Emotion and Engagement Analysis

by
Garvit Pahal (12264)
Siddhant Saurabh (12715)

Aim of this project was to analyze emotions and engagement of a person while he watches a video online. As the video plays, our project records pictures of users through their webcam. The pictures are taken at a fixed regular interval and sent to our servers for processing. After processing, relevant information are made available to the administrator. Some example usages of the project can be:

- ➔ In our own mookit platform to study the behaviour of students while they watch lectures
- ➔ User response in video advertising campaigns

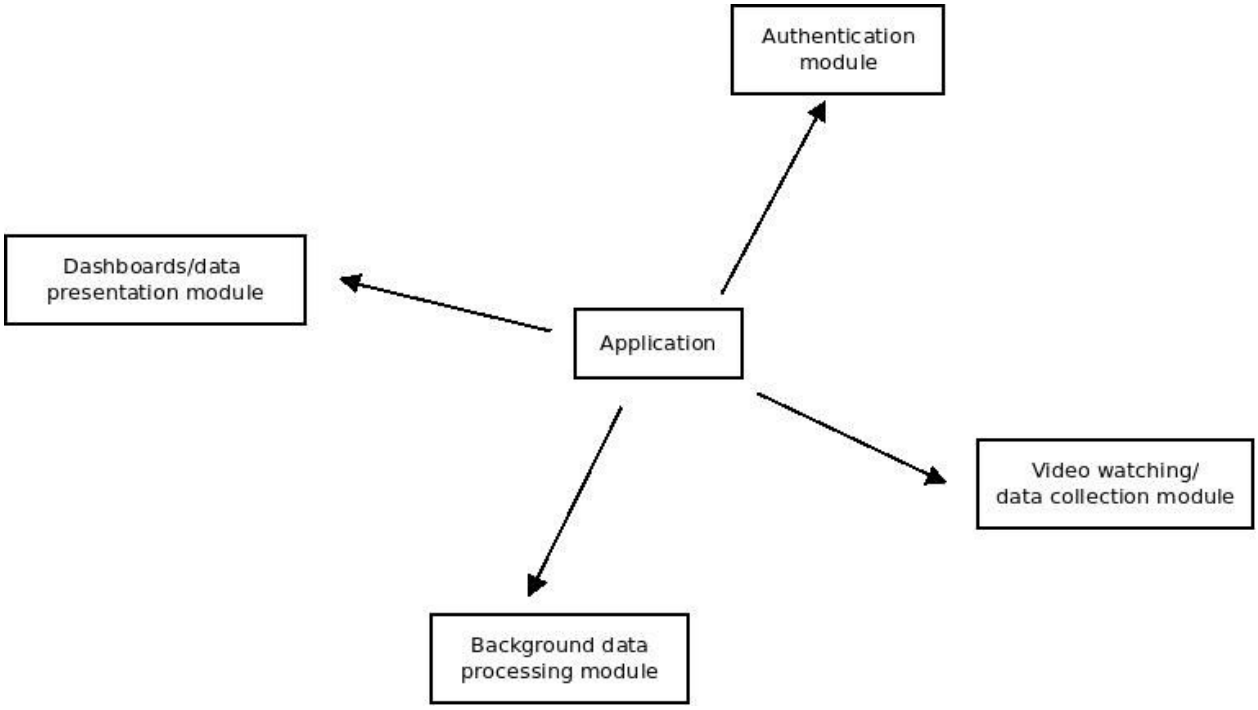
There are two kinds of users in our platform:

- ➔ Who can upload or share links of videos, we call them administrators. Example: Professors and TAs on mookit
- ➔ Who are supposed to watch these videos, we call them consumers. Example: Students on mookit

Both the users need to authenticate on our platform to use the services.

For describing the architecture of our project let's start with describing logical and process views:

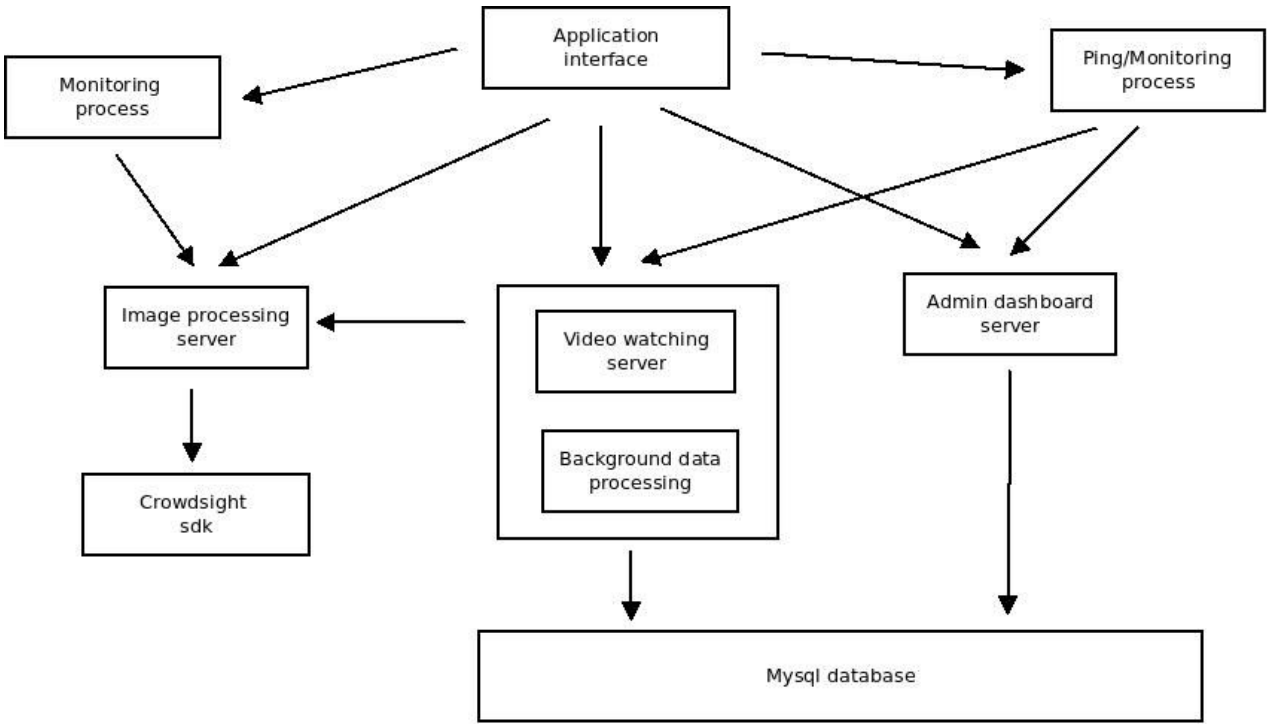
Logical View:



Our application is composed of 4 logical components:

1. Authentication module: This module deals with registration and login of both the administrators and consumers.
2. Video watching/data collection module: This module is for the consumers and is used when they watch a video on our platform. It is responsible for presenting them with videos and collecting images through their webcam.
3. Background data processing module: This module is responsible for processing the photos sent to our server using the webcam.
4. Dashboard/data presentation module: This module is for the administrators and presents them with information that we get after processing the pictures.

Process View:



The application is mainly composed of 3 main services plus 2 monitoring services. They are as follows

1. Image processing server: This component is responsible for processing the images. Communication with this component happens through HTTP.
2. Video watching server + background data processing: This is the server for consumers. The background data processing component calls the image processing server via HTTP for image processing. This server also interacts with mysql database for storing results that we get after processing.
3. Admin dashboard server: This is server for the administrators. It presents the them with the data we collected. This component also interacts with mysql server for fetching the stored data.

4. Monitoring process for image processing server: This regularly checks the image processing server for any errors and restarts it if it finds any.
5. Monitoring process for consumer and admin servers: This regularly checks both these servers for error and inform us via text messages if they are down.

Our platform follows a microservice like architecture

We have 3 separate services which serve different needs:

- ➔ Administrators have a dashboard which runs as a separate service
- ➔ Similarly consumers watch videos through a different service
- ➔ Image processing runs as a separate service

Relevant quality attributes analysis

1. Performance:
   As we have 3 different services running independently on 3 different machines, we can expect our performance to be high. Also Performance critical image processing is done in C++ which is fast as compared to other languages.

2. Availability and Scalability:
   As previously mentioned, we have two monitoring process to check the availability of services. Out of the 3 services, users interact the most with video watching service. So it's critical to ensure its availability. Hence we have a load balancer which balances traffic across two instances of this server. It can tolerate loss of one server. This also improves scalability.

3. Maintainability, Replaceability and Testability:
   Since our application follows a microservice style architecture, we have loose coupling between the services. This leads to better maintainability. If required a service can be replaced keeping the interface same. Example: In our application the image processing service can be replaced easily in case a better solution is available. This architecture also leads to fault isolation and each service can be tested separately to find and fix errors. Also if one service is down, the others may still keep going

4. Technology Heterogeneity
   Because of microservice architecture, we were able to use different technologies for different services: 2 of them are written in Go, 1 is written in C++, Monitoring processes are written in bash and python

Tradeoffs

There were many compromises we made while deciding to use multiple services:

1. Having many services means interaction between them need to happen through network calls, which are slower. So this affects our performance. Also network calls can fail from time to time adding to complexity. Network calls are also insecure so proper care need to be done to protect data from adversaries. We could have chosen an encrypted channel, which again reduces the performance of the system.
2. Since we have many services, we need to manage all of them. This is a complex task. Even for monitoring the servers, we had to write two different monitor programs.
3. All of the services have some common code. This leads to duplication of code and affects future refactoring.

Future Work

Improvements that can be made in the system:

1. Instead of images, videos of the users can be used to improve the engagement and attention analysis.
2. In the current system, MySQL server is the bottleneck. Instead of a single server, a cluster can be deployed to improve availability and latency.