CS 492
Senior Design Project

Final 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

TABLE OF CONTENTS

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 capture physiological reactions in real-time to accurately research how humans appreciate it. 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 system provides a feedback report to the video viewers about how much they liked or disliked the video scene by scene, where do they look at in the video mostly. The application can be used by every age, gender, and ethnic group who watches videos.

2. Requirements Details

2.1. Functional Requirements

- The user can sign up with their Google accounts.
- The program can generate a plot about the liking rates throughout the video.
- The program can generate a plot about the rate of emotion types throughout the video.
- The program can generate a heatmap of gaze throughout the video.
- The program can show trend videos globally and in Turkey.
- The program can show most recently liked/disliked videos.
- The program can show search result videos via YouTube.
- The program can show the previous analysis results.
- The user can share the link of videos and analyses the results of the videos.
- The user can choose between the local or the web option for the system.

2.2. Non-functional Requirements

2.2.1. Security

- The application should not keep the data on the user's face.
- The application should provide security of personal data of the analysis for videos. They should be private to the user.
- SSL certificate should be taken for a secure connection.

2.2.2. Usability

- Our user interface will be easy to use for anyone who can use YouTube. The design of the site should be easy to understand for them.
- The system should be available to everyone with regard to YouTube's age limitation for specific videos.
- The system should be stable in order to prevent any kind of interruptions.

2.2.3. Cost

- The system should be free of charge for all users.
- The system should be implemented at no cost.

2.2.4. Performance

- The system should analyze the face of the user in real-time.
- The average model accuracy rates should be higher than %60.
- The system load time should be lower than 5 seconds.

2.2.5. Extendibility

- The system can also be adapted to other platforms such as Netflix, Instagram, Ted Talks, and any other platform on the internet that contains videos.
- The system has comprehensive models that can be adapted for other deep learning projects.

2.2.6. Marketability

- Advertisement companies can measure how their advertisements affect end users.

- Also, companies can produce their own advertisement in light of this feedback without advertisement companies.

3. Final Architecture and Design Details

3.1. Overview

In this part, we will firstly explain subsystem decomposition, where the subsystem structure of our system is described in detail with diagrams and the classes. Then, we will provide hardware/software mapping of the system which shows the allocation of resources in our project. Then we will explain the persistent data management.

3.2. Subsystem decomposition

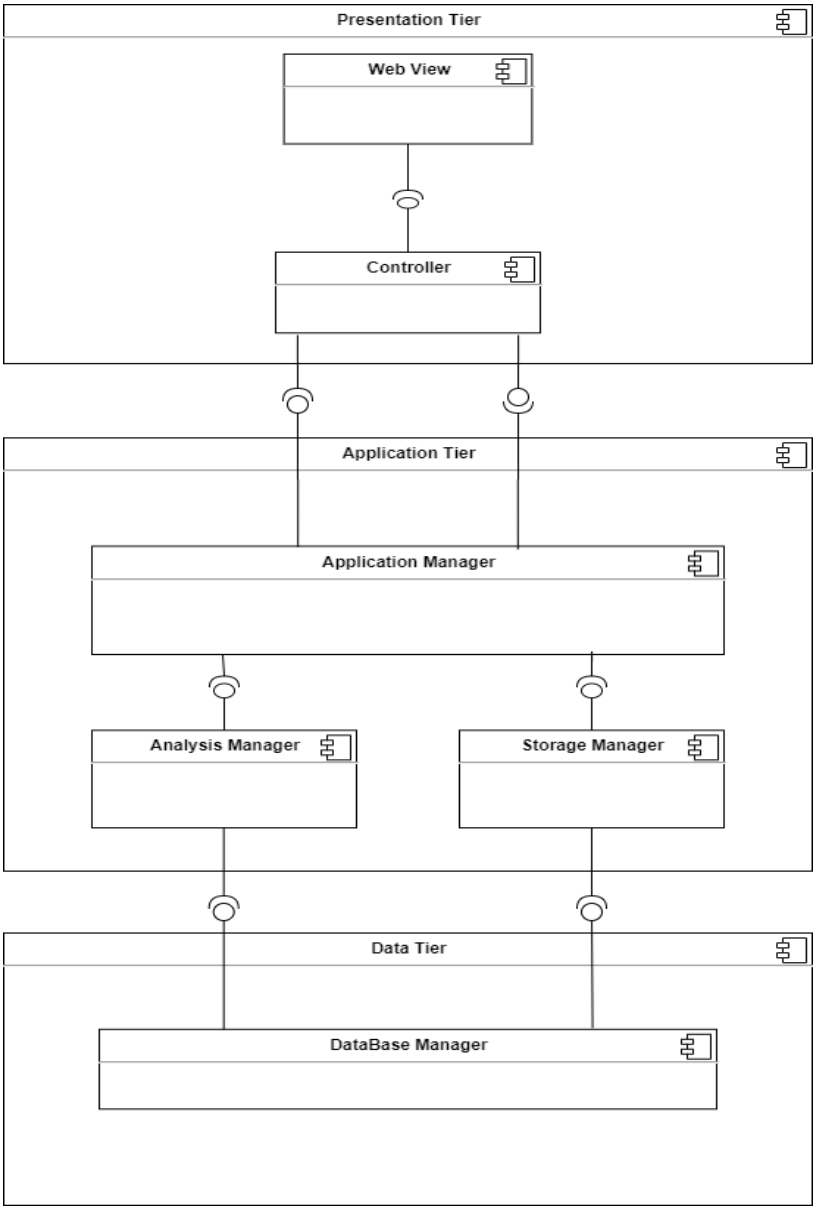


Figure 1: Decomposition Diagram

Our system is formed with a 3 Tier Decomposition. These tiers are presentation tier, application tier, and data tier. This tier decomposition gives us the flexibility to update project since tiers are independent of each other

The presentation tier consists of user interfaces. Web view, controller, and model packages are in this tier where they have different classes under each of them. You can see details of the web view, controller, and application manager from the below diagrams:

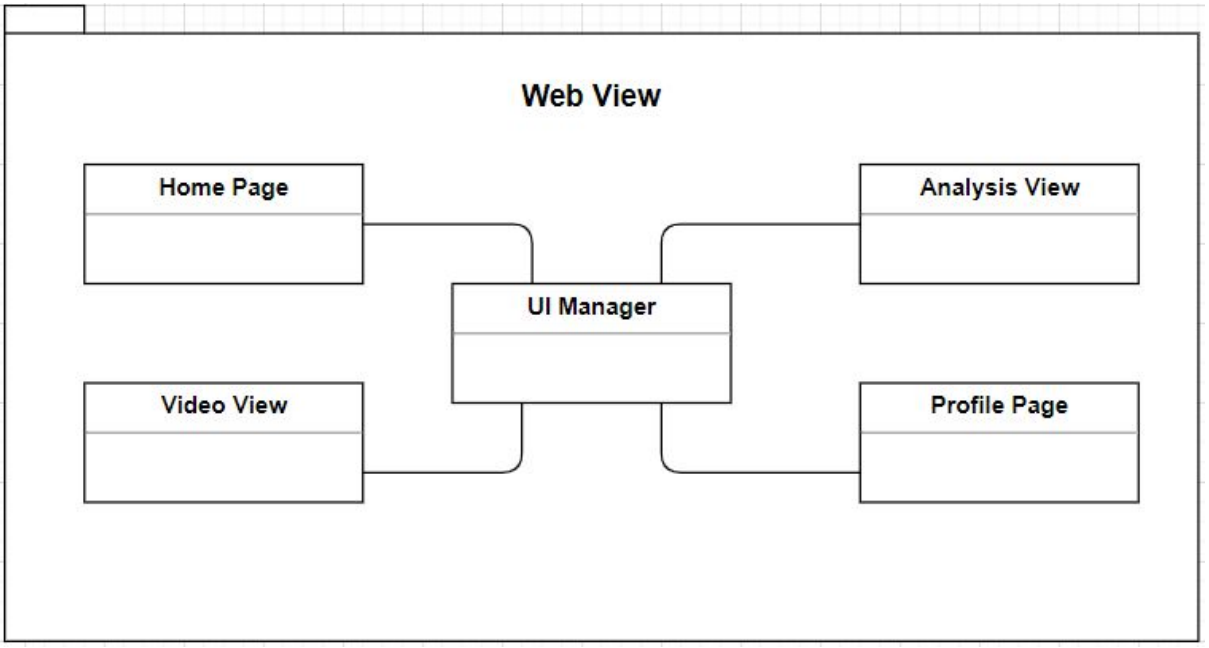


Figure 2: Web View

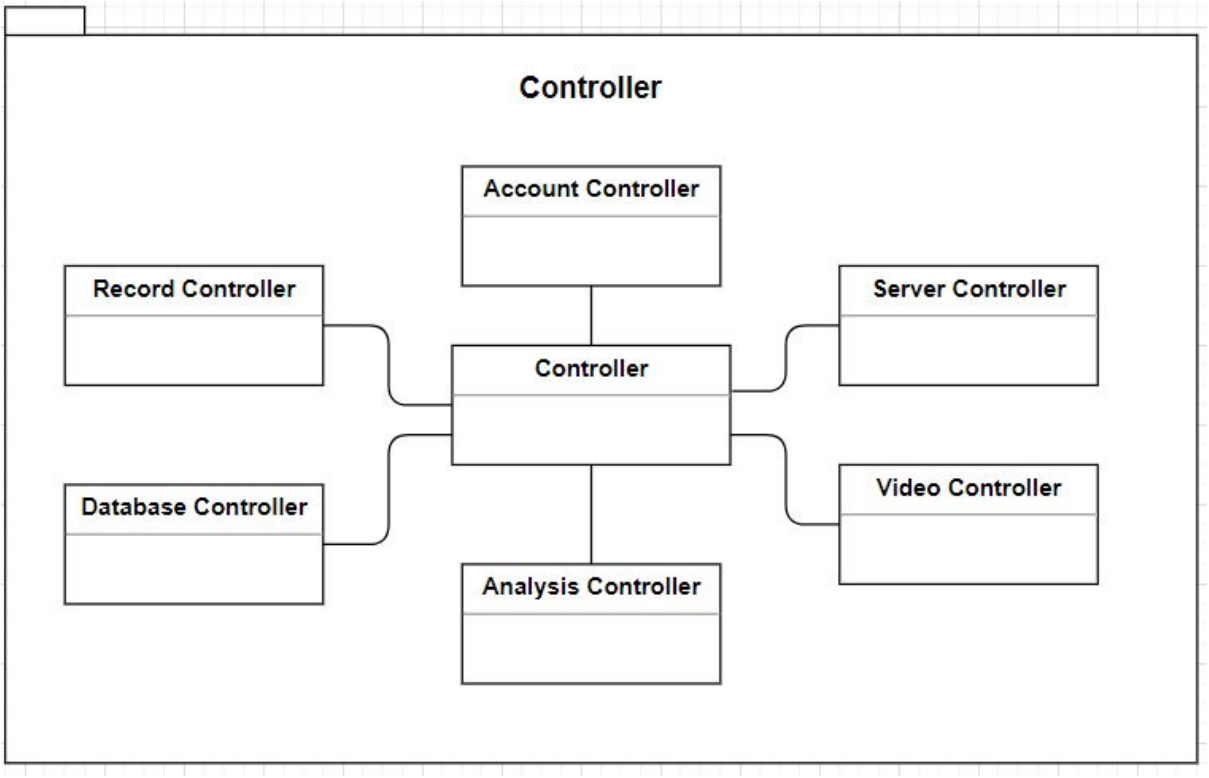


Figure 3: Controller

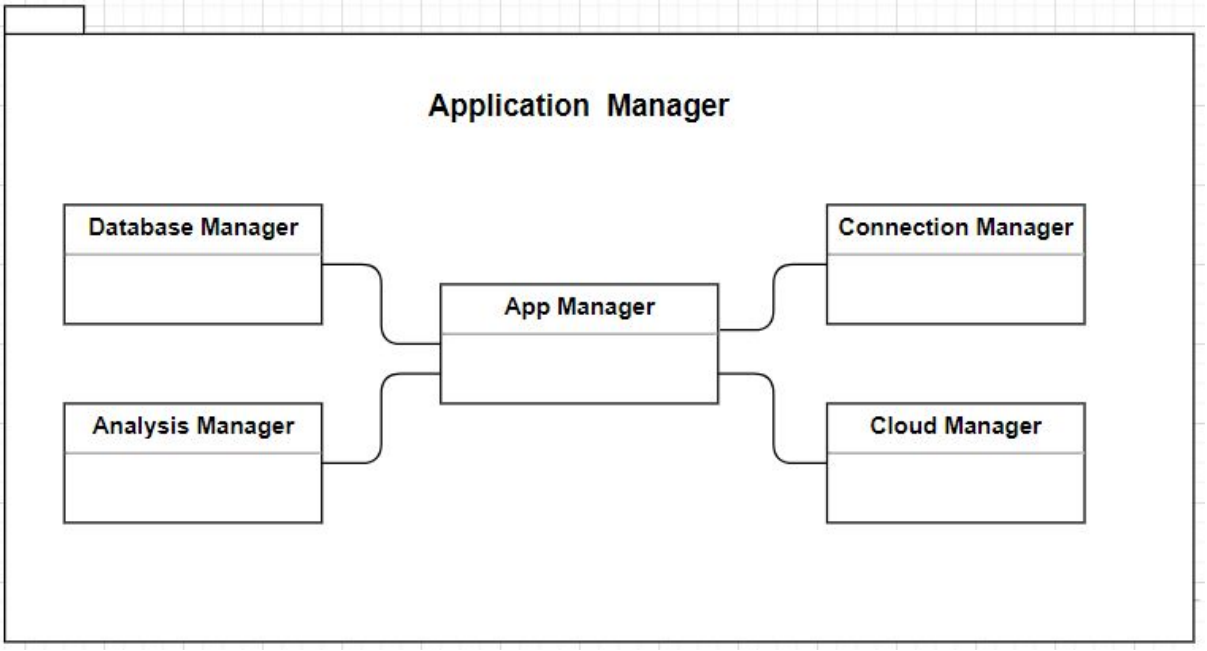


Figure 4: Application Manager

3.3. Hardware/Software Mapping

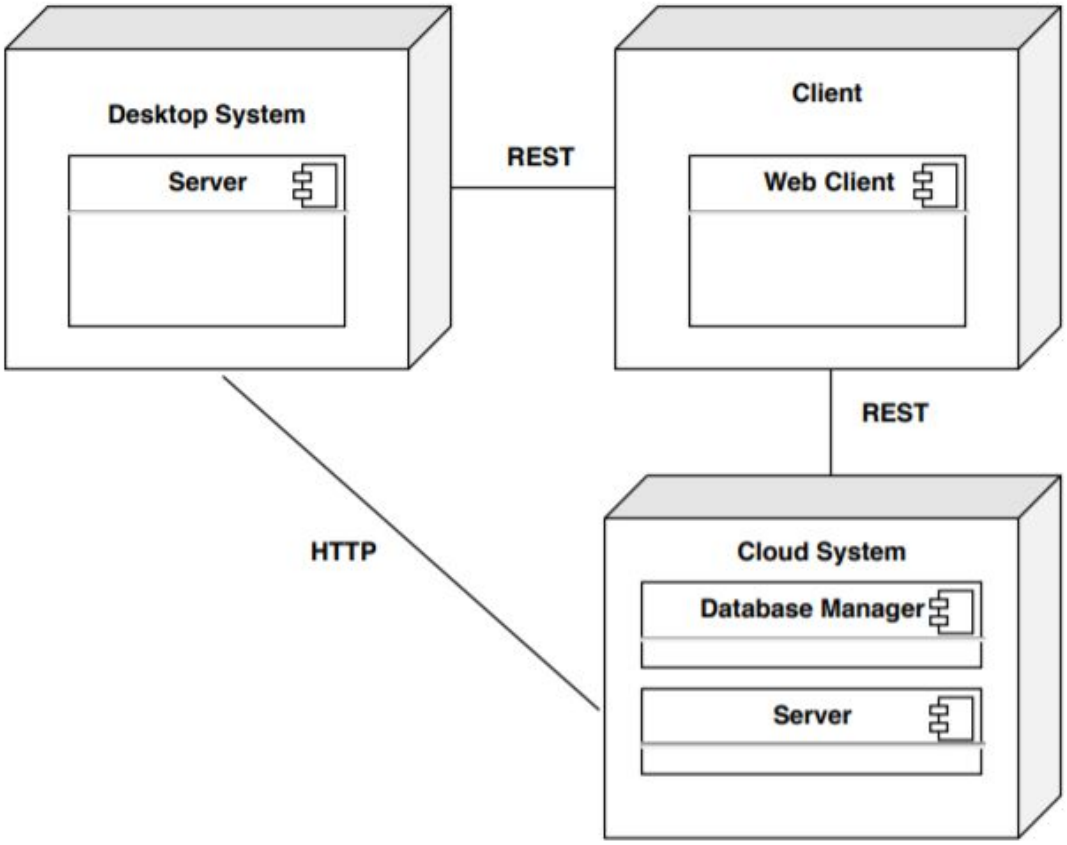


Figure 5: Mapping

The tiers of the system will be divided into desktop systems, clients, and cloud systems. The system has one client, which is a web client and two servers which are run on desktop and cloud. The client will connect to the servers with a Rest API. The web client will be responsible for the presentation layer of the system. The desktop server software will be in the same machine with the client and will be responsible for running necessary models on the background if necessary. The cloud tier will have 2 different components. One of them will work as a database component and stores user data and application data. The second part will be responsible for running the necessary models in the background if necessary.

3.4. Persistent Data Management

For LikedIt there is a big need for a database. We should keep the information of our users and their results of analyzes. Also, we should have the details of the videos watched from our website. So, to provide all of these we create a database in our Google cloud account. Because we all familiar with MySql thanks to our

education in Bilkent, we chose this language for our database. After making connections between the system and the database, we created our tables.

4. Development/Implementation Details

4.1. Frontend

About the main page: On the main page of the site, we show the user, 6 trend videos globally, and 6 trend videos of turkey of that time. When the user signs in via the sign-in button on the page, users can also see 6 videos they liked and disliked most recently. There is also a search bar which when the user searches something, 12 best results show up. There is also a slider that gives minimal information about privacy and how to use it. There is a sidebar at almost every page including the main one that has the list of pages that users can redirect to. Users can toggle the sidebar by clicking on the menu again.

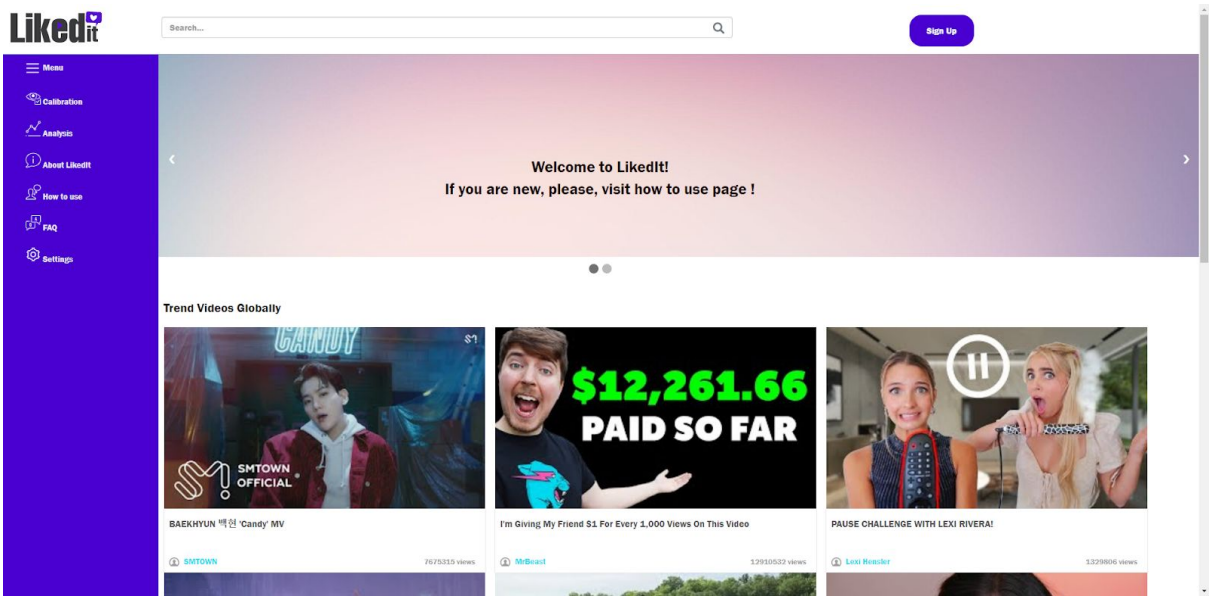


Figure 6: Main page before sign in

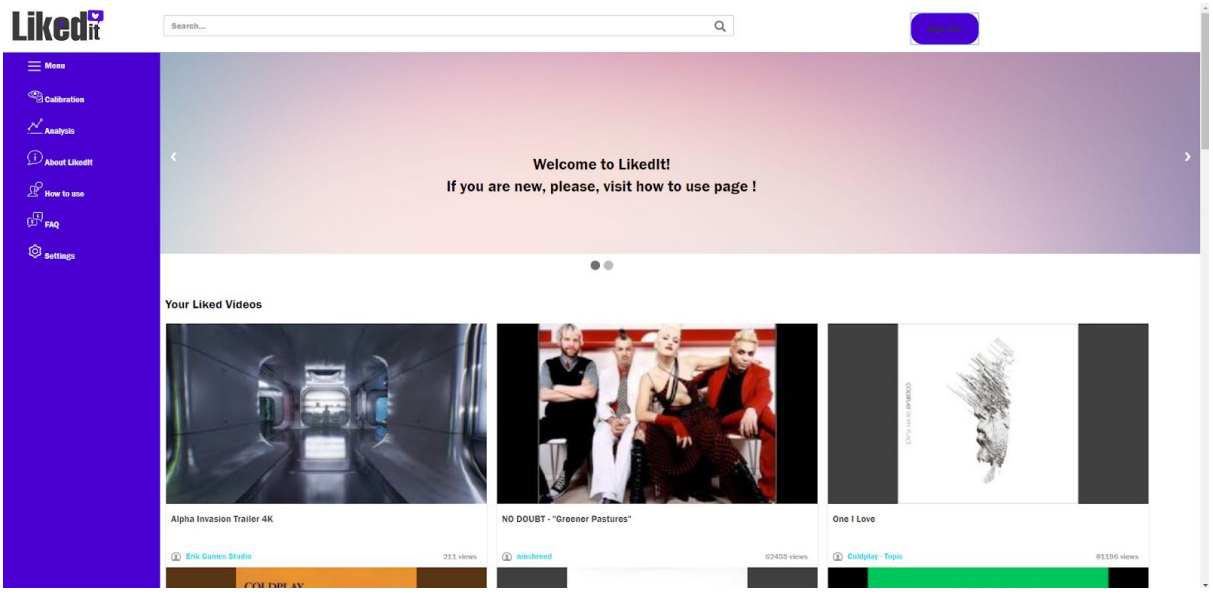


Figure 7: Main page after sign in

About calibration: For the purpose of collecting the gaze data to generate the heatmap, we created a page called webgaze. At this page, we draw multiple cyan dots for the user to click in order to properly adjust the gaze data. Colors of the clicked dots change in order to let the user know the remaining dots to click. Users see their face and the face detector on top of their face and a text that informs the user of how to properly adjust the calibration. After starting to click on dots, users see a new circle that tracks their gaze and shows the location that users are looking at the moment. The more they click the dots, the more accurate this new circle will point to the location they are looking for. After clicking all of the dots, the user presses the “Done” button to return to the homepage.

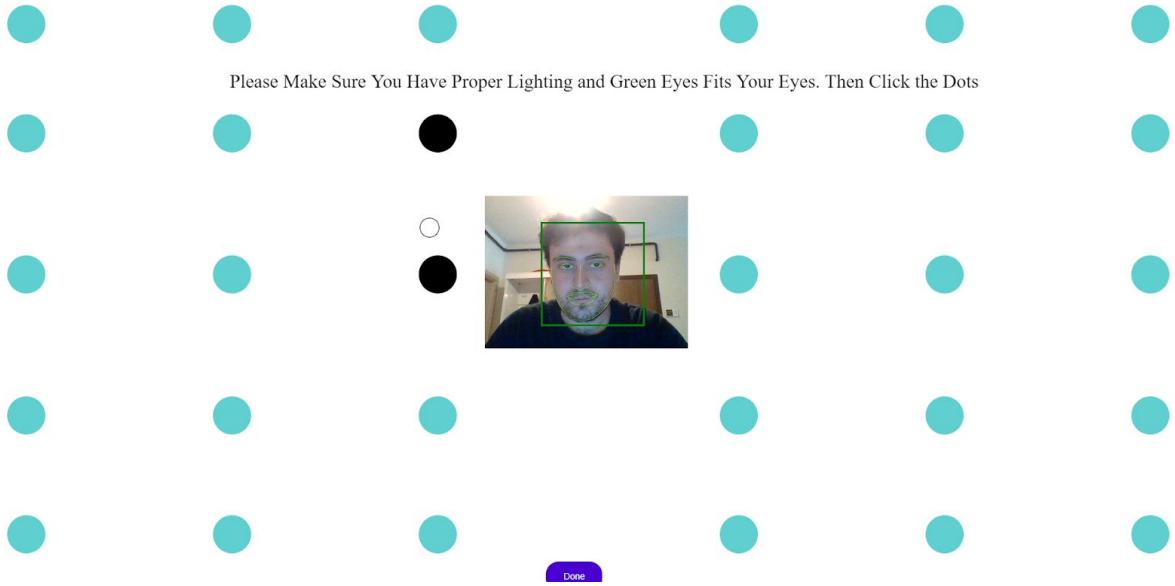


Figure 8: Calibration page

About player: When the user clicks at a video, they are redirected to the player page that has the video and the video information. Users first see their webcam videos to properly adjust their faces if it's lost. After that, users can click the "toggle face" button to hide their faces, and see again if they like.

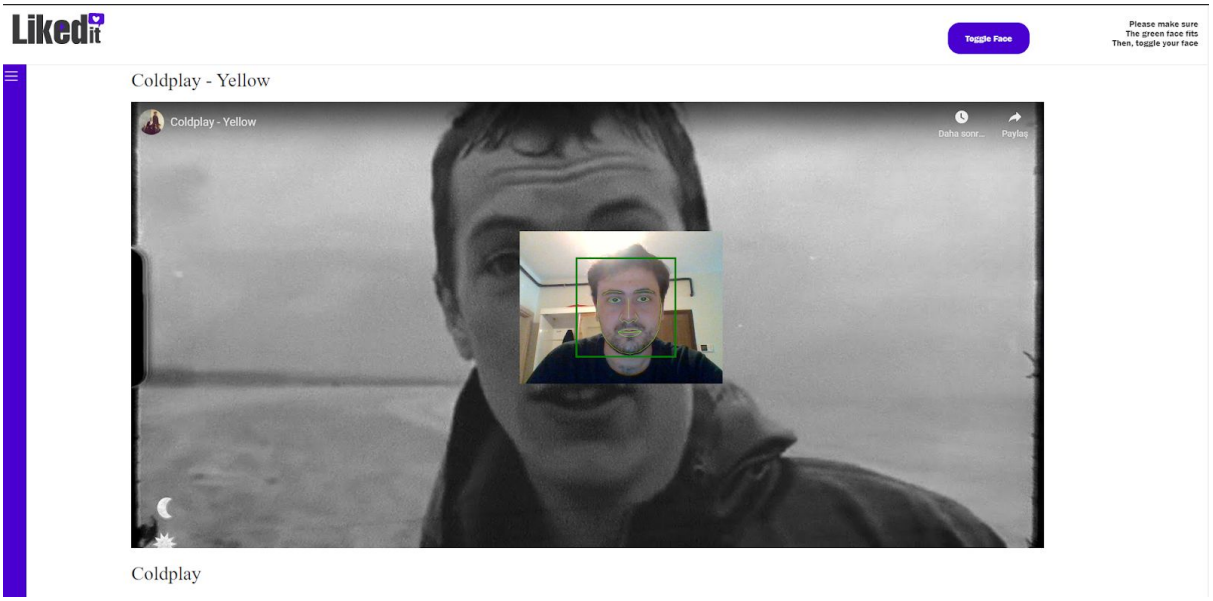


Figure 9: Player

About analyses: If the user signs up to LikedIt and watch at least a video from the website, he/she can see their analyses list for each watching with the dates that they watched the videos in the analyses page.

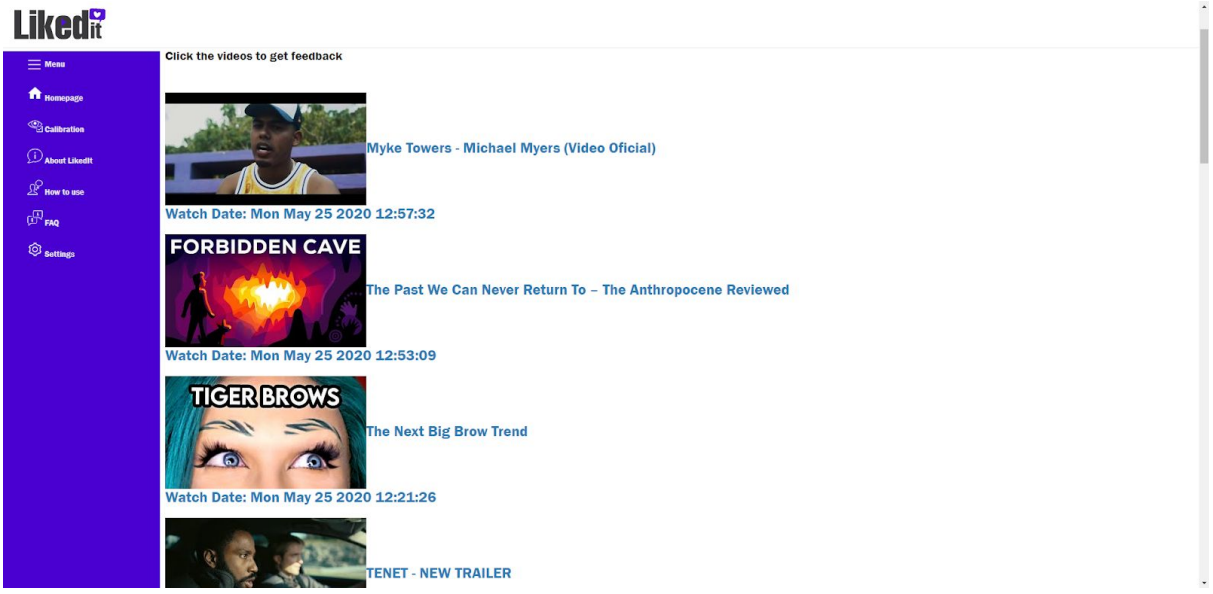


Figure 10: Analysis

About feedback: When clicked on an analysis on the analyses page, users are redirected to the feedback page of that specific analysis. There, they can see their heatmap of gaze and two plots about their liking scores and emotions scores. The

user can choose the type of analysis that he/she wants to see. This feedbacks are created by considering the time slots when the user watches. If the user fasts forwards the video during the watching, then any analysis is not created for the unwatched parts of the video. Users can click on a point at the plot to seek that second at the video and watch it from there. There are also buttons of pause/resume to help the user to navigate.

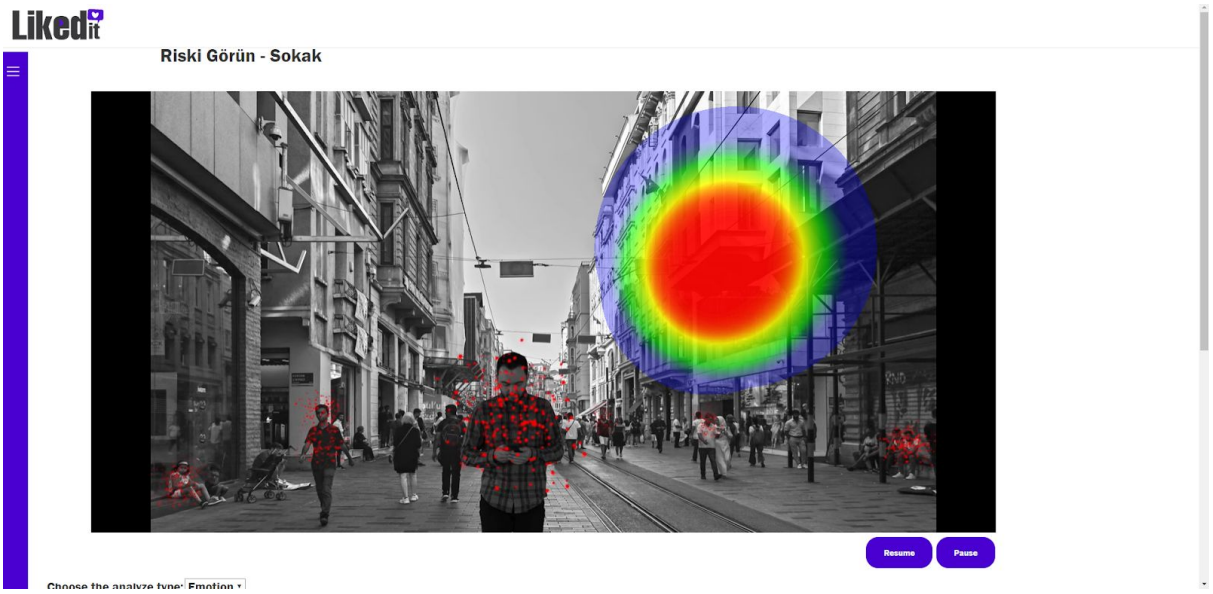


Figure 11: Feedback page, heatmap while the video is playing

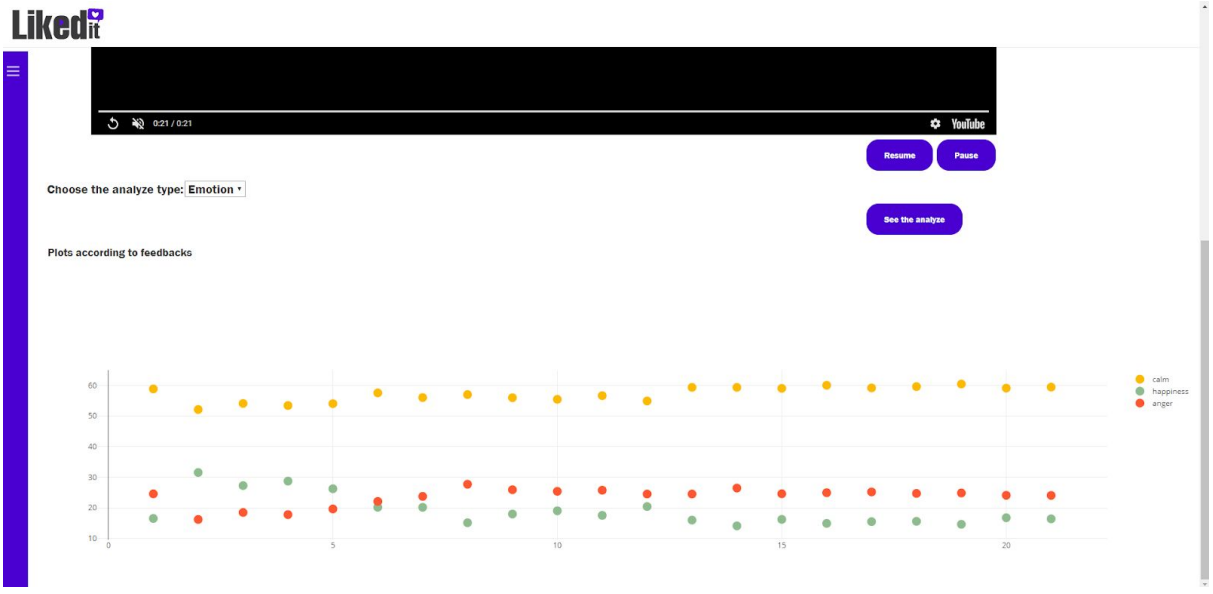


Figure 12: Feedback page, emotion plot

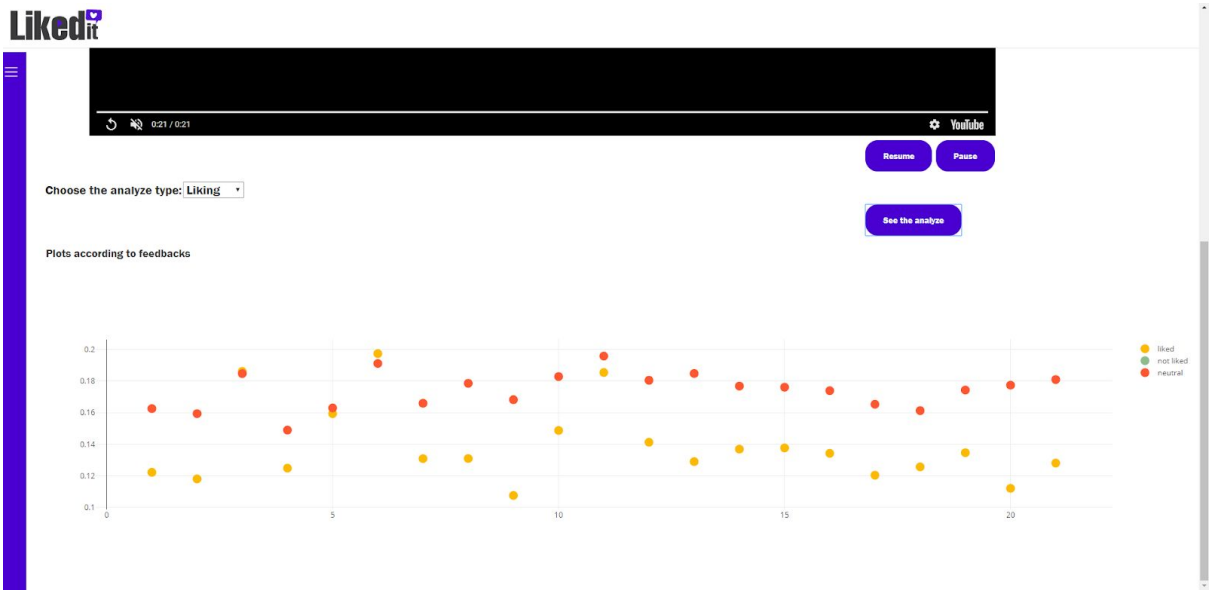


Figure 13: Feedback page, liking plot

About how to use: How to use page includes information about how to use the system and its features properly. There are texts which are supported by images to help to clarify the way that site works.

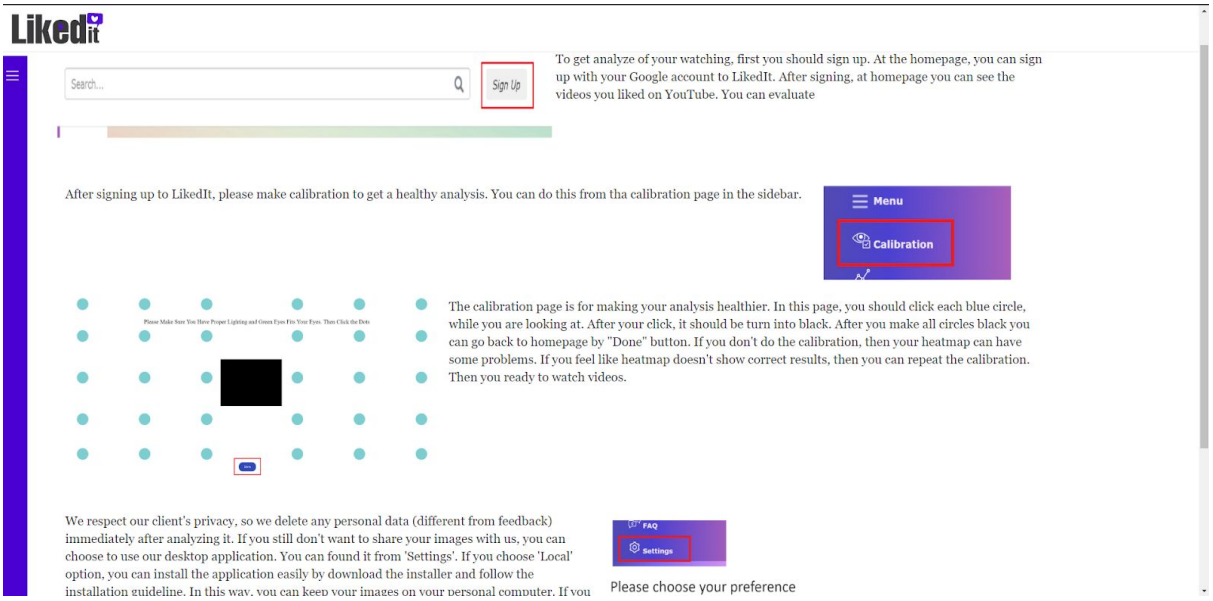


Figure 14: How to use page

About settings: Users can select the version of LikedIt that they want to use from the settings page. If they select the local version, the desktop version of LikedIt will be downloaded.

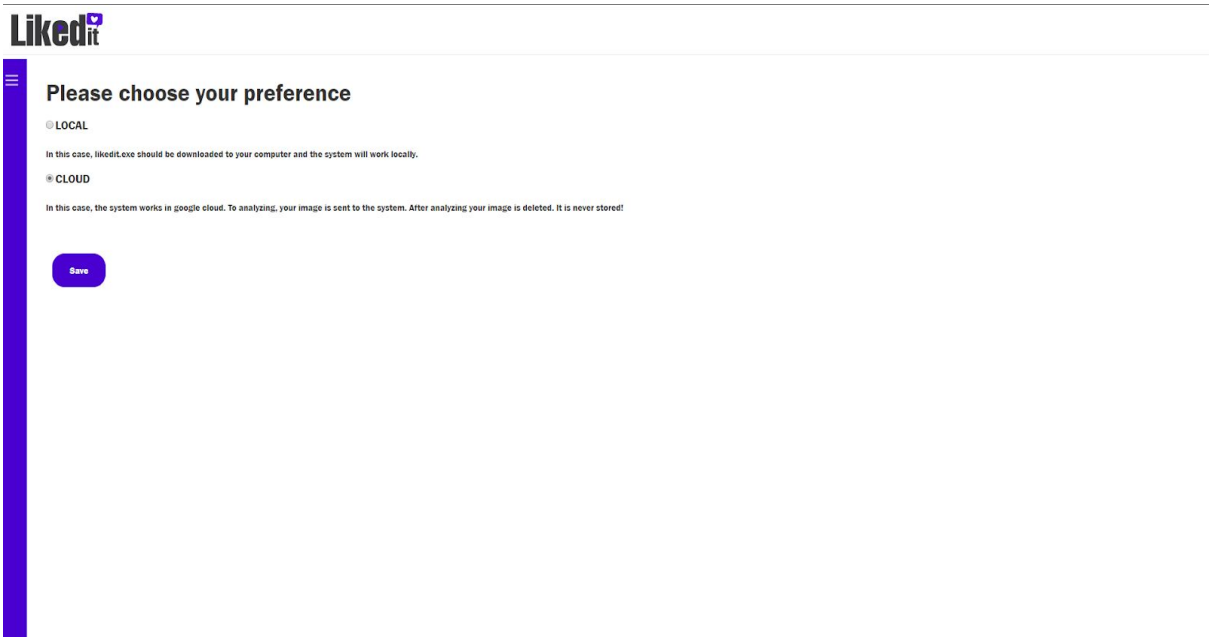


Figure 15: Settings page

4.2. Backend

For serving frontend content, we use Nginx which is software that provides a web server for our application. NGINX is a free, open-source, high-performance HTTP server and reverses proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.[2] We integrate it with a cloud compute engine instance. To provide a secure connection, we took an SSL certificate for our web page. SSL certificates are small data files that bind a cryptographic key to an organization's details and it activates the lock and the https protocol and allows secure connections from a web server to a browser.[3] Just before to start backend implementation, we adjusted these settings at our cloud.

For backend, we use Node.js. Node.js is an open-source, cross-platform, JavaScript runtime environment. It executes JavaScript code outside of a browser.[4] The reason that we choose Node.js as a backend is, it is easy to implement Rest API on it and it provides fast execution as a server. We use the Express module of Node.js to implement Rest structure. Node.js instance runs on the cloud compute engine instance and communicates with the frontend through Rest API. We also implement a desktop application with Node.js. It runs on the user's machine and communicates both with the frontend and server that run on the cloud.

Our backend does several jobs. Firstly, it gets image info(blob) for each frame from the frontend and saves them as images. Once 100 images saved for a video, it sends these images to the OpenFace[5] queue and emotion model queue. The

reason behind we run 100 images together but not each frame individually is it run each frame individually decreases the speed of the server. Since Node.js run asynchronous, we implement a system with queues in the parts that we want to run server synchronously. For example, we need to first analyze each frame with OpenFace, then we need to call our model to analyze them and to save the result to the database. Each part depends on the output of the other part and has to wait for each other. Emotion model queue analyzes images to get emotion out of them and then call the database queue to save results to the database. To run our emotion model, we have to build a python virtual environment. We did setups for python virtual environment and other dependencies about the emotion model. OpenFace queue run images in OpenFace queue and call both pre model and database queue. The database queue saves the result of OpenFace to the database. Pre model queue prepares images for the model. After pre model queue is done, it calls the model queue. It where we analyze our images for liking rates. When it's done, it calls again the database queue to save results to the database.

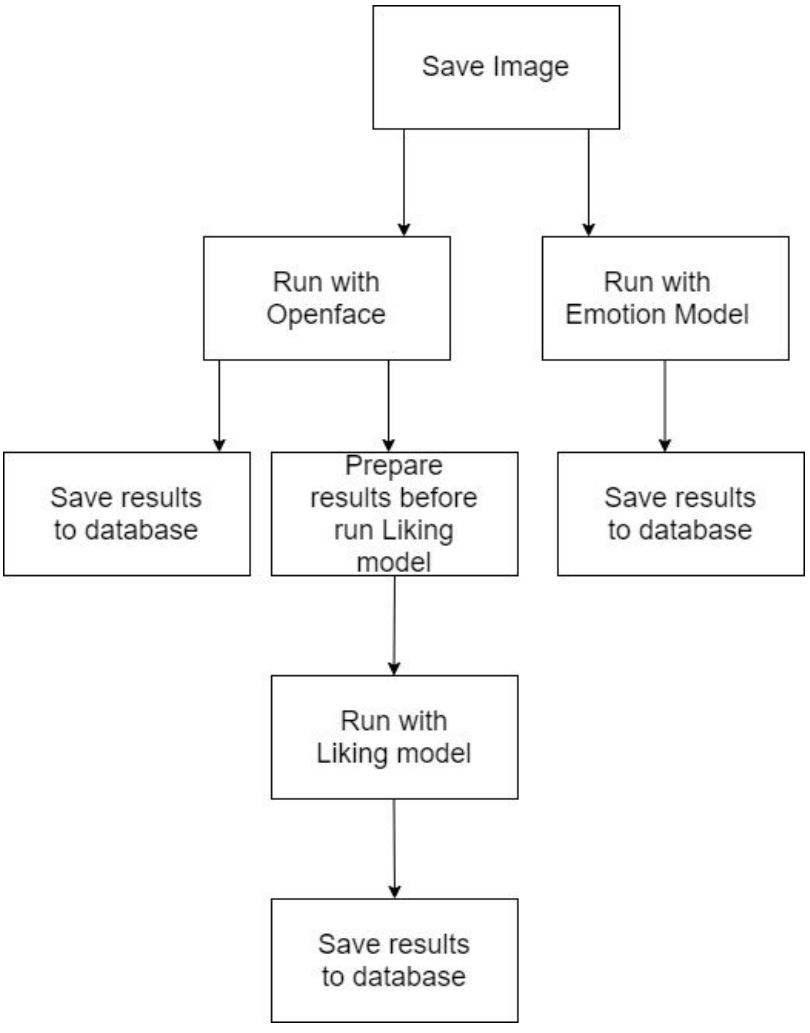


Figure 16: Architecture of the backend

Therefore, some parts of the backend run asynchronous and some parts of it run synchronously. The advantage of this kind of structure is we done necessary jobs

in order and at the same time run independent parts asynchronous and save some time to get final results and decrease latency.

If the user chooses to use a desktop application, our frontend sends image data to node.js instance run on the user's computer. Similar to the logic of the backend run on a cloud instance, this Node.js instance run images on OpenFace when the number of images reached 100. Then it sends the result of OpenFace to the instance runs on the cloud. Then the cloud continues its job as saving these results on the database and so on. For users choose to use desktop application, we don’t provide emotion analysis currently but implement this feature later. Also, the model that runs for these images different from other models since we don’t save images of users but just use OpenFace results.

In both case, when video end or user stops watch the video and go to another page, the frontend sen request to backend servers and both instances analyze last frames as same logic. When all frames run on the model, the frontend can show analysis on the website by access to info from the database.

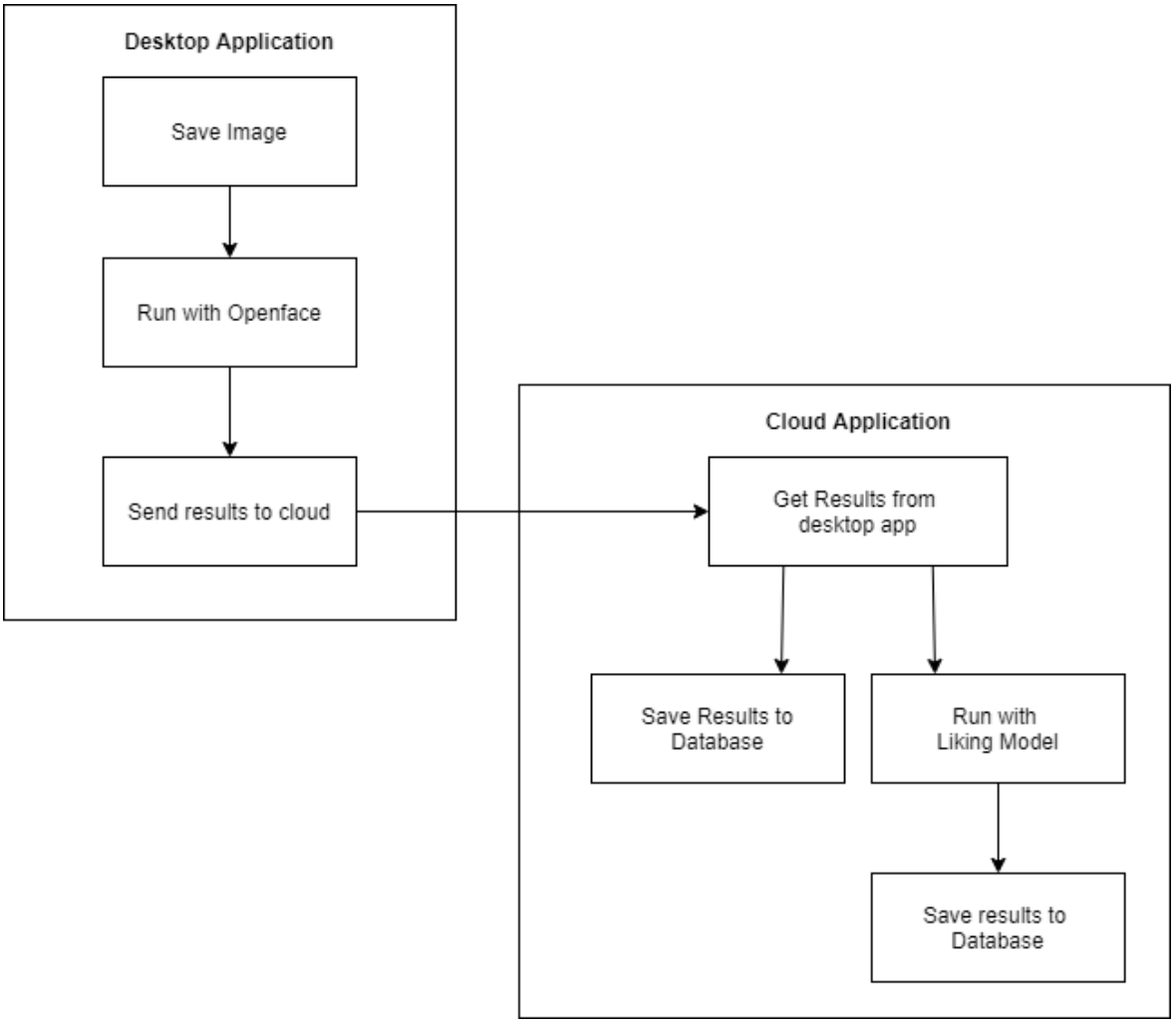


Figure 17: Architecture of the backend

For collecting the gaze data, we use webgazer.js. [6] At the calibration page, we make the user click the dots while the webgazer module automatically calibrates itself. After it properly calibrated, at the player page, we save the position of where the user is looking at while the user watches the video, every 200 milliseconds by using the functions of webgazer.js. After the video ends, or before the user tries to exit the player page without finishing the video, we store the collected gaze data in the database.

After the user clicks on analysis, we get the gaze data that belongs to that specific video. After getting the gaze data, we use canvas on top of the video player to paint the locations of where the user gazed at, at that second. In order to do that, we also use the heatmap.js module. [7]

We collect the data on which parts of the video did the user watched, in case of fast-forwarding or exiting in the middle of the video, etc. By using this data, we exclude the parts where the user didn't watch from the heatmap of gaze and the plots of models since there will be no data coming from these parts.

We use Google's Javascript Youtube API to get the trend videos, liked/disliked videos, searched videos, the clicked video, etc. [8] By using the Youtube Player element of this API, we show the video at the player page and the feedback page. By using the event handlers, we are navigating or modifying the users' data when the video ends, stops, resumes, starts, etc. This API has 10.000 requests quota per day so that the system can only handle approximately 2000 users per day, which we thought to be enough for demo purposes.

In order to handle the sign in and sign out purposes, we use Google Sign-In API and Google OAuth 2.0, so that the user will know that they will be safe. [9] Furthermore, we only request a "read-only" version for the YouTube information of the user to get the liked/disliked videos from YouTube.

4.3. Database

4.3.1. User Table

For the signing into the website, the user can use his/her own Google account. So we don't need to keep his password. However, to track the user's actions on the LikedIt, we get the user id, the username, and the e-mail address from his/her Google account by using the Google API.

UserToken (varchar(255))	**Name** (varchar(255))	**Surname** (varchar(255))	**E_mail** (varchar(255))

PRIMARY KEY			

Table 1: User table

4.3.2. Videos Table

The videos on our website are pulled from YouTube with YouTube API. In this situation, we should keep the information on the videos. At the homepage, if the user doesn't sign up, there are 2 sections for videos. One for trend videos in Turkey and one for trend videos in the world. If the user signs up, then there is another section too. this section is for the videos that liked by the user before on YouTube. We don't add all of these videos to our database to avoiding overloading. If we make it too big unnecessarily, then our systems' performance decreases. So, we add just the videos that are watched. When the user clicks a video, the code gets its information such as video id or name of the channel of the video with YouTube API and saves them to the Videos table. Also, there are other columns filled by our system, such as the column named LikedItScore which is the score that videos get from our system. This column is written according to the average of the scores for each watching of this video. Another column WatchedNumber is for the number of watchings from our website of a specific video. The last column is the duration. This column keeps the length of a video as seconds.

VideoID (varchar(40)) PRIMARY KEY	**ChannelName** (varchar(30))	**LikedItScore** (double)	**WatchedNumbe r** (int)	**Duration** (int)

Table 2: Videos table

4.3.3. User_Watch_Videos Table

User_Watch_Videos table is the most important table in our system. Basically, this table is for keeping the watching information. First of all, we give an id for each watch. Even the same user watches the same video, the system gives this watching a different id because each watching's scores can be different. In this table we keep the watch id and the information about this watching like the id of the video watched or the token of the user who watches. If the user watches the video without signing in then this column gets null. Also we keep the liking score of this watching. This score is the average of liking results of each frame. And in the time column we keep the time slots that the user watches. For example, if the user fasts forwards, we specify it in the database. So, we give our feedback considering this. We also save the date of the watching. The last 4 column of this table is for tracking the other tables. In these columns we keep the name of the tables that we write the analyze results. These result tables are created in the code in real-time. We determine the names according to the watch id. The type of the result is added to the end of the watch id and then the table is created with this name.

WatchID (int) PRIMARY KEY	**UserToken** (varchar (40)) FOREIGN KEY (User Table)	**VideoId** (varchar (40)) FOREIGN KEY (Videos Table)	**LikedItScore** (double)	**Time** (int)	**watchDate** (datetime)	**gazeTable Name** (varchar (40))	**OpenFace TableName** (varchar (40))	**model Results Name** (varchar (40))	**emotion Results Name** (varchar (40))

Table 3: User_Watch_Videos Table

4.3.3.1. Gaze Table : “WatchId” + Gaze

These gaze tables are created to keep the gaze results. In the system, the gaze model detects the locations where the user look while he/she watching the video. This detection is done constantly during the watching. And all the results are sent to the gaze table of the current watching. In this table we keep the X and Y coordinates of the detected locations. In the analysis part these coordinates are using for the heatmap.

Xcoordinate (text)	**Ycoordinate** (text)

Table 4: Gaze table

4.3.3.2. OpenFace Table : “WatchId” + OF

Open face tables are created for keeping the results come from our OpenFace model. During the video, we get frames of the user and insert them into the model. This model gives us much information about the frame like the coordinates of the mouth or eyes. To use this information easily, we split them according to their titles, and a column is created for each title. From the model results we get 23 titles, so we have 23 columns in that table. Also, we have 4 more columns too. The first one, ‘tag’, is to understand that the result is coming from the local open face model or the cloud model. The other one is ‘ImageSeq’ is to get the image which the result belongs to. Also we have a ‘timeStamp’ column to understand the order of the results. The last one, ‘resultNo’, is for keeping the frame number of the results. We don’t put the open face table into the report. As it is mentioned above, it is a table with 27 columns. It would decrease the readability of the report.

4.3.3.3. Model Table: “WatchId” + Model

These model tables are for the results of our own model. This model is created to detect the user like the video or not. Again this model works on many frames too. So for avoiding any complication we keep ImageSeq and timestamp too. Also we have columns for the results of the model. This model gives us the scores of the 3 cases: liked, not liked and neutral. We keep these three scores on our table. This model table is used for drawing a liking graph in the analysis part.

ImageSeq (int)	**timeStamp** (varchar(55))	**Neutral** (double (40,2))	**Liked** (double (40,2))	**NotLiked** (double(40,2))

Table 5: Model table

4.3.3.4. Emotion Table: “WatchId” + Emotion

These tables are created for keeping the results of the emotion model. Each frame got during the watching is also sending to the emotion model too. This model is for detecting the emotion of the user for each second of the video. It is not about liking or not liking a video. This model is to detect the rate of the 6 emotions during the watching. These emotions are anger, happiness, calmness, fear, surprise, and disgust. This model sends to the system these 6 rates and also the dominant emotion for each frame. This emotion table is used for drawing an emotion graph in the analysis part.

ImageSeq (int)	time Stamp (varchar (55))	Anger (double (40,2))	Happiness (double (40,2))	Fear (double (40,2))	Surprise (double (40,2))	Disgust (double (40,2))

Table 6: Emotion Table

To add all of these model results properly and in order, we put them in queues. In this way the results can be added into the right tables and without any interruption.

4.4. Model

4.4.1 Emotion Model

For the emotion model, at first phrase, we benefit from EmoPy which is a python toolkit with deep neural net classes that aims to make accurate predictions of emotions given images of people's faces is open source[10]. This toolkit benefits from Facial Expression Recognition (FER). As neural network architecture, there are combinations of layers that feed outputs to each other in sequence. Initial architectures based on approaches general research of FER implementation. FER has its own datasets, algorithms, and architectures for facial expressions. Its algorithm uses machine learning module Multiclass Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Convolutional Long Short Term Memory (ConvLSTM) [11]. The algorithm of EmoPy uses different neural network architecture using the Keras framework with a Tensorflow backend, allows users to experiment and see which one performs best for their needs. On the other hand, with the experiment, we discover that this model

is a little bit problematic about some emotions. Below, there is an accuracy rate of the model.

Neural Net Model	7 emotions		3 emotions	
	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
ConvolutionalLstmNN	0.6187	0.4751	0.9148	0.6267
TransferLearningNN	0.5358	0.2933	0.7393	0.4840

Table 7: Accuracy rates of emotion model

To have more accurate results, we investigated the train and test methods of EmoPy. As a dataset, they trained their models with 85% of its images and validated with the remaining 15% as well as with the Cohn-Kanade dataset during testing. [12] EmoPy provides determined emotion sets but each set has a different accuracy rate, inside. There are also some design choices for models like ConvolutionalLstmNN, TransferLearningNN, and ConvolutionalNN. ConvolutionalLstmNN is the convolutional long short term memory neural net that is a convolutional and recurrent neural network hybrid. TransferLearningNN which is pre-trained deep neural net models are used as starting points the pre-trained models it uses are trained on images to classify objects. ConvolutionalNN is a 2D Convolutional Neural Network that implements dropout, batch normalization, and L2 regularization. It is currently performing with a training accuracy of 0.7045 and a validation accuracy of 0.6536 when classifying 7 emotions [10].

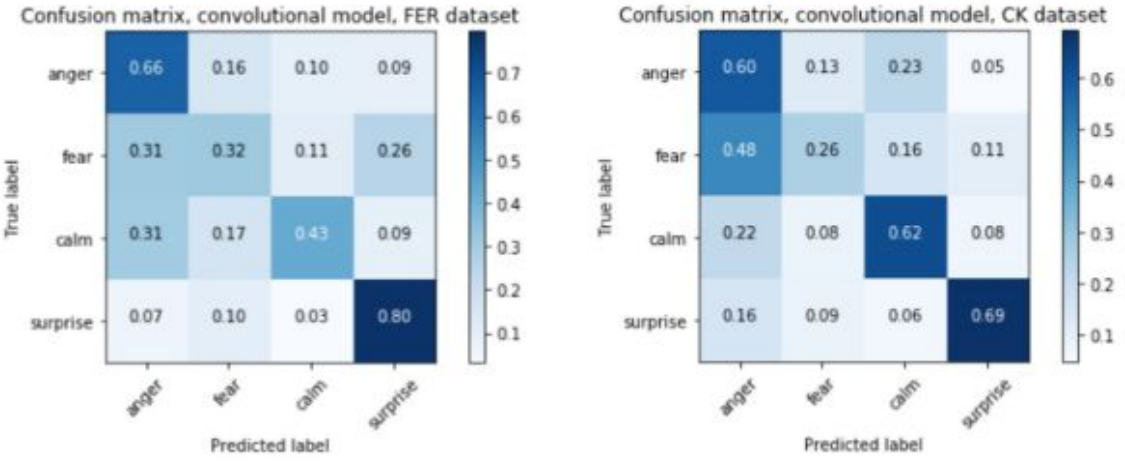
Although creators of EmoPy suggest ConvolutionalNN for better results, we realize that it differs according to the dataset. Their dataset is limited and it causes some failures. For example, when we run whole emotions at the same time, EmoPy fails and the result is wrong.

	TransferLearningNN		ConvolutionalNN		ConvolutionalLstmNN	
Emotion Set	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
Anger, fear, surprise, calm	0.6636	0.4947	0.6064	0.5637	0.6451	0.5125
Disgust, happiness, surprise	0.7797	0.7877	0.9246	0.9045	0.7391	0.7074
Anger, happiness, calm	0.5385	0.5148	0.7575	0.7218	0.7056	0.6653
Anger, fear, surprise	0.771	0.5914	0.6851	0.6503	0.5501	0.3523
Anger, disgust	0.9182	0.9094	0.958	0.9404	0.8971	0.9118
Anger, fear	0.6691	0.6381	0.7791	0.7029	0.5609	0.5567
Disgust, surprise	0.9256	0.9019	0.9893	0.9624	0.8846	0.8806

Model performance test results

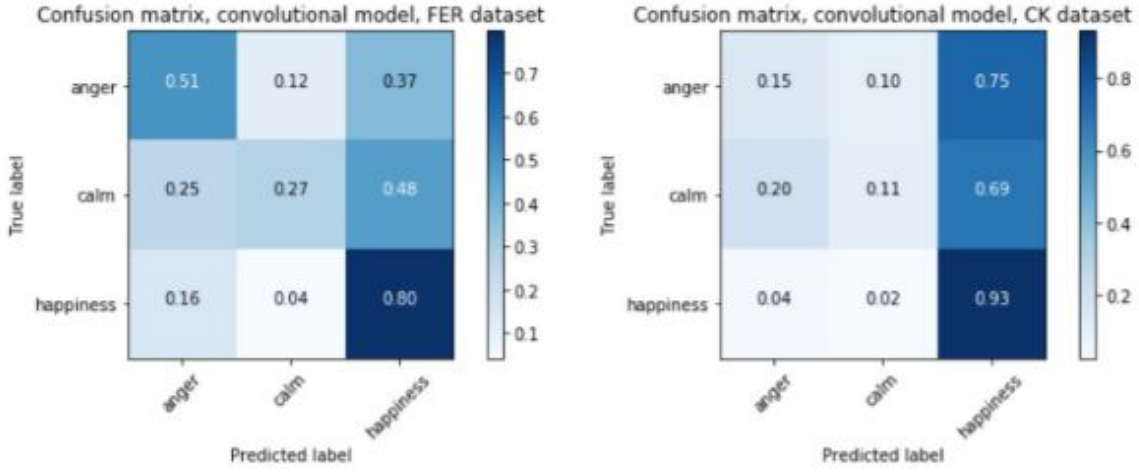
Table 8: EmoPy model performance table for emotion sets

As we know, facial expressions are by their nature ambiguous, and certain distinct emotions can lead to very similar facial expressions. In the datasets of EmoPy, we observe anger and calm subjects are a little bit similar. Therefore, results are similar, too. The most effective solution is to extend the dataset. The larger dataset can solve this type of confusion, easily. They are also aware of these misclassifications and they share with confusion matrix which is a specific table about the visualization of the performance of the machine learning algorithm. Below, confusion matrices represent the basic emotions about our feedbacks.



Confusion matrices for anger, fear, calm and surprise

Figure 18: Confusion matrices for anger, fear, calm and surprise for EmoPy



Confusion matrices for anger, calm and happiness

Figure 19: Confusion matrices for anger, calm and happiness for EmoPy

On the other hand, we can't interfere with their model so we try to solve the problem with different approaches. Then, we try to improve the model with new implementation but we use the main code of EmoPy, too. Our algorithm chooses which neural network algorithm provides the best accuracy for the related emotion set and main code allows these types of experiments. We did some modifications inside code and we fixed some problematic sides in the light of learnings from our liking model. Also, to take better results, we will run emotion sets more than one and change emotion sets because limitations within emotion sets cause ambiguity, too. With the experiment, we try to figure out best practices. Although we still try to improve our model accuracy result, anger and happiness emotions have a great accuracy rate, intermediate emotions may confuse a little bit. Our plan to provide

happiness, surprise, fear, disgust and anger emotion feedbacks. On the other hand, with our additions on EmoPy, it has better results from the beginning. We will share our findings with EmoPy creators.

4.4.2. Dataset Collector Application

Before we implement the liking model, we decide to collect our data for the model. In addition we want to automate this part so we create a data collector application. This application is written with Java. The main feature of this application is to show several short videos to the user and record them as there are watching the video to capture their facial expressions. At the beginning of the app, we show our terms of use and ask users if they agree with it or not. We also demonstrate the honor code to indicate that we do not share any personal information of users collected by this application without knowledge of the user. The information collected with this app only uses for academic purposes. Then, we make a personality analysis test. Then we make calibration to the user in order to track which side user looks. Then we show 5 short videos to the user. At the end of each video, we ask them to rate video between 0-10 points. We collect this information in our private dropbox account. We use these data to train our liking model.

4.4.3. Liking Model

Liking model is a model that guess from a frame if the person is liked the video, neutral, or doesn't like it at all. To implement this model, we used two types of datasets. One of them is the Facial Expression Data (AM-FED) and the other one is the data we collected from people.

AM-FED (Affectiva-MIT Facial Expression Dataset) is the dataset collected in March 2011 by the collaboration of Affectiva and MIT. This dataset captures naturalistic and spontaneous facial responses to three advertisements. It has 242 facial videos in recorded real-world conditions.

We also use OpenFace 2.2.0: a facial behavior analysis toolkit. This toolkit is capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. In this project, we used landmark detection for normalization process, Facial Action Unit for the RNN network, and support vector machine.

4.4.3.1. RNN and Support Vector Machine

At first we only worked with Action Units(AU) of OpenFace has given to us. OpenFace gives 18 different AU units that represent the actions of faces. We put every single video to OpenFace and used the CSV results. We extract the AUs from

CSV and get each frame mean value for every AUs and their standard deviation and maximum values. This results' evaluated in RNN neural network and also Support Vector Machine(SVM) to see the results of AU units. For RNN we used a 4 layered linear sequential network that has been used. The accuracy accomplished as 57% in SVM and 52% in RNN

4.4.3.2. Normalization

To work with those data, we had to normalize the frames as only the face of the user will be seen as figure 20. To get these normalized face polygons, we used OpenFace results that give the coordinate of face's landmarks as figure 21. The cut images' ratios are calculated from an article which is named Visual Transformation Aided Contrastive Learning for Video-based Kinship Verification[13] and implemented in python 3.6environment. We calculate the middle point of two eyes. Based on the eyes, the image is rotated as two of the eyes should be parallel to the image's side. The length from the middle to the side of the image is calculated as dx and the ratio from eyes to the forehead is calculated as (2/5)dx while from the eye to the bottom is calculated as (8/5)dx. This gives us a square image by 2dx to 2dx which we resize it as 80 by 80 pixels.

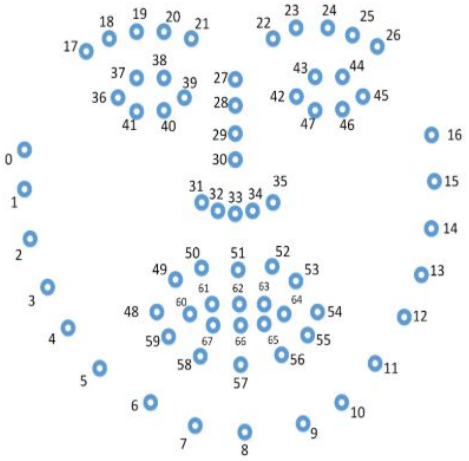


Figure 20: OpenFace result for landmarks of the face

4.4.3.3. Background for CNN implementation

In all the modelings we worked on the PyTorch library. We used different GPUs. The one locally implemented has GPU was NVIDIA GeForce GTX 950M which has 10.2 Cuda in it. To work on two different GPUs at the same time the Google Colab has been used online. It provides Tesla P100-PCIE-16GB GPU which has 10.1 Cuda in it. The epoch is a number of iteration needed to call every crucial

function to develop the model. It includes forward and backward propagation. In every epoch we activate the neural network, calculate the loss, get partial derivatives of the loss function, and update the new values. As a loss function we tried cross-entropy loss, mean square error, mean absolute error. Loss function can be broadly categorized into 2 types: Classification and Regression Loss. The classification has a discrete output on each image it sees while regression has a continuous output. Therefore, we tried both of them to see which one we need to use. Batch size is the term that is used in machine learning and deep learning which refers to the number of training examples utilized in one iteration. We tried different batch sizes like 32, 64,100,128,256, and decided to use 128. ReLU (Rectified Linear Unit) is a linear function that will output the input as positive if it is positive else as zero. Batch normalization is the function that allows each layer of the network to learn by itself a bit more independently of other layers. The pooling layer operates on image to reduce the spatial size of the representation to reduce the number of parameters. Dropout function is a technique that reduces the overfitting in the network.

4.4.3.4. Model Training in CNN

After the normalization of all images is done, the data is prepared for training. At first we worked on the AM-FED data only. Because AM-FED data was unbalanced and we didn't check for balancing, the first model that is created was a biased dataset. In the training method it receives approximately 80% liked videos, and 15% neutral videos and 5% not liked videos. Therefore, the trained dataset only returns liked labels as output to each image it sees.

The second problem we faced while modeling is we forget the fact that the data remember every person with their result. We equally balanced the data but shuffled all of them and separate the test from this set. Therefore, the network remembered every picture which gave us a 98.8% accuracy rate. [14]

4.4.3.5. 11 layered Regression CNN network without dropout function

After the error that we faced, we separate the test data by selecting as not a person will be in the train and test at the same time. Although the training accuracy is 98% in 50 epochs and batch size 32, the test accuracy was 22% which is highly low. We tried different batch sizes to find a better result like 64, 100, 128, 256. The best results for batch size was 128 which gives us 26% test accuracy rate. In this Model we implement 3 layers of convolutional neural network and 3 ReLU layers and 3 pooling layers and 2 linear layers and in total 11 layers for three different classes. Because the accuracy rate was not enough, we tried other modeling ways. [14]

4.4.3.6. 11 Layered Regression CNN with dropout function

We tried to add a dropout function to the 11 layered networks. With 11 layered network which has no batch normalization but has pooling, conv2d, and ReLU, the training accuracy was %56 and test accuracy rate occurs as 41% while it has mean absolute error as loss and error functions. Then changing this regressor to the mean square error for error function, the test accuracy becomes 37%.

4.4.3.7. 11 Layered Classification CNN with dropout function

After this network we change this 11 layered code from regression to the classification by using cross-validation instead of mean square error. Then the training accuracy increased to the 97% while test accuracy increased to 50%

11 layered classification with drop out Confusion matrix	Real Label			
Predicted Label		liked	neutral	not liked
	liked	44%	12%	2%
	neutral	12%	18%	10%
	not liked	44%	70%	88%

Table 9: Confusion Matrix of 11 layered classification with with drop out function

4.4.3.8. 11 Layered Regression CNN for 11 class with dropout function

Then we worked on data in Data Collector App with calculating 11 class from 0 to 10 evaluation. We implemented these data with regression by using mean square error with 11 layers. Without dropout function it the training accuracy was 98%, however, test accuracy is only 12%, and with mean absolute error, it has given 9% test accuracy. When the dropout is used the training accuracy decreased to 18% and the test accuracy decreased to 6% by using mean square error. By using Mean absolute error the test accuracy calculated as 9%. Because Mean square error gave better result the confession matrix calculated for it. In total 33% accuracy rate.

11 classed regression without dropout Confusion matrix	Real Label											
Predicted Label		0	1	2	3	4	5	6	7	8	9	10
	0	0.0465	0.0801	0.002	0.027	0.0505	0.0079	0.0001	0.0944	0.079	0.0849	0.0592
	1	0	0	0	0.0021	0.0006	0	0.0462	0.0006	0	0.0137	0
	2	0.0188	0.0014	0	0.0011	0.0004	0.0001	0.01	0.0029	0	0.034	0
	3	0	0	0.0019	0.0002	0.0011	0	0	0.0006	0	0.0007	0
	4	0	0	0.0004	0	0	0	0	0	0	0.001	0
	5	0.0072	0	0	0.0022	0	0.0791	0.02	0	0	0.0009	0.0213
	6	0	0	0	0.0014	0	0	0	0	0	0.0001	0.0001
	7	0	0	0.0889	0.0406	0.0003	0	0	0.0004	0	0.0001	0.0001
	8	0	0	0	0	0	0	0	0	0	0	0.0001
	9	0.0011	0	0	0.0414	0.0003	0.0001	0.0017	0.0265	0.0058	0.0003	0.002
	10	0	0	0	0.00005	0	0.0071	0.0005	0	0	0.0006	0.0093

Table 10: Confusion matrix for 11 classed regression

4.4.3.9. 62 layered CNN Network without dropout

Another network that we used is composed of 62 layers, which composed of 14 convolutional layers, 14 ReLU layers 14 batch normalization layers, 4 pooling layers, and 1 linear layer in sequential class with cross-validation classification.[15] In this code, we implemented 128 batch size 50 epochs. and the results of train accuracy are 86% while test accuracy is 27% which is still not enough for our purpose.

4.4.3.10. 62 layered CNN Network with dropout

Then, we added dropout layers for training the images with different probabilities to the 62 layered networks to see the best probability for dropout. In the first try we only add one dropout layer with 0.25, 0.5, 0.75 probabilities. The best result was 0.5 dropout probability which gives the 31% test accuracy rate while it gives 76% train accuracy. Then, adding the dropout between every layer instead of in the last layer, it decreased train accuracy to the 65% but increased test accuracy to the 33% which is not satisfying.

Networks	SVM	11 layered without dropout	11 layered with dropout	62 layered with dropout	62 layered without dropout	11 layered with 11 classes
Regression	x	26%	41%	x	x	33%
Classification	57%	31%	50%	33%	27%	X

Table 11: The results of each try shown in this table for simplification. The x means that it is not tried.

4.4.3.11. Results

Based on our observation and the test accuracy we choose to use the data which has 50% ratio. The reason that this accuracy is low that our data is one of the hardest ones. Even in the AM-FED data or Data Collector's data, the video frames include mixed types of actions. For example, if a person likes the video he or she can smile, or else she or he didn't like it they smile because of temper. Also, most of the data that is collected for the like class is very close to the neutral one also it applies for the, not like the class. Most of the frames in real life watching videos are the neutral look on their faces. As can be seen in the confusion matrix for 11 layered classifications with drop out, data is more convincing for finding the dislike and like videos. Because we have a lack of unliked video data we couldn't feed further with 0's. However, when we put more like and neutral videos to test besides the confusion matrix, the test results are closing to the 60% but we need to support this accuracy with more dislike data. Although we tried many different ways to increase the testing accuracy, we ended up using the 11 layered classifications with a drop out model which gives 50% testing accuracy. We define specific code just for testing by only using the output of training result, and connect it to the website. Also when we see the SVM has 57% accuracy rate, which we choose to implement in our site in the end.

4.5. Cloud

As cloud services, we benefit from Google Cloud. We get $300 credit free with google account and it finished in the middle of the semester. Therefore, we get $300 credit free with another google account. We didn't pay anything for the cloud. Although pricing and simplicity of use is an advantage, we have other causes to choose Google Cloud. In terms of latency, speed, processing power, or redundancy in networks, Google Cloud creates difference so we preferred it because of that. Generally, we use a compute engine for all services. Google Compute Engine is Google's Infrastructure as a Service (IaaS) component which is an instant computing infrastructure, provisioned, and managed over the internet. We create a virtual machine instance and as an operating system, we prefer Ubuntu. Moreover, we use an online file storage web service as an IaaS called Google Cloud Storage (GCS) to store model-related storage. It combines scalability and performance with advanced sharing and security capabilities. As we mentioned above at the backend part, we did setups, provided environments for models, and established connections for convenient client-server architecture in our virtual machine instances at the cloud.

5. Testing Details

After each iteration, we did some manual testing for functionalities. We determine our functionalities according to our requirements and we determine test cases. We did some unit tests. Units that are the smallest testable pieces of software are individually and independently investigated for proper operation. In our case, units of the program are functions. Unit testing can be done manually or with automation. We did both of them. We benefit from Selenium Java API which provides functionality about unit tests. Moreover, we use Selenium for database testing which is vital for our project because we have to test if any errors are shown while executing queries or check the response time of queries and fine-tune them if necessary. Data integrity is another important concern for our project so we check whether data is maintained while creating, updating, or deleting data in the database. Moreover we test whether data retrieved from your database is shown accurately in your web application with Selenium, too. The next step of testing was integration testing which is individual units are combined and tested as a group. We have plenty of software module and we have to expose defects in the interaction between these software modules when they are integrated. As a method of integration testing, we follow a bottom-up approach. Each module at lower levels is tested with higher modules until all modules are tested. After this step, we did system testing which is a test phase when the system is complete and integrated. We did black box testing and verify that it meets specified requirements. The next step is about acceptance testing. We investigated the system as business requirements and assess whether it is acceptable for delivery. Additionally, we benefit from the Google developer tool

that provides a test for performance, accessibility, best practices, and SEO which is search engine optimization. In the beginning, we countered very low rates. Fortunately, Google provides a detailed report about how can a website be improved so we follow these recommendations. We improve results every day until we will take good percents or demo. Below, there is a figure which is updated.

Also, for performance testing, we benefit from some other online tools to ensure the site works all loads.[16] Some investigations about requests, page size, and load time.

For models, testing is very easy because we calculate the accuracy rates of the model in each iteration. We did a lot of experiments and a lot of tests to figure out the best practice of our model.

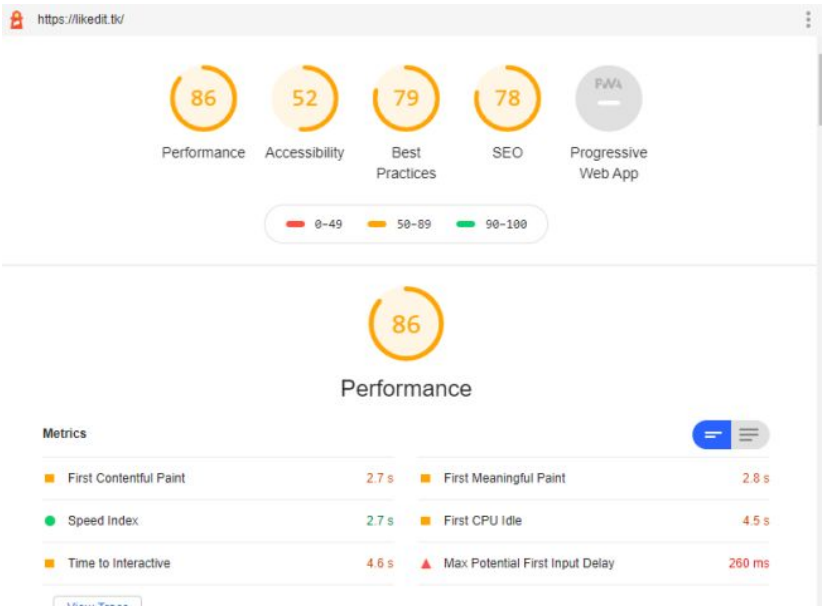


Figure 21. Google developer tool test results about our web page

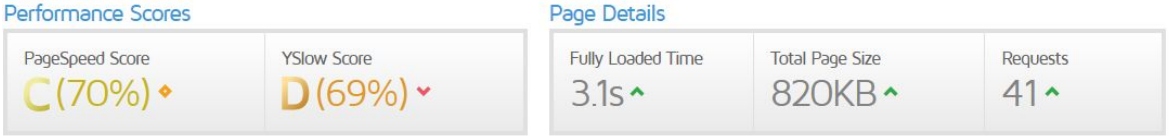


Figure 22. Online testing tool results for our web page

6. Maintenance Plan and Details

Since we run our project on google cloud compute engine, we can monitor specific metrics such as CPU usage, memory usage, disk usage, API usages (Youtube API), and some other metrics as well. So, we develop our maintenance plans according to the observation of these metrics. Some maintenance tasks can be seen in the following table:

Task name	**Frequency**
Check youtube API request metric	Hourly
Check health of the website	Hourly
Check health of the backend server(Node.js)	Hourly
Check health of the Nginx server	Hourly
Check health of the cloud compute engine instance	Daily
Check memory usage of cloud compute engine	Weekly
Check disk usage of cloud compute engine	Weekly
Check CPU usage of cloud compute engine	Weekly
Check google cloud credit	Weekly

Table 12: Task Table

Our maintenance plans include observe the health of the website, backend servers, and google cloud compute engine instance. We can observe the health of the website as a user. Also google chrome provide its user to health check of the website (in terms of accessibility, latency, etc.). We can observe the health of the backend server both from google cloud or other programs such as Putty by

accessing instance and see servers condition. We can observe google cloud compute engine instance health from monitor services provided by google cloud.

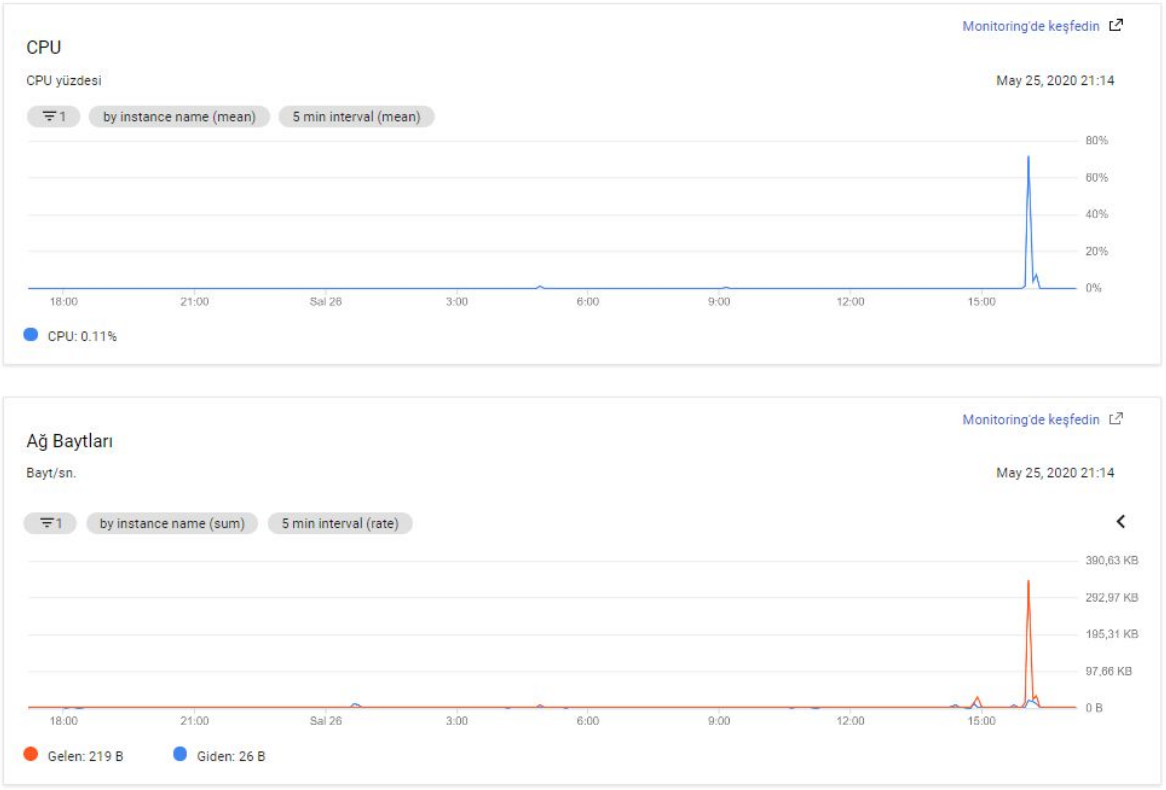


Figure 23: Metrics on the cloud

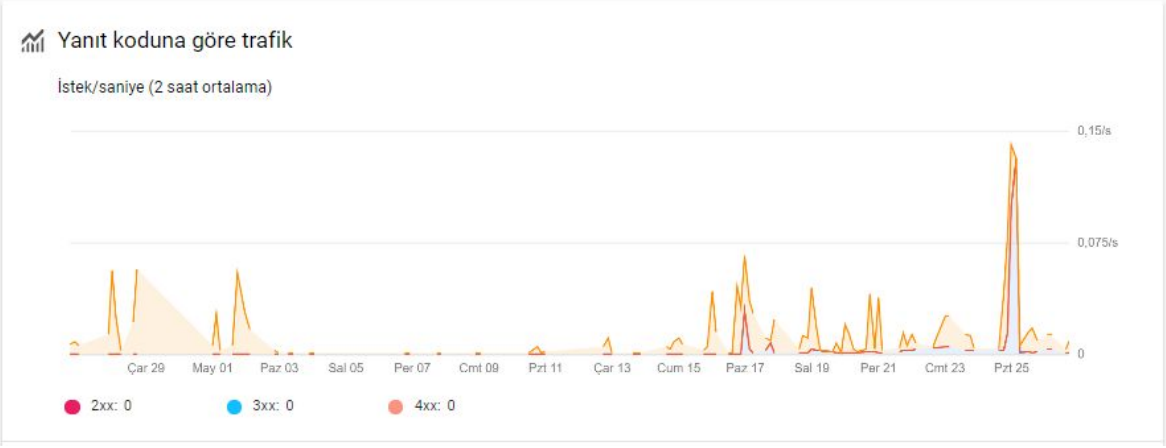


Figure 24: Metrics of Youtube API

After observing these metrics according to schedule, we take some actions if some actions if metrics indicating a problem. For the website and backend server(Node.js), we make some improvements in the code if they cannot provide accessibility and minimal latency. Also, we always restart our backend server automatically if it's going to down. For google cloud instance, google cloud provides

appropriate messages if there is a problem and provide a proposal to solve this problem. We can easily add or remove disk to the cloud, increase the core number of CPU, and increase the memory capacity of the cloud instance. We determine this instance according to average usage and our remaining credit for google cloud engine. Since we have limited credit in the google cloud platform, we try to limit the number of services that we use in this platform. However, as future work, we want to develop more reliable and automated maintenance systems that google cloud provides. For example, using the Kubernetes service of the cloud, we can direct requests to the server according to the number of requests and run several Node.js instances at the same time to decrease the workload on that server. In this way we planning to increase our availability and decrease latency.

7. Other Project Elements

7.1.Consideration of Various Factors

7.1.1. Public Health

LikedIt is an application that requires a digital platform which results in the usage of digital screens like the computer screen. This may result in harmful effects. Also, people can use our feedback which is based on emotions and liking. This feedbacks contribute to improving their social skills like expressing and understanding themselves, well. Therefore, it contributes to their mental health.

7.1.2. Public Safety

LikedIt keeps private information out of reach. Since the credentials of users are kept secure within the Google Account system, that is out of consideration. We also provide choices as desktop and cloud options to investigate data. Users don't have to send their private data to the cloud. They can use the desktop app so their data is processed on their computer. However, the analyses should be anonymous and private to users. In the feedbacks, LikedIt shouldn't give the names of users who watched the video, only the results of the analysis. Moreover, we take an SSL certificate to provide secure connections from a web server to a browser.

7.1.3. Public Welfare

LikedIt will not be affected by public welfare factors. Since it has no potential threats against public welfare. Since the system gets videos from Youtube and Youtube don't keep videos that contain violence or put a warning to mature content and don't show them to users under the age of 18.

7.1.4. Global Factors

LikedIt supports a trends list which so that user would see globally popular videos. This way, users can follow the globally related and important trends of the world. Also, our page is globally accessible via a link. Moreover, we prefer English for our web page's language and it provides reaching more people.

7.1.5. Cultural Factors

LikedIt should support a trends list of the user's country so that users would much likely to see their own culture related videos in their newsfeed.

7.1.6. Social Factors

In order to increase the social interactions of the users, LikedIt enables users to share their analyzes with other people if they like.

7.1.7. Environmental Factors

LikedIt will not be affected by environmental factors. Since it's a software application that has potential threats against the environment.

7.1.8. Economic Factors

In order to make the application available to all users, LikedIt should be free of all charges.

	Effect level	Effect
Public health	4	Overexposure of screen time

Public safety	8	Securing the private data
Public welfare	0	None
Global factors	3	Global trends list
Cultural factors	4	Country trends list
Social factors	2	Sharing of analysis
Environmental factors	0	None
Economic factors	4	Zero cost for application

Table 13: Factors that can affect analysis and design.

Other than these factors, we had to collect our own data to not deal with any copyright issues with foreign datasets, so that's why we've spent a semester to implement such software called Dataset Collector App. Finally, to keep our spendings on the project as zero, we've used the free quotas of Cloud, APIs and found free themes for the front end. But to be novel, we've changed the themes to our own tastes with respect to usability.

7.2. Ethics and Professional Responsibilities

The system keeps the information about the user that is got from the Google API. To respect our users' privacy, we don't get unnecessary information like a birthday. People's privacy is the most important thing for us. We try everything to provide safe software to our users.

The system holds data about users' rates for liking, types of emotions, and gaze data of the videos. However, the program doesn't hold any video or any photo of the user. So, our system does not violate the privacy of users. Personal data about the results of the analysis of the videos of the user is not shared with third parties.

Furthermore, we will inform the users about why we need permission and we ask for their permission for such things as using the camera of the computer. We also explain all of our working principles in our 'About' page on the website.

Lastly, during the implementation stage of our system, we avoided copyright infringement issues by giving priority to open source libraries.

7.3. Judgements and Impacts to Various Contexts

Judgment Description:	Fully analyze the user's reaction while watching a video and give proper feedback	
	Impact level	Impact Description

Impact in Global Context	7	The real liking scores can be used globally which will result in the change of liking systems of many websites such as YouTube and also can be used to generate lists like Global Trends.
Impact in Economic Context	0	It is completely free, so it has no economic context.
Impact in Environmental Context	0	None.
Impact in Societal Context	3	People can share their feedback reports on social media or their friends which will draw attention.

Table 14: Impact Table

7.4. Teamwork and Peer Contribution

From the beginning of the year, every week, we schedule meetings to share responsibilities, equally. The whole team works as a full stack developer. We share leadership and democracy is the cornerstone of our team working. We discuss ideas, designs, problems, and analyses with respect. In the first semester, we've designed, analyzed, and planned our project and we've implemented a dataset app to collect datasets. In the second semester, even from the semester break, we've started our main project with implementing models, building cloud, front-end, back-end, their connections, and testing. For reports and general design, everybody takes equal responsibility.

Every team member worked with almost every feature of the site but they are mainly focused on:

Zeynep Hande Arpakuş: Worked mostly on building the database of the system. Implemented the requests from the frontend to the backend for the database and the connection between the system and the cloud database. Also, for frontend worked on the general design of the website, HTML part, and visualizing the analysis result.

Zeynep Ayça Çam: Worked mostly on building the backend of the project. Create the basics of Node.js code and implement a system that runs models. In the frontend part, implement requests to backend servers and view of analysis. In the dataset collector app, work on interface and video player.

Muhammet Said Demir: Worked mostly on building the back end and the front end of the project. Especially worked on the camera-related features on the dataset collector app and the website, such as gaze tracking, heatmap, calibration. Also, it mainly focused on video players, YouTube APIs, and the overall design of the site.

Zeynep Nur Öztürk: Especially worked on deep learning model, training and testing the CNN-RNN model, normalize the data. Focused on Liking Model. Worked on the YouTube APIs and google authentication for the login system as a backend. Worked on Dataset Collector App.

Elif Beril Şayli: Worked mostly on building deep learning models, the front end of the project, and the cloud. Especially, worked on deep learning models mainly focused on the emotion model. Established cloud, cloud connections of the project, and client-server architecture. In the frontend part, the general design of the web page and demonstrations of the model's feedback. In the Dataset Collector App, I worked on data storage and Dropbox requests.

7.5. Project Plan Observed and Objectives Met

The project met most of the expectations. All of the main functionalities are implemented. However, there were two features that we were not able to implement. One of them is the recommendation engine which bases the liking rates of the user. We've added this part to our feature work. The second one is to automatically send the feedback results to the video owner anonymously which we thought to be too early for the demo stages of the project. But the user can still share the feedback link, which enables them to send it to the video owner but not anonymously.

7.6. New Knowledge Acquired and Learning Strategies Used

In this project, we've acquired a great deal of backend, frontend, cloud systems, and deep learning model knowledge. Especially we've worked with JavaScript, Node.js, Python which we were not really familiar with and after the project we've learned many things about them. Since all of the team members worked mostly as Full Stack Developer, we all had the chance to try new areas and learn about them.

For the Data Collector App, we've worked with Java and enhanced our knowledge about object-oriented design, threads, libraries, data storage and requests to dropbox, etc. The rest of this section will be about the likedIt.

For the frontend parts, we've used mainly HTML/CSS and JavaScript codes which every team member learned a great deal about them. We've learned about the styles, designs, page elements, interactions between pages, etc. For the frontend, we realize that there are many details for the user interface which is very important to interact with the user. Little things create a big difference.

For the backend parts, we've mainly worked on Node.js and JavaScript for post and get requests to store the persistent data, retrieval of the data, validation of data,

etc. We've learned about requests and data management from this part. Data security is also another thing for our project.

To establish a cloud, we follow the documentation of Google Cloud. These documents are very beneficial for cloud computing. Also, connections between front-end, back-end, and cloud are new challenges for us. We learn and implement client-server architecture with detail.

For machine learning parts, we follow some concepts and we learn these concepts from websites and related lessons such as CNN, RNN by using PyTorch and TensorFlow libraries, and so on. We also benefit from some online courses about deep learning and machine learning. Before the senior project, we didn't do a large scale project about deep learning. It is a very valuable experience for us.

In our project, the user can select to use the desktop application to run OpenFace. So, the user should have OpenFace and Node.js instance on the computer. To solve this, we learned how we can create an installation package for Windows, open this application through a web page with custom URL protocol, and make communicate our website and the computer.

LikedIt is basically a video platform and these videos are pulled from YouTube and to log to the website, the user should use his Google account. For these features, the system is using APIs of Google and YouTube. So far, we learned how to use these APIs and integrate them into our program. Again for this, we benefited from Google's tutorials. Also, we used other tutorials too [17].

YouTube was the most important tool to learn new technologies that we don't know and was the place that we've first visited place in each unknown method or technology [18]. Finally, for all kinds of bugs, StackOverFlow was one of the most visited places for our project [19]. Of course, in all these processes we've always consulted with our project advisor. He directed us about how we can improve our strategies and ideas, with his experiences and knowledge.

In conclusion, we've experienced every part of the project, from the creation to the release. In each and every part, we've experienced difficulties, tried to overcome them, tried to create new ideas and new ways to implement our ideas. Overall, this project has thought us all, including teamwork and social skills.

8. Conclusion and Future Work

As a future work of LikedIt, we have several aims as listed below:

- Recommendation engine: Currently, LikedIt doesn't show recommendations based on users like rate. In the future, we want to create an algorithm to recommend videos from youtube based on users like rate.
- Emotion feedback for desktop app users: Since we cannot include the emotion model for desktop application, we cannot provide emotional feedback to the desktop app users. In the future, we plan to add the emotion model to this application as well.
- Increase accuracy of liking and emotion model: With large datasets, the model's accuracy levels can be increased. Especially, for liking model, we need more data.
- Increase availability rate of our system: Since our system has limited user numbers, we don't create any system to handle a large number of users such as 1000. In the future, we will use Docker and Kubernetes systems to run multiple instances of our server and increase the availability of our system while providing reasonable latency.

9. User's Manual

The website is available for any kind of user at https://likedit.tk/

9.1. Homepage

On the homepage of the site, it is shown, 6 trend videos globally and 6 trend videos of Turkey at that time. When the user signs in via the sign-in button on the page, users can also see 6 videos they liked and disliked most recently. To use the website properly and get feedback on the watchings, the user should sign up. After signing up the user should make a calibration for healthy gaze data. It can be done via the 'Calibration' page that can be reached from the sidebar.

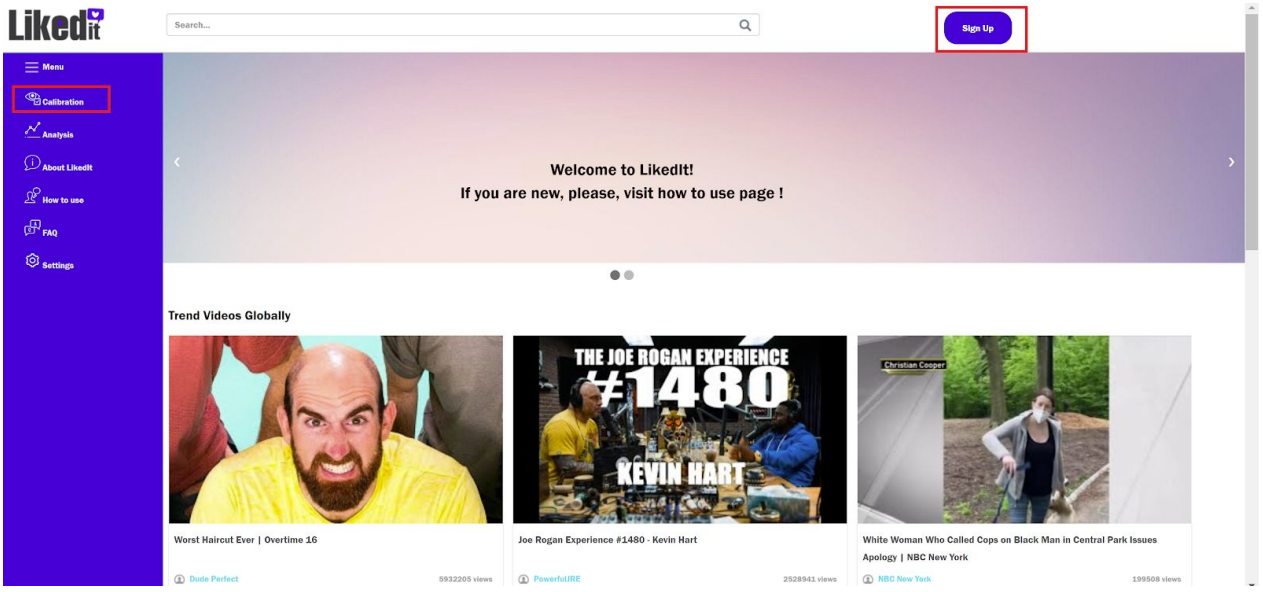


Figure 25: Homepage of LikedIt

9.2. Calibration

In the calibration page, the user will calibrate the gaze tracker module. In order to do that first, users have to adjust their faces properly so that the face detector algorithm can work and puts the green face on top of the user's face. Then, users have to click on the dots that are seen on the page while looking at them. When pressed, the colors of the dots will change from cyan to black. After the first click, there will be a circle that tracks the user's gaze. When the user finishes clicking all the dots, this new circle should be pointing accurately to the location where the user's looking at. After testing that, the user should press the "Done" button to return to the main page.

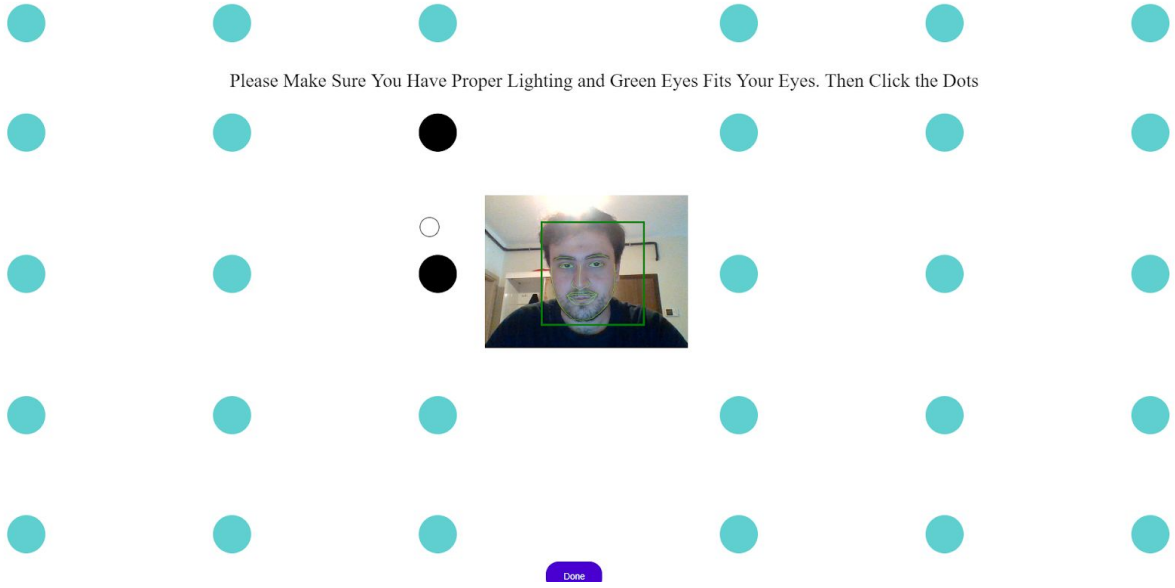


Figure 26: Calibration

9.3. Watching a Video

For watching a video the user can click a video at the homepage or search for a specific video by using the search bar at the homepage. For searching it is enough to write a word with English characters to the bar and then press 'Enter' or the click the search icon. After that 12 videos will be shown on the page. To watch one of them again, the user can just click.

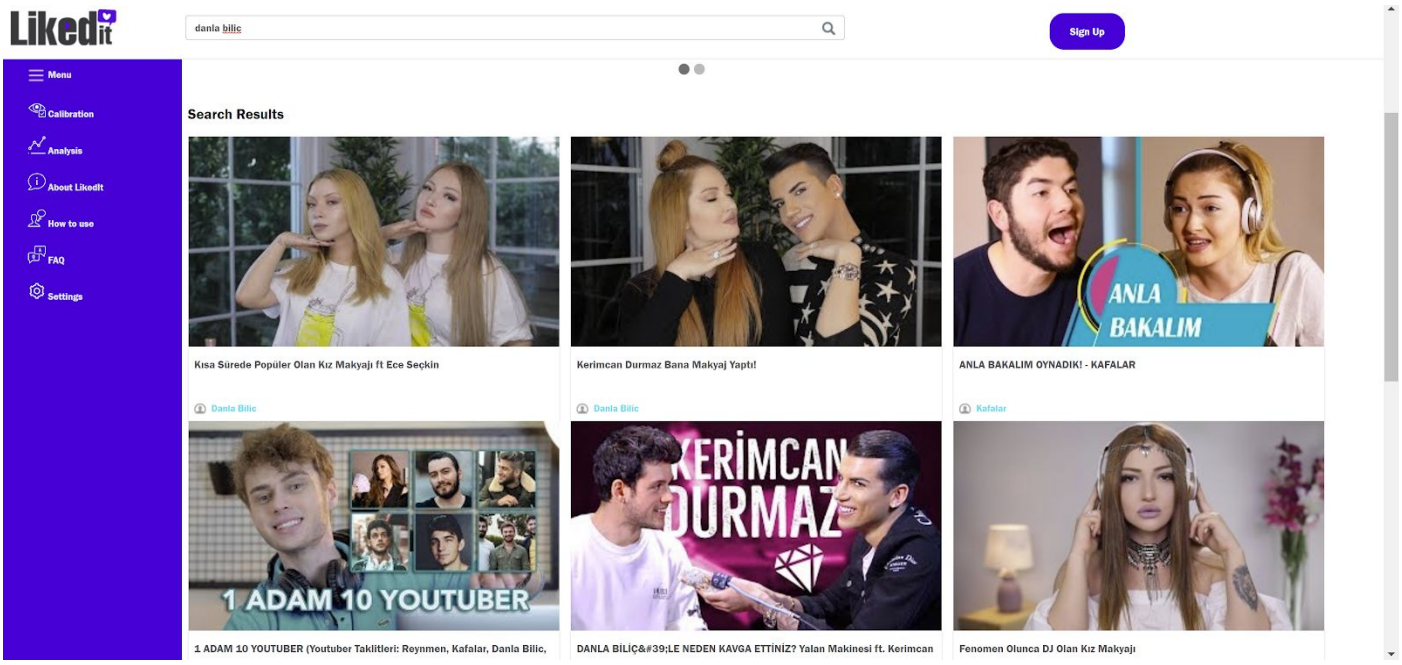


Figure 27: Searching

9.3.1 Playing the Video

After clicking a video, the user is directed to the player page. On this page, in the beginning, the system wants permission to access the camera of the computer. If the user gives permission, then there will be a little box that shows the user's face via the camera. This box is for the user's adjusting the position for proper gaze data. The user should arrange themselves as fitting the green face to their own face. After fitting they can close the box by clicking the 'Toggle Face' button. If they want they can see it again by using the same button. When they close the box the playing button will appear and the video can be played. After watching the system will send the information to the backend. If the user tries to close the page before the process is completed, the page will warn the user that the analysis can be affected.

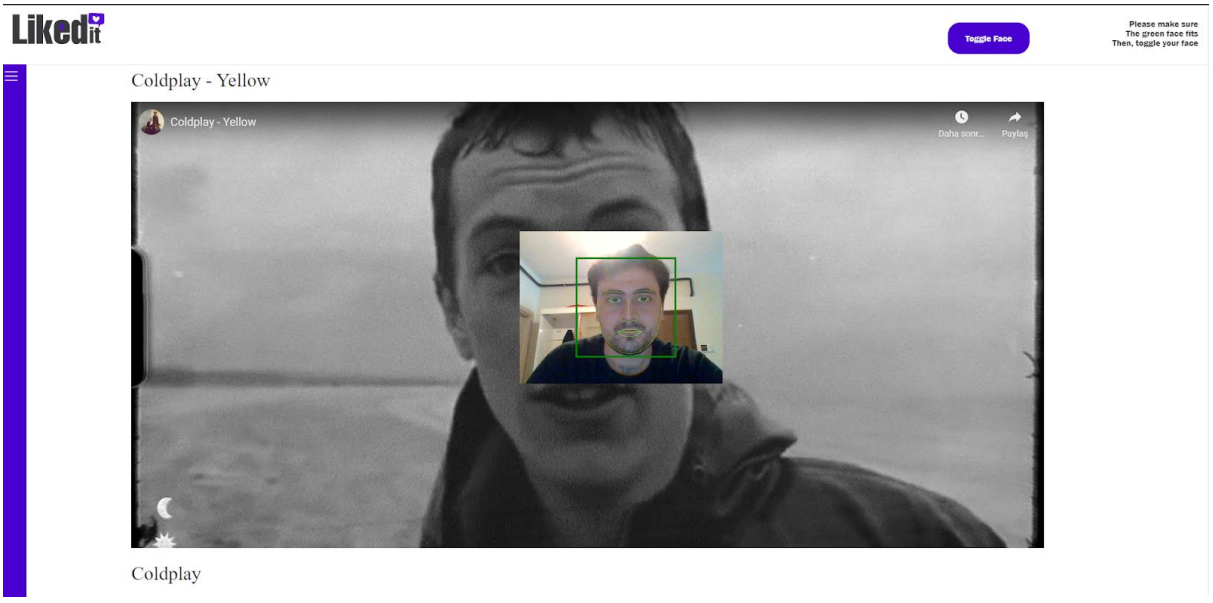


Figure 28: Player

9.4. Analyses

User can see their previous analyses on this page. User can see the source image of the video along with its title and the date of watching. Users can press any one of these to see their specific analysis. If the user had never watched a video, this page will be empty and there will be a text that says “Please watch a video to get feedback”. If the user didn’t sign in, there will be a text that says “Please sign in”.

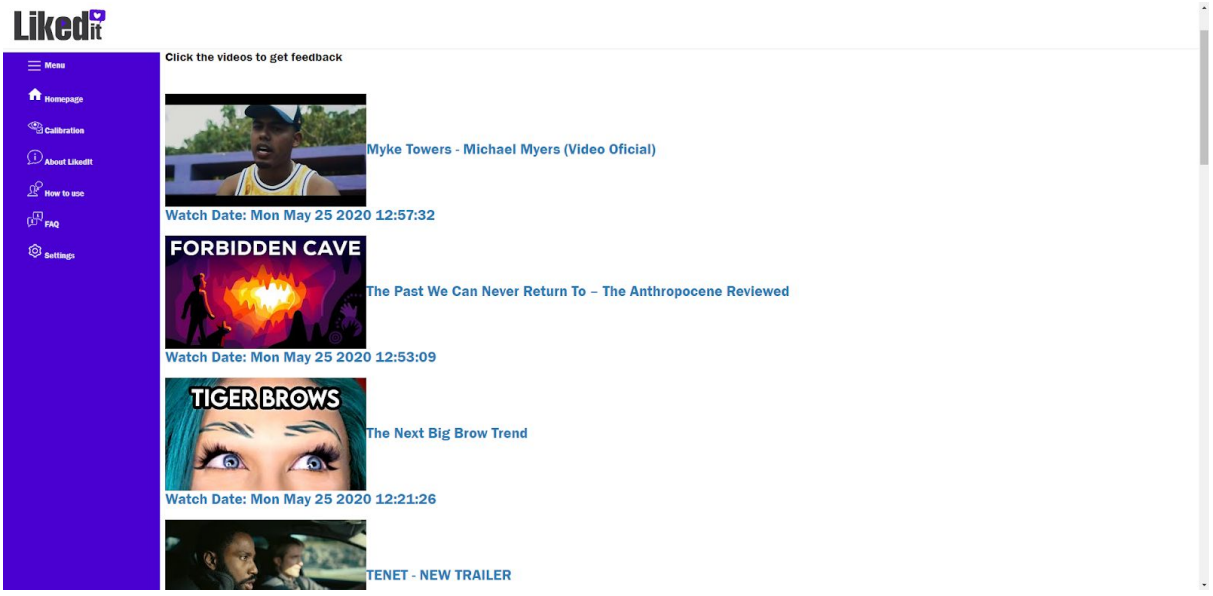


Figure 29: Analysis

9.5. Feedback

When clicked on an analysis on the analyses page, users are redirected to the feedback page of that specific analysis. There, they can see their heatmap of gaze. The heatmap can be watched by playing the video. Also there are two plots about their liking scores and emotions scores. The user can choose the type of analysis that he/she wants to see by using the dropdown list on the page. Users can click on a point at the plot to seek that second at the video and watch it from there. There are also buttons of pause/resume to help the user to navigate. When stopped, the heatmap of the gaze will also be stopped.

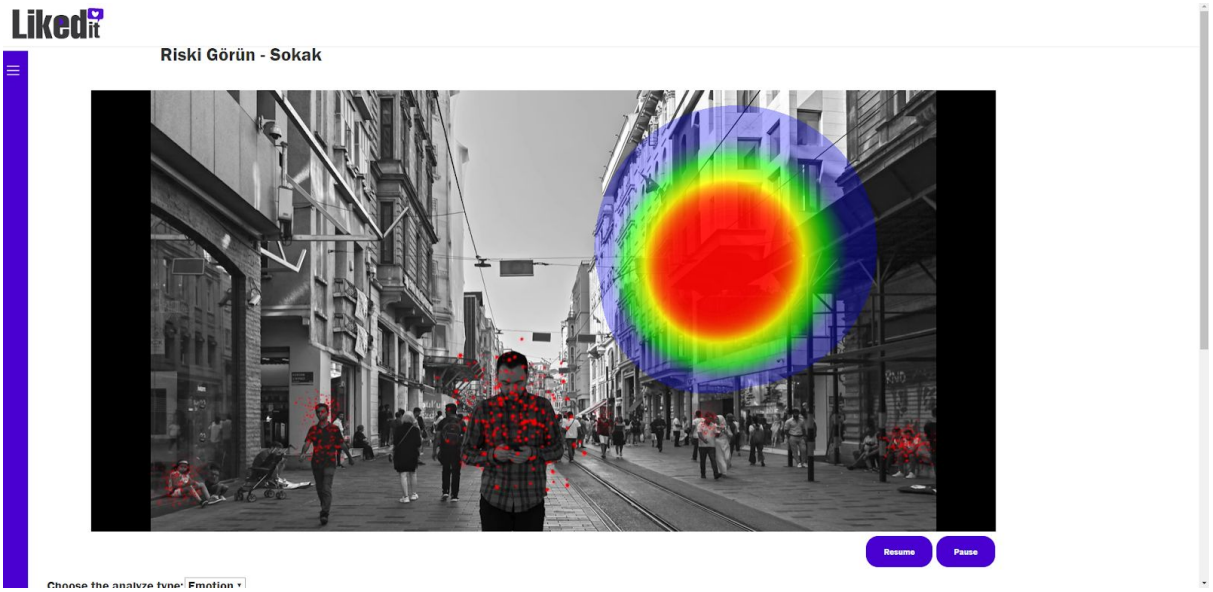


Figure 30: Feedback page, heatmap while the video is playing

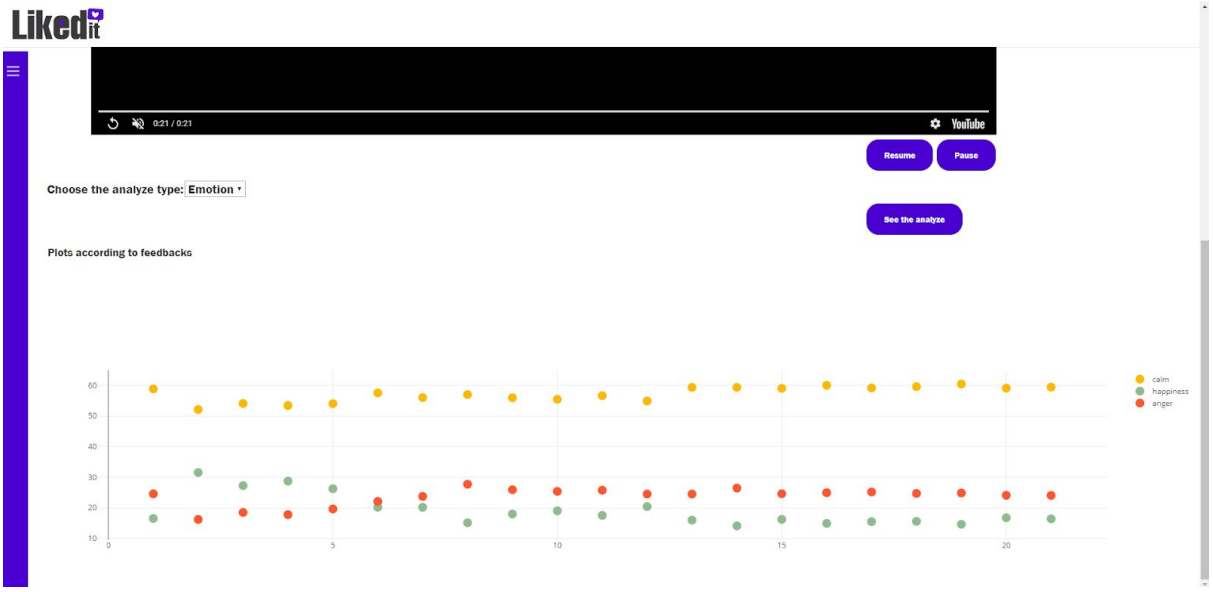


Figure 31: Feedback page, emotion plot

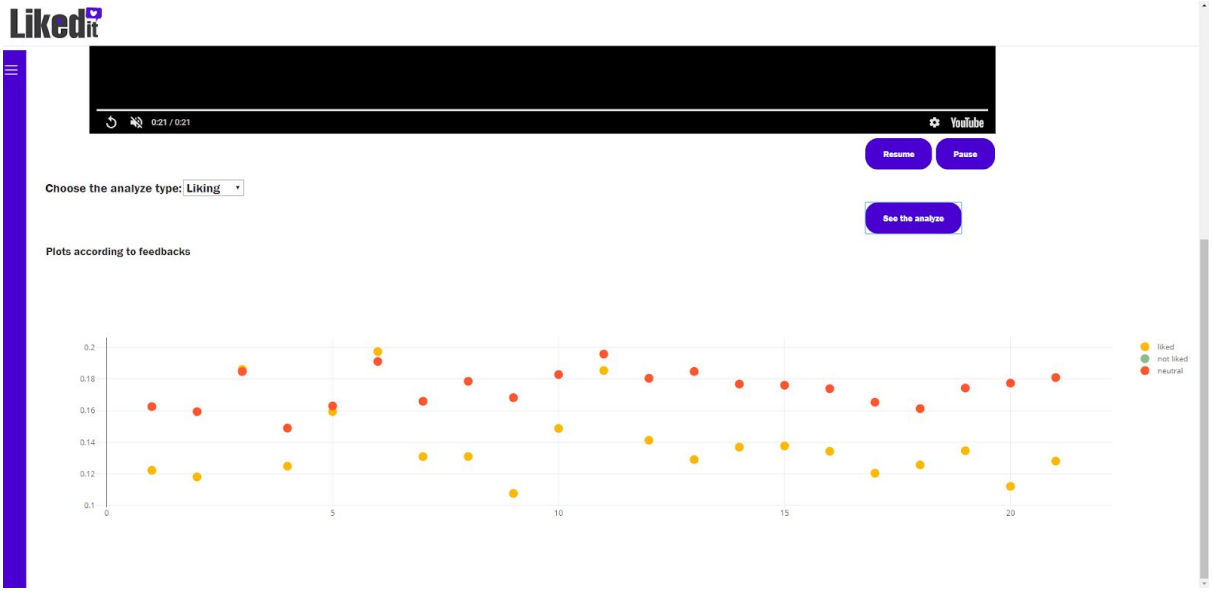


Figure 32: Feedback page, liking plot

9.6. How to Use

Whenever a user feels stuck within the page, they can go to the how to use page from the sidebar. This page includes information about how to use the system and its features properly. There are texts which are supported by images to help to clarify the way that site works.

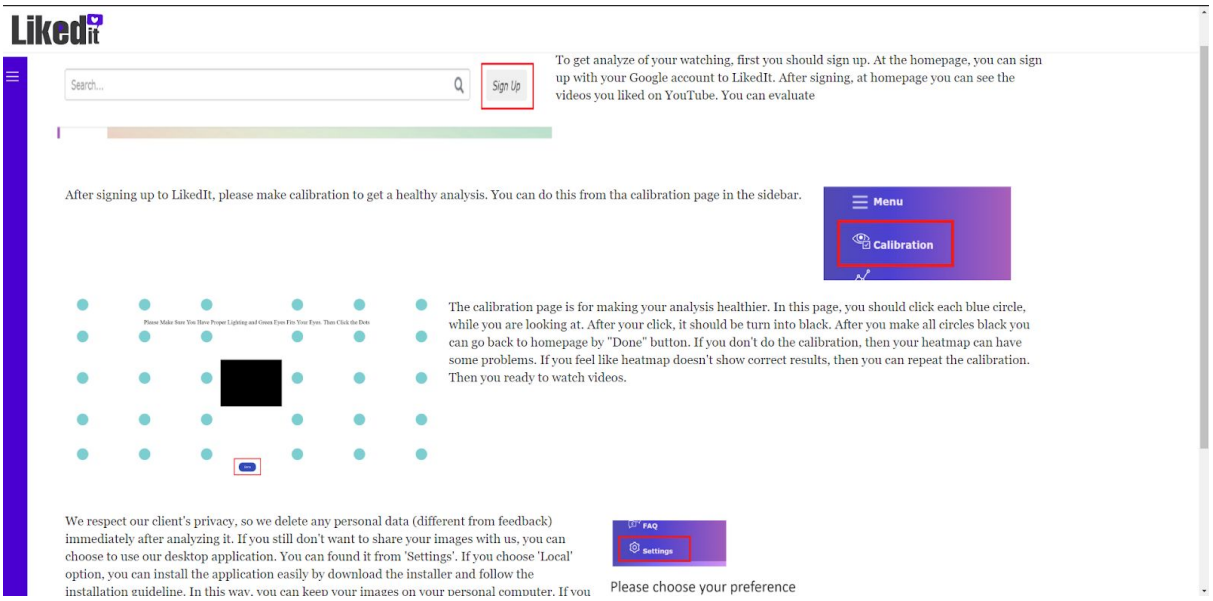


Figure 33: How to use

9.7. Settings

To this page, the user can reach by using the sidebar. At this page the user can choose the preference for the way of working the system. If the ‘Cloud’ option is chosen, the open face will work on the Google cloud and the images will be sent to it. If the user chose the ‘Local’ option, then the local LikedIt will be installed and Open Face will be worked locally so the images of the user will stay at the local. If any choice is not done, then the system assumes the preference is ‘Cloud’.

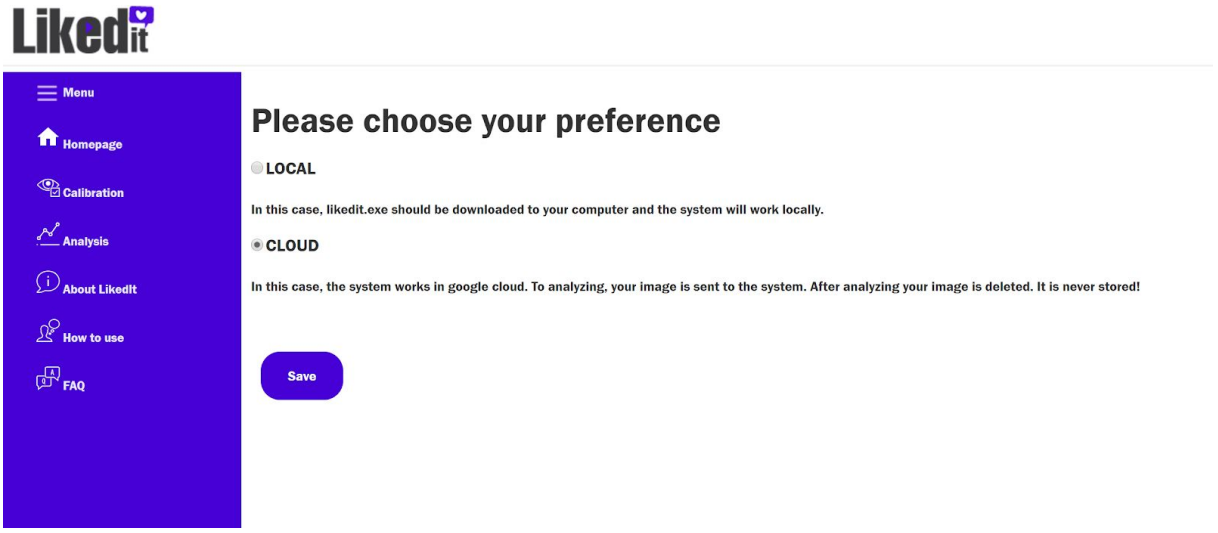


Figure 34: Settings

9.8. Installation Process

To install the desktop application, you can download the installer from the website. This installer is for Windows users. After downloading the installer, the user can easily install the desktop application by following the installation guide. The installation process does not take more than 10 minutes. There is no need for any other installation process. If users don't want to use desktop applications, there is no need for any installation. Users can use the system directly by accessing our website.

10. Glossary

-A-
Analysis: Detailed examination of the elements or structure of something.
API: Application programming interface
Application: A program or piece of software designed and written to fulfill a particular purpose of the user.

-C-
Calibration: The action or process of calibrating an instrument or experimental readings.
Client: A desktop computer or workstation that is capable of obtaining information and applications from a server.
Cloud: A network of remote servers hosted on the Internet and used to store, manage, and process data in place of local servers or personal computers.
CNN: A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data.

-D-
Database: A structured set of data held in a computer, especially one that is accessible in various ways.

-E-
Estimation: A rough calculation of the value, number, quantity, or extent of something.

-G-
Gaze: A steady intent look.
Google: A search engine to obtain information about (someone or something) on the World.

-M-
MySQL: An open-source relational database management system

-N-
Node.js: An open-source development platform for executing JavaScript code server-side.

-O-
OpenFace: Open source tool capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation.

-R-

RNN: A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence.

-S-

Server: A computer or computer program which manages access to a centralized resource or service in a network.

-Y-

Youtube: A video sharing service that allows users to watch videos posted by other users and upload videos of their own.

11. 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] "Welcome to NGINX Wiki! | NGINX", Nginx.com, 2020. [Online]. Available: https://www.nginx.com/resources/wiki/. [Accessed: 26- May- 2020].
[ikinci referans]
[3] “SSL” [Online]. Available: https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate [Accessed: 26- May- 2020].
[4] "nodejs/node", *GitHub*, 2020. [Online]. Available: https://github.com/nodejs/node. [Accessed: 26- May- 2020].
[5] Open Face”, 2019. [Online]. Available: https://github.com/TadasBaltrusaitis/OpenFace [Accessed: 5- Oct- 2019].
[6] "WebGazer.js: Democratizing Webcam Eye Tracking on the Browser", Webgazer.cs.brown.edu, 2020. [Online]. Available: https://webgazer.cs.brown.edu/. [Accessed: 24- May- 2020].
[7] "Dynamic Heatmaps for the Web", Patrick-wied.at, 2020. [Online]. Available: https://www.patrick-wied.at/static/heatmapjs/. [Accessed: 25- May- 2020].
[8] "JavaScript Quickstart | YouTube Data API | Google Developers", Google Developers, 2020. [Online]. Available: https://developers.google.com/youtube/v3/quickstart/js. [Accessed: 26- May- 2020].
[9] "Integrating Google Sign-In into your web app", Google Developers, 2020. [Online]. Available: https://developers.google.com/identity/sign-in/web/sign-in. [Accessed: 26- May- 2020].
[10] "thoughtworksarts/EmoPy", GitHub, 2020. [Online]. Available: https://github.com/thoughtworksarts/EmoPy. [Accessed: 26- May- 2020].
[11] "Recognizing Human Facial Expressions With Machine Learning | ThoughtWorks Arts", Thoughtworksarts.io, 2020. [Online]. Available: https://thoughtworksarts.io/blog/recognizing-facial-expressions-machine-learning/. [Accessed: 24- May- 2020].
[12] "EmoPy: A Machine Learning Toolkit For Emotional Expression | ThoughtWorks Arts", Thoughtworksarts.io, 2020. [Online]. Available: https://thoughtworksarts.io/blog/emopy-emotional-expression-toolkit/. [Accessed: 26- May- 2020].
[13] Dibeklioglu, H. (2017). Visual Transformation Aided Contrastive Learning for Video-Based Kinship Verification. *2017 IEEE International Conference on Computer Vision (ICCV)*. doi: 10.1109/iccv.2017.269

[14] (n.d.). Retrieved February 2020, from https://pythonprogramming.net/analysis-visualization-deep-learning-neural-network-pytorch/
[15] Olafenwa, J. (2020, January 29). Basics of Image Classification with PyTorch. Retrieved May 27, 2020, from https://heartbeat.fritz.ai/basics-of-image-classification-with-pytorch-2f8973c51864
[16] "GTmetrix Performance Report: C (70%) / D (69%)", *Gtmetrix.com*, 2020. [Online]. Available: https://gtmetrix.com/reports/likedit.tk/I8ISM3yi. [Accessed: 26- May- 2020].
[17] “Machine Learning”, 2019. [Online]. Available: https://developers.google.com/machine-learning/crash-course/ml-intro
[18] “Youtube”, 2019. [Online]. Available: https://www.youtube.com/
[19] “StackOverFlow”, 2019. [Online]. Available: https://stackoverflow.com/