finally, it sends the results to the client. And also manages the trending and recommended videos.



2.1.3. Web View Package

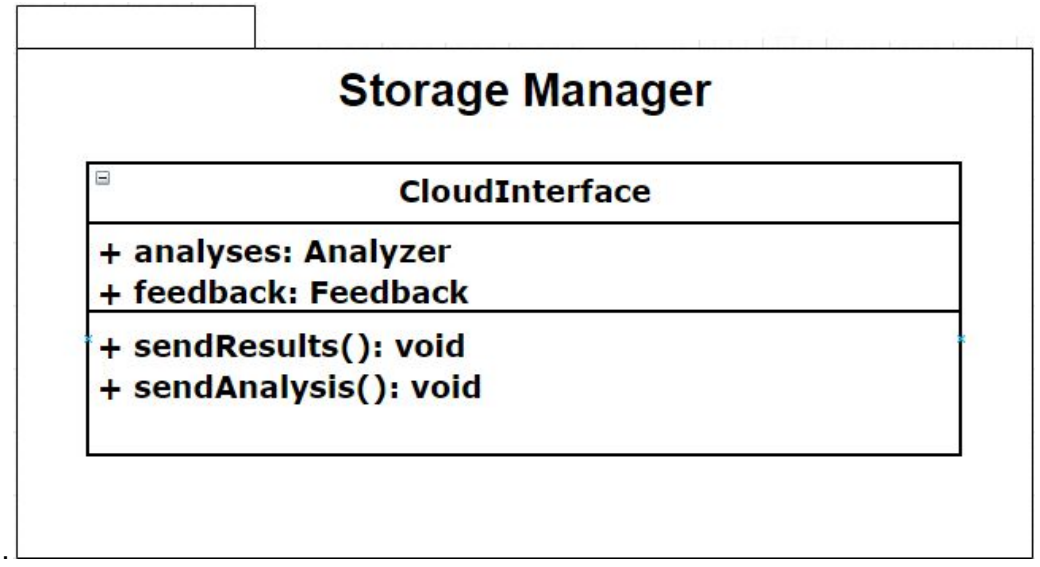
View package handles all user interface related operations as showing feedbacks, playing the videos, showing the profiles, and showing the main page.



2.2 Application Package

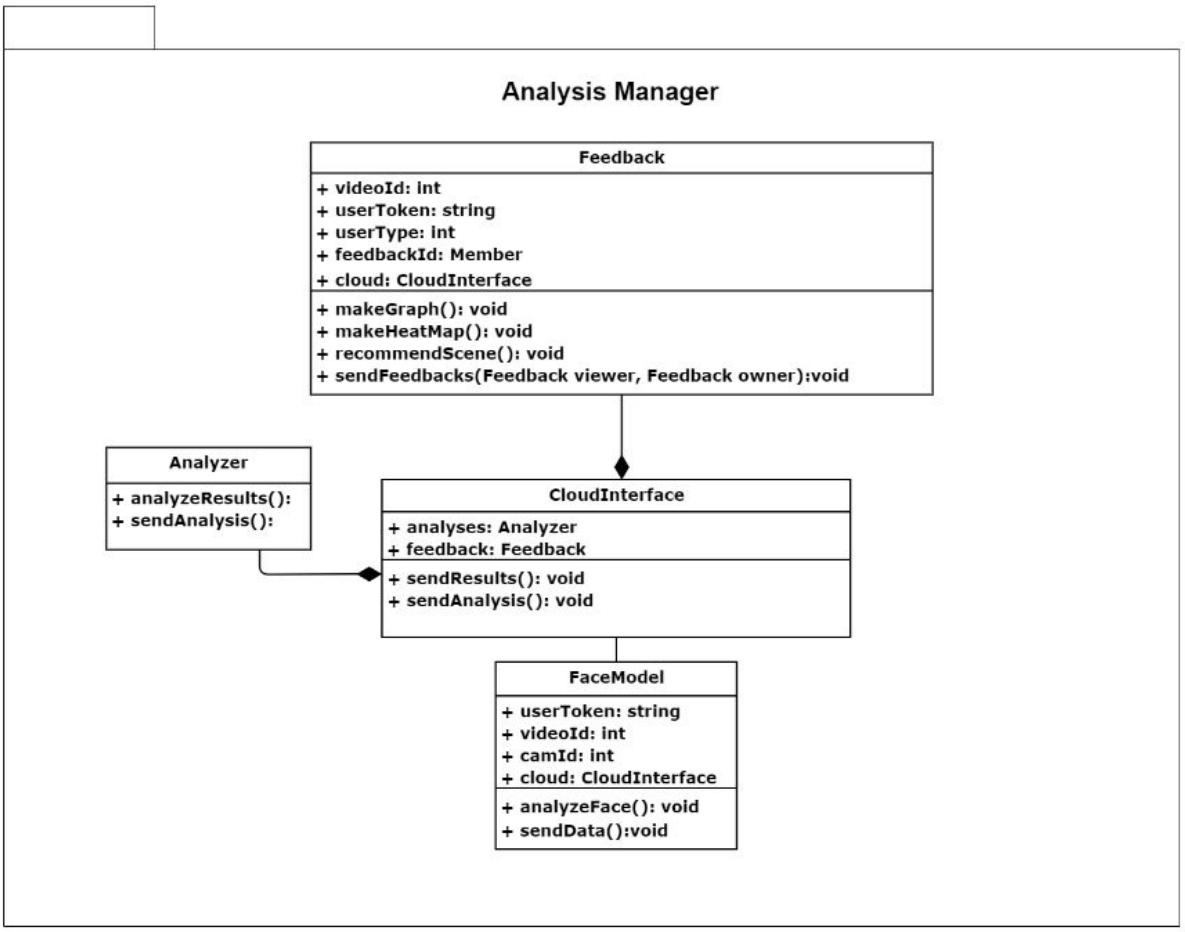
2.2.1. Storage Manager Package

The storage manager is related to storage within the cloud. Relations that are about storage files between cloud and web site are done in this package.



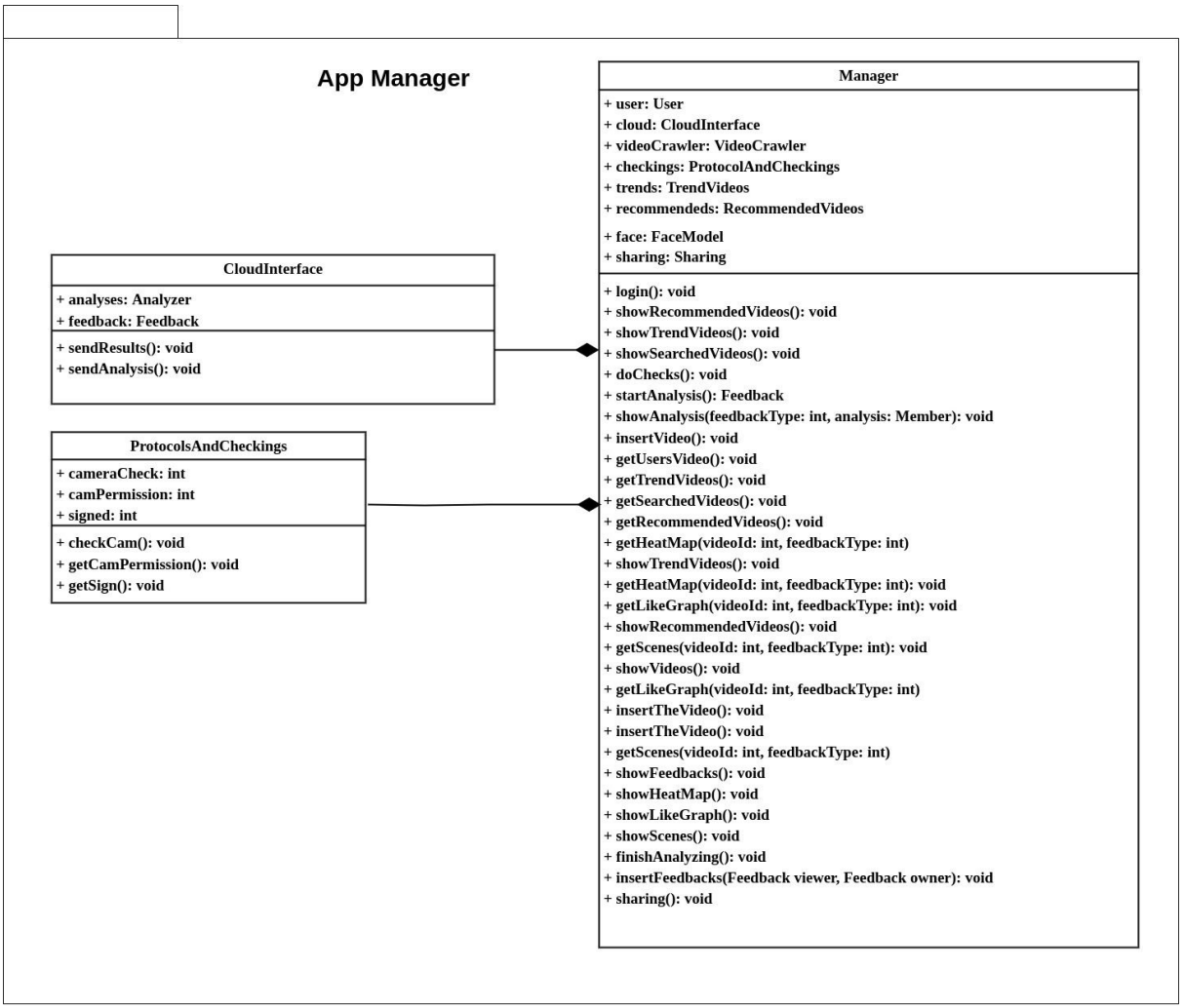
2.2.2. Analysis Package

The analysis manager has classes that are responsible for analyzing the face gestures of the user and create feedback for video. It involves cloud interaction, model that analyse and feedback class for showing results to the user.



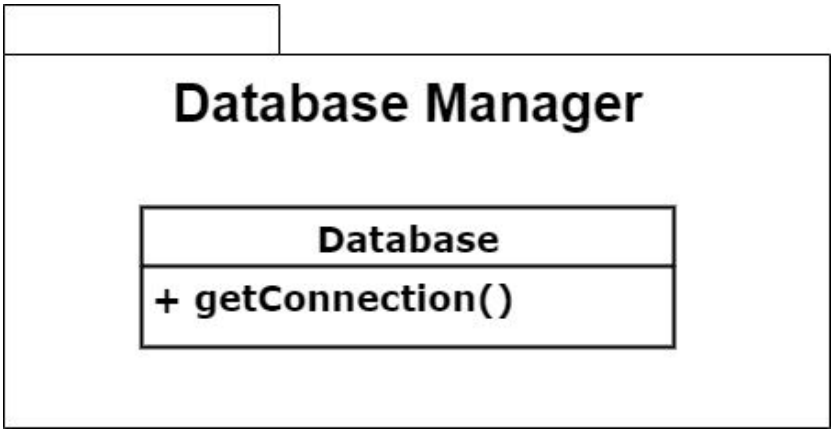
2.2.3. App Manager Package

The app manager package classes are responsible for the general management of the application. It creates a connection between different layers. It also creates a connection between different classes.



2.3 Data Package

Data package is related to whole data operations. The database class has a connection with each class. Within every class, whole data operations are managed by itself and this data package.



3. Class Interfaces

3.1 Model

class RecommendedVideos
This class is created for recommending videos to the user according to the videos they liked
Properties
public string userToken
Methods
public int[] findLikedVideosDB(String userToken): This method finds videos the user watched and liked from the database and return their video ids as an array. *public int[] findRecommendation(int[ ] likedArr):* In this method videos to recommend are found according to videos in the likedArr list.

class Videos
This class represents the videos watched from likedIt website. Inserting new videos to the database or updating their feedbacks are done in this class.
Properties
public string userToken public string videoLink public int videoID public string videoType public int feedbackID public double likeRate
Methods
public void insertVideoDB(String videoLink, String videoType): A new video watched from the website for the first time, is inserted to the database with its YouTube link and its type by this method. *public void updateVideoDB(int videoID, int feedbackID, double likeRate):* This method updates the feedback of a video in the database. *public void sendVideos():* This method is for sending the user video to the system if the one wants to help to improve the system.

public void sendHeatMap(): This method sends the heatmap of the video to the manager class for feedback. *public void sendLikeGraph():* This method sends the liking graph of the video to the manager class for feedback. *public void sendScenes():* This method sends the scenes liked most of the video to the manager class for feedback.

class TrendVideos
This method is created for finding the trend videos among the videos watched from likedIt website.
Methods
public void calculateRates(): This method pulls liking rates of videos from the database *public void findTrends():* This method selects the trend videos and sends them to the manager class.

class VideoCrawler
This class is for searching and pulling the video the user searched to the website.
Property
public int videoID
Method
public void searchAndGet (int videoID): This method is for searching and getting the video the user wants

class User
This class represents the users of likedIt website
Properties
public string userName public string userToken public Videos[] histVideos public Videos[] recomVideos public Videos[] ownVideos public string gMail

public RecommendVideos recVideos
Methods
public string login(): This method is for providing to user a login process

3.2 View

class Player
This class represents the frontend of video player
Properties
public YoutubePlayer Player
Methods
public void onYouTubeAPIReady(): This method is for connection with YouTube API. *public int getStart():* This method returns the time when the video is started. *public int getStop():* This method returns the time when the video is stopped. *public int getFinish():* This method returns the time when the video is finished.

class WatcherResultPage
This class represents the frontend of the feedbacks coming from the manager class
Methods
public void showHeatMap(): This method is for handling the frontend of the showHeatMap() method of the Manager class for the viewer of the video. *public void showLikeGraph():* This method is for handling the frontend of the showLikeGraph() method of the Manager class for the viewer of the video. *public void showScenes():* This method is for handling the frontend of the showScenes() method of the Manager class for the viewer of the video. *public void showFeedbacks():* This method is for handling the frontend of the showFeedbacks() method of the Manager class for the viewer of the video.

class MainPage
This class handles the frontend of the main page.
Methods
public void search(): Handles the frontend on the showSearchVideos method of the Manager class. Shows the searched videos by specified properties. *public void showTrendings():* Handles the frontend on the showTrendVideos method of the Manager class. Shows the trend videos by specified properties. *public void showRecommended():* Handles the frontend on the showRecommendedVideos method of the Manager class. Shows the recommended videos by specified properties. *public void signup():* Handles the frontend on the login method of the Manager class.

class UserProfile
This class takes the user's profile ID
Methods
public void takeUserProfile(): Handles the extraction of the user's profile for future uses.

class UploaderResultPage
This class handles the frontend of the feedback results to the owner (uploader) of the video.
Methods
public void showHeatMap(): Handles the frontend on the showHeatMap method of the Manager class. Shows the heatmaps of the video by specified properties. *public void showLikeGraph():* Handles the frontend on the showLikeGraph method of the Manager class. Shows the like graphs of the video by specified properties. *public void showScenes():* Handles the frontend on the showScenes method of the Manager class. Shows the most liked scenes of the video by specified properties. *public void signup():* Handles the frontend on the showFeedbacks method of the Manager class. Shows the feedbacks of the video by specified properties.

class ViewController
This class represents the controller of the general frontend
Methods
public void getURL(): This method is for getting the URL of the video the user wants the watch. *public File sendPages():* This method is responsible to process-related pages. *public void startServer():* This server is for starting the server's working

3.3 Controller

class Manager
This class is the main manager class of the system. It controls the overall system and implements the functionality between the user and the system.
Properties
public User user public CloudInterface cloud public VideoCrawler videoCrawler public ProtocolAndCheckings checkings public TrendVideos trends public RecommendedVideos recommendeds public FaceModel face public Sharing sharing
Methods
public void login(): This method implements the login system by using Google Authentication system. *public void showRecommendedVideos():* This method implements the recommended videos to users basing on their data. *public void showTrendVideos():* This method implements the trend videos basing on the trend videos around the world. *public void showSearchedVideos():* This method implements the searching algorithm by using YouTube API. *public void doChecks():* This method is checking the promotions for camera and and data usage by calling ProtocolsAndCheckings class.

public Feedback startAnalysis(): Starts the analyzation and take frames of the user and sends them to cloud for proper processing.

public void showAnalysis(int feedbackType, int analysis): Shows the average summary of every over who watched the video excluding their identity.

public void insertVideo(): This method is for inserting each new video to the database.

public void getUsersVideo(): Gets the video list published by the user.

public void getTrendVideos(): Gets the trend videos around the world.

public void getSearchedVideos(): Gets the videos found after search.

public void getRecommendedVideos(): Gets the recommended videos to user according their specific data.

public void getHeatMap(int videoID, int feedbackType): Pulls the prepared heat map of the users who watched the specific video to the manager class.

public void getLikeGraph(int videoID, int feedbackType): Pulls the prepared liking graph of the users who watched the specific video to the manager class.

public void getScenes(int videoID, int feedbackType): Pulls the prepared the scenes most liked by the user while the watching to the manager class.

public void showVideos(): This method is for showing the user's video while the watching to the user.

public void showFeedbacks(): Shows the feedbacks of the users who watched the specific video to the owner of the video.

public void showHeatMap(): Shows the prepared heat map of the users who watched the specific video.

public void showLikeGraph(): Shows the prepared liking graph of the users who watched the specific video.

public void showScenes(): This method is for showing the scenes most liked by the user while the watching as feedback.

public void finishAnalyzing(): Finishes the analyze and send data to make them a proper feedback.

public void insertFeedbacks(Feedback viewer, Feedback owner): Inserts the feedback of a video to the database with its type.

public void sharing(): Enables for user to share their analyzes by calling Sharing class.

class Analyzer
This class analyzes the video of the user and give results in real time, frame by frame.
Methods
public void analyzeResults(): This method analyzes the frames of video of the user by using OpenFace library. *public void sendAnalysis():* This method sends the result of the analysis to the interface.

class Feedback
This class creates feedback of the video of the user from the analyze that comes from Analyzer class
Properties
public string userToken public int videoID public int userType public int feedbackID public CloudInterface cloud
Methods
public void makeGraph(): Creates the graph of like percentage each second by using the analyze. *public void makeHeatMap():* Creates heat map of user's sight through the video. Contains information about where the user looked at each second. *public void recommendScene()*: By using the information of where user liked the most, the method gives a recommendation to watch that part again. *public void sendFeedbacks(Feedback viewer, Feedback owner):* Sends the feedback created by the user to the video owner anonymously.

class CloudInterface
This class builds the interface between cloud and other classes and implements cloud based methods.
Properties
public Analyzer analysis public Feedback feedback

Methods
public void sendResults(): Sends the result data of the user video to the owner anonymously. *public void sendAnalysis():* Sends the analysis of the user video to the Feedback class.

class ProtocolsAndCheckings
This class requests the necessary protocols and permissions from the user.
Properties
public bool cameraCheck public bool camPermission public bool signed
Methods
public void checkCam(): Checks if the user has a camera on their device or not. *public void getCamPermission():* Requests camera permissions from the user. *public void getSign():* Informs the user about the usage of their data and requests the users permission.

class Sharing
This class handles the sharing of the analyze of the video.
Properties
public string userToken public int videoID public int analyzeID public int platform
Methods
public void share(int platform, int videoID, int feedbackID): Shares the analyze of the user of a specific video.

class FaceModel
This class uses OpenFace library to process the face of the user.

Properties
public string userToken public int videoID public int camID public CloudInterface cloud
Methods
public void analyzeFace(): Analyzes the face of the user frame by frame by using the OpenFace library and extracts the key points. *public void sendData():* Sends the key points data to the cloud.

4. Glossary

-A-
Analysis, 4: 8: 15: 16: 17
API, 12: 14

-F-
Facial, 2: 3: 4

-L-
LikedIt, 10: 11

-O-
OpenFace, 4: 16: 17: 18

-U-
UML, 4

-Y-
Youtube, 2: 10: 12: 14

5. References

[1] **“**Multimodal Sentiment Intensity Analysis in Videos: Facial Gestures and Verbal Messages” *Sentic.net*, 2019. [Online]. Available: https://sentic.net/multimodal-sentiment-intensity-analysis-in-videos.pdf [Accessed: 5- Oct- 2019].

[2] Pitt.libguides.com. (2020). *LibGuides: Citation Styles: APA, MLA, Chicago, Turabian, IEEE: IEEE Style*. [online] Available at: https://pitt.libguides.com/citationhelp/ieee [Accessed 9 Feb. 2020].

[3] Open Face”, 2019. [Online]. Available: https://github.com/TadasBaltrusaitis/OpenFace [Accessed: 5- Oct- 2019].

[4] "MVC Framework - Introduction - Tutorialspoint", *Tutorialspoint.com*, 2020. [Online]. Available: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. [Accessed: 16- Feb- 2020].