Emotion and Engagement Analysis

CS654 Term Project

Garvit Pahal (12264)
Siddhant Saurabh (12715)

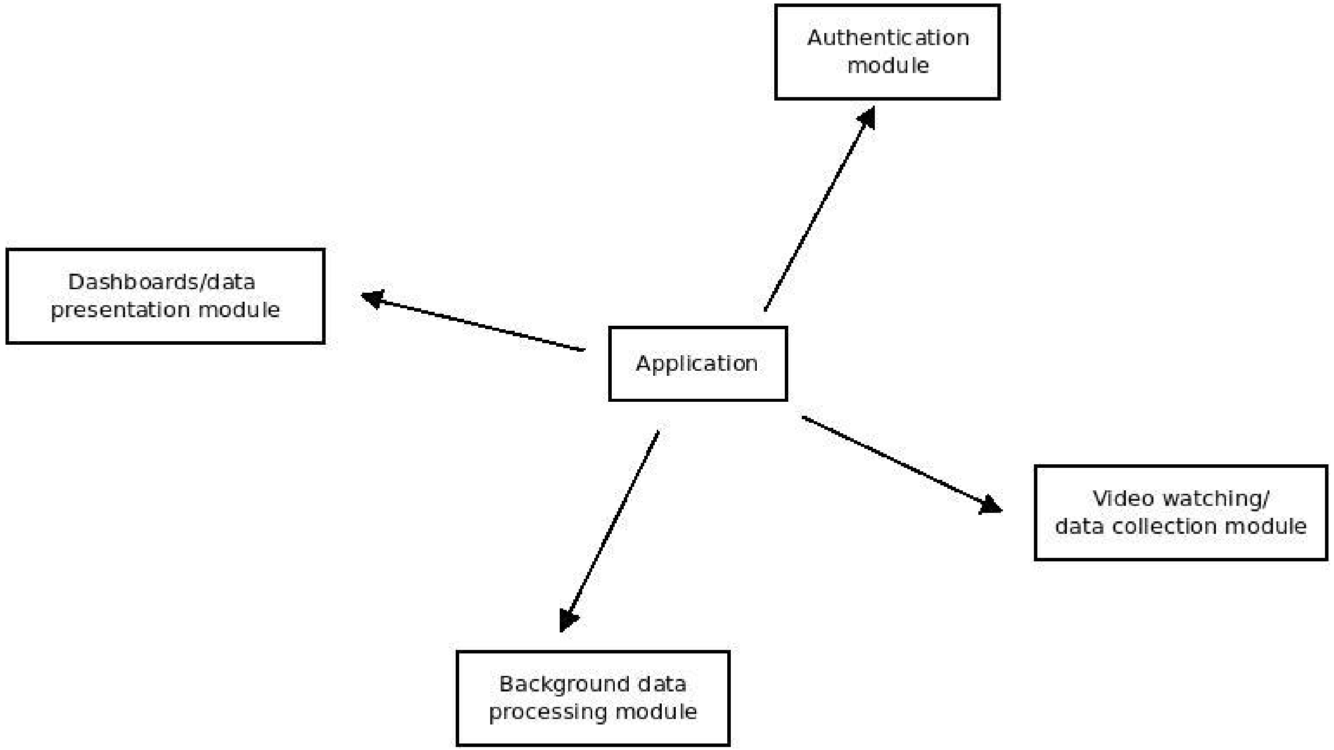
About the project

- A way to analyze emotions and engagement of a person while he watches a video online
- Records viewers pictures at various intervals through his webcam while the video plays
- Processes those pictures and makes related information available to administrator
- Example usage:
  - Our own mookit platform to study the behaviour of students while they watch lectures
  - User response in video advertising campaigns

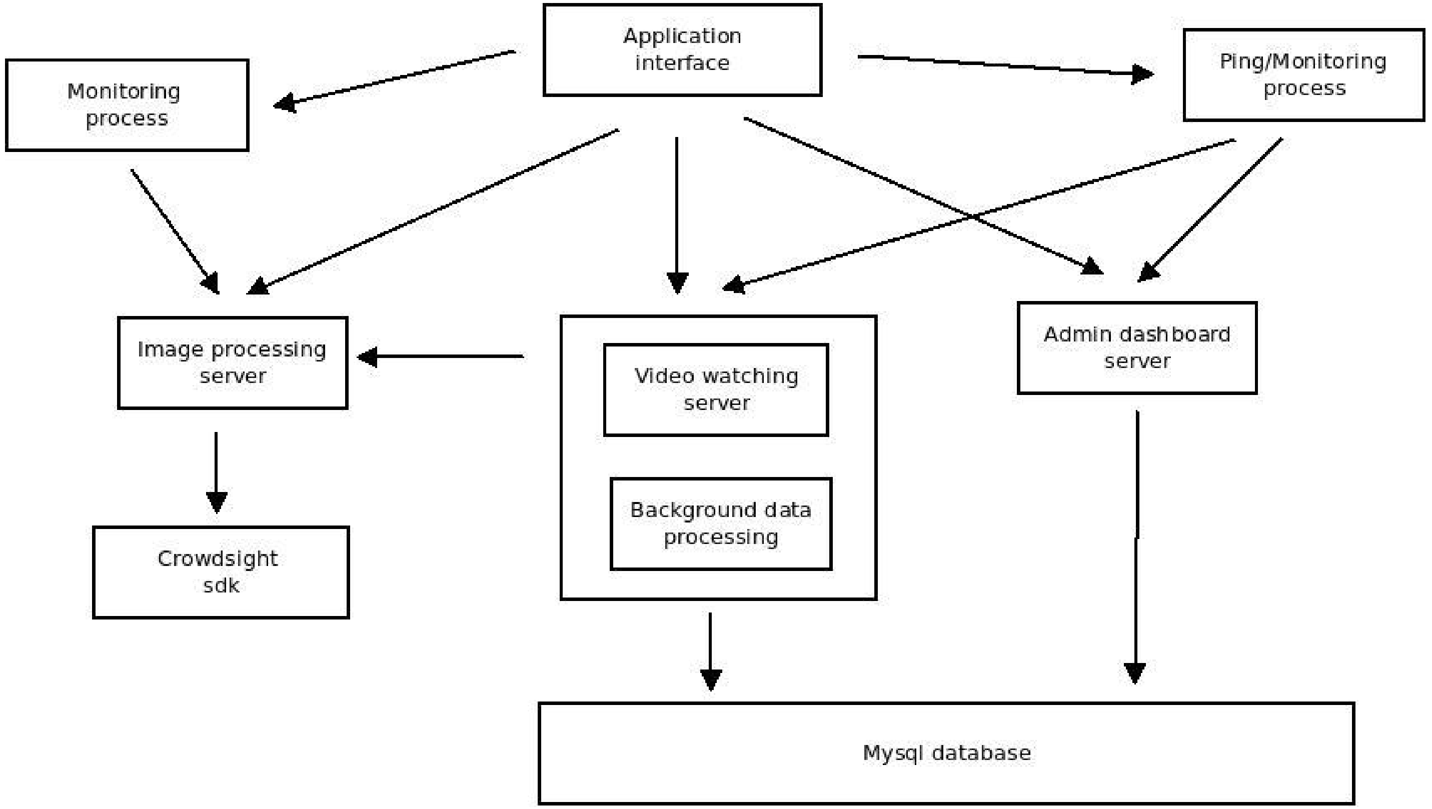
Two types of users

- Administrators: Who can upload or share links of videos
  - Example: Professors or TAs in mookit
- Consumers: Who are supposed to watch these videos
  - Example: Students in mookit
- Both types of users need to authenticate to our platform for accessing the services

Logical view


Authentication module
Dashboards/data presentation module
Application
Video watching/ data collection module
Background data processing module

Process view



Performance

- We have 3 separate services which serve different needs
- Administrators have a dashboard which runs as a different service
- Similarly Consumers watch videos through a different service
- Performance critical image processing is done in C++ as a separate service

Availability and Scalability

- We have two monitoring process
  - One for our image server. It keeps checking it for faults on regular interval and restarts it if required
  - Another for our video watching and admin dashboard server. If fault is detected it notifies us through text messaging

- 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
  - It also improves scalability

Maintainability, Replaceability and Testability

- Our application follows a microservice style architecture
- Loose coupling between the services 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 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

Technology Heterogeneity

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

DEMO

And

Thank You