RACA- Real Time Automated Classroom Attendance

An Automatic Attendance Management System using Face Recognition

Project Report - 1

Team #3

Team Members:

Nikhita Sharma	- 22
Ganesh Taduri	- 24
Aparna Pamidi	- 18
Harsha Komalla	- 11

1. PROJECT OBJECTIVES:

Objective:
In an improving educational world it's been very difficult to keep track of the student's attendance for a lecturer with their limited time for the class. Though we have many other systems to keep track of the attendance they can be easily manipulated. Therefore our objective of this project is to develop a system which keeps track of the attendance automatically using face recognition algorithms with the help of live video.

Significance:
"Automatic Attendance Management System using Face Recognition" is useful not only to keep track of the attendance system but also to reduce paperwork, save time, eliminate duplicate data entry and errors and auto generate various types of reports of class for student attendance. We can also implement this system to other general public meetings to generate statistical information of the members attended and also to keep track of the security surveillance of the meeting.

Features:
It is a mobile as well as a static application, where a user can record a live video and generate statistical reports out of it. The various reports that can be generated are attendance report, total revenue generated out of the meetings, etc. It also provides features to detect an unusual activity in public gathering.

2. APPROACH

Data Sources:
We collected a sample classroom videos and seminar videos for training the model. The various sources for the data are:
- Youtube.com
- Ted.com

Analytical Tools:
We used latest tools and various API's for developing this project.
Major tools used:
- Spark
- Storm
- Kafka

Major API's used:
- Microsoft Cognitive Services
- Califai

Analytical Task:

By using this system we can process a live video of a seminar and can generate various interesting and useful reports through some analytical functions, Attendance report, revenue generated from a seminar, reports about males vs females in a hall, are the major analytical tasks for this project.

Expected Inputs/Outputs:

Input:

A scanned video of a classroom or a seminar hall. Input can also be a live video through a camera in hall.

Output:

Expected output will be the analysis report of attendance of a class/hall. In class room scenario the system gives information whether a particular student attended the class or not using face recognition.

Algorithms:

We used a few Machine Learning algorithms provided by Spark MLlib during training the model to classify the data. The algorithms are:

1. Decision Tree
2. RandomForest

3. RELATED WORK:

Open Source Projects:

- https://github.com/opencv-java/face-detection
- https://github.com/levackt/face-detection

Literature Reviews:

- http://klresearch.org/IJMSTM/papers/v2i3_2.pdf
- http://blog.mashape.com/list-of-10-face-detection-recognition-apis/
- http://ieeexplore.ieee.org.proxy.library.umkc.edu/stamp/stamp.jsp?tp=&arnumber=6227479

4. APPLICATION SPECIFICATION:

System Specification:

Software Architecture:

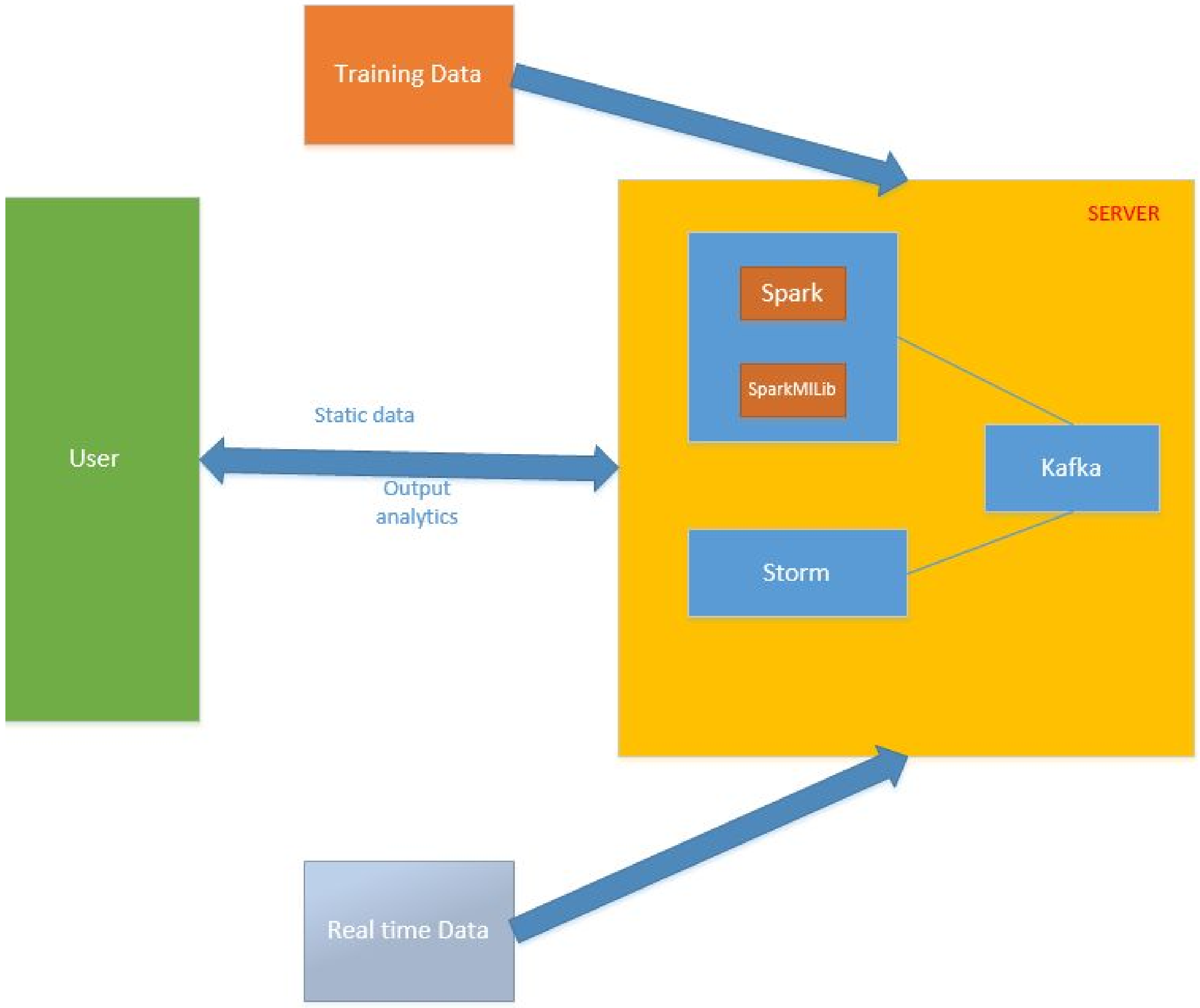


Fig 1: Architecture Diagram

System Flow

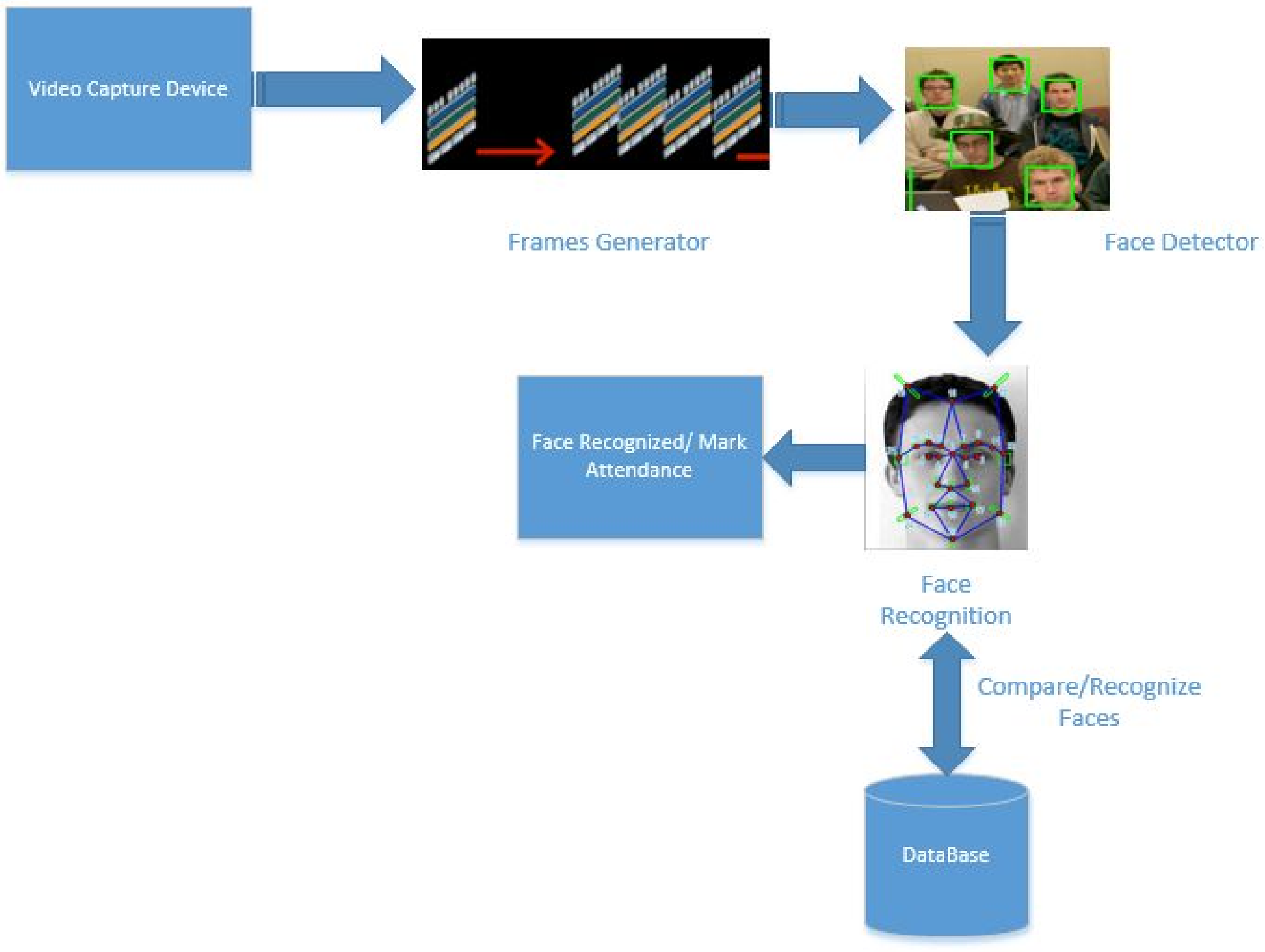


Fig 2: System Flow Diagram

Class Diagram:

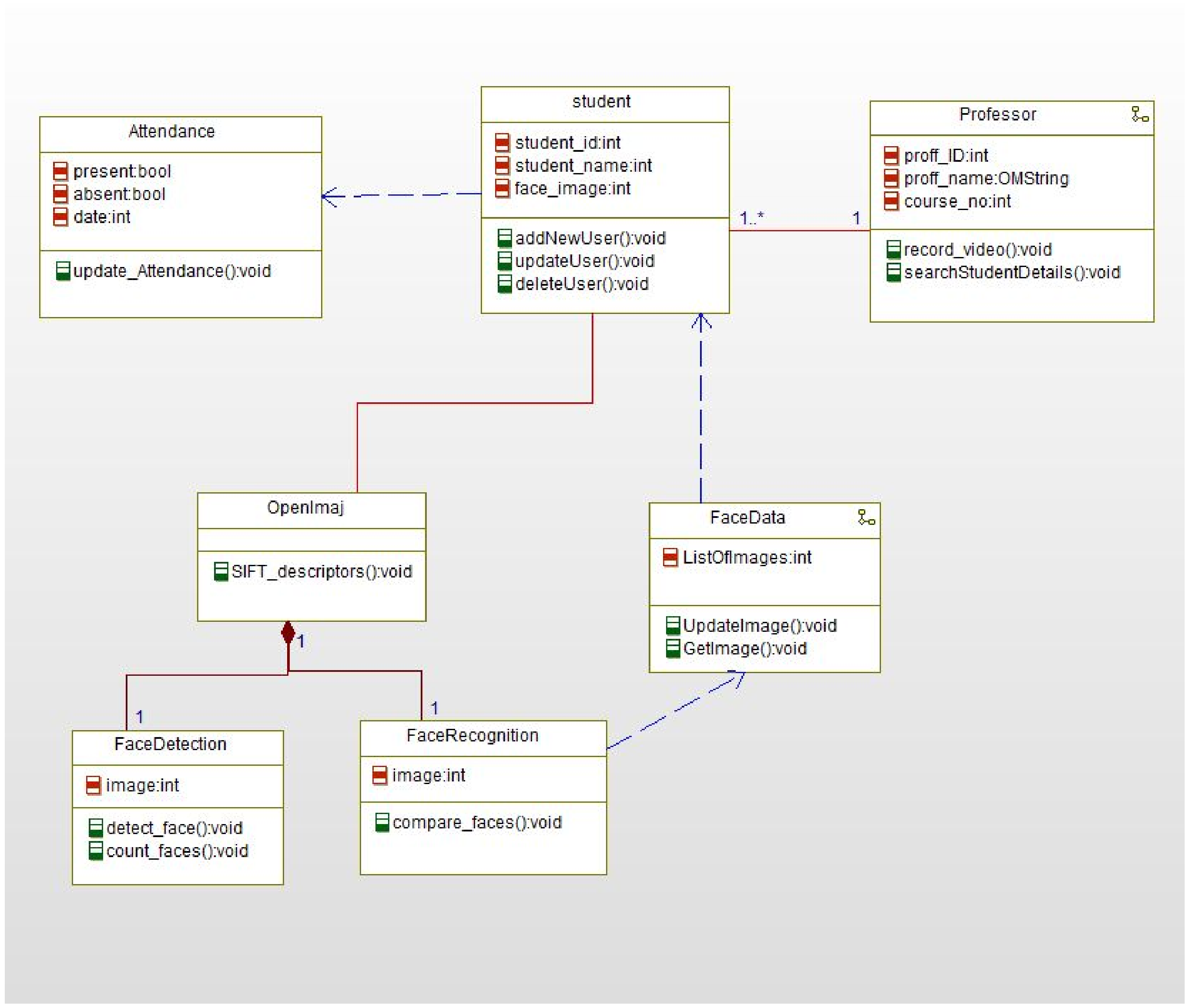


Fig 3: Class Diagram

Sequence Diagram of Whole System:

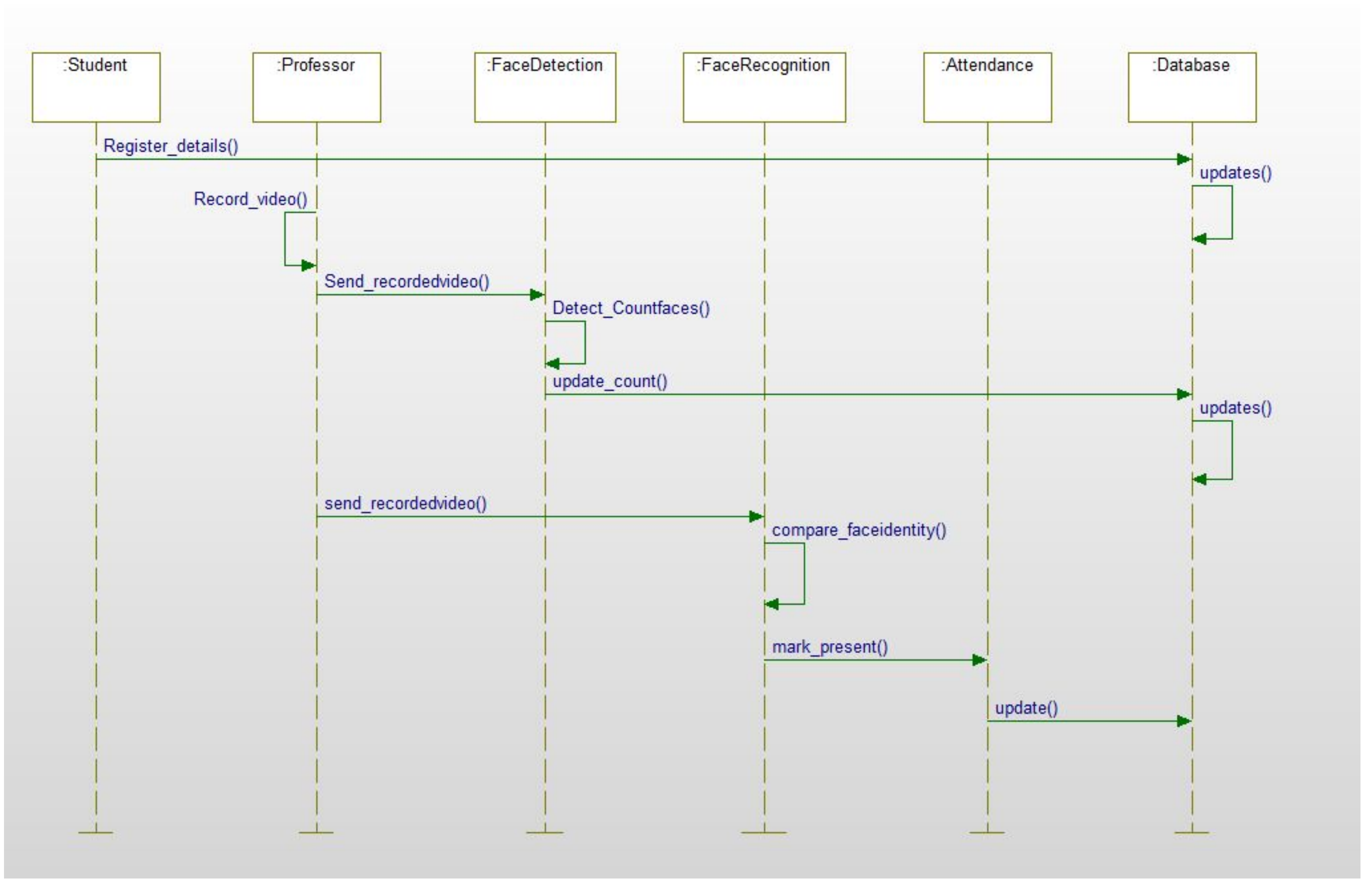


Fig 4: Sequence Diagram

Activity Diagram:

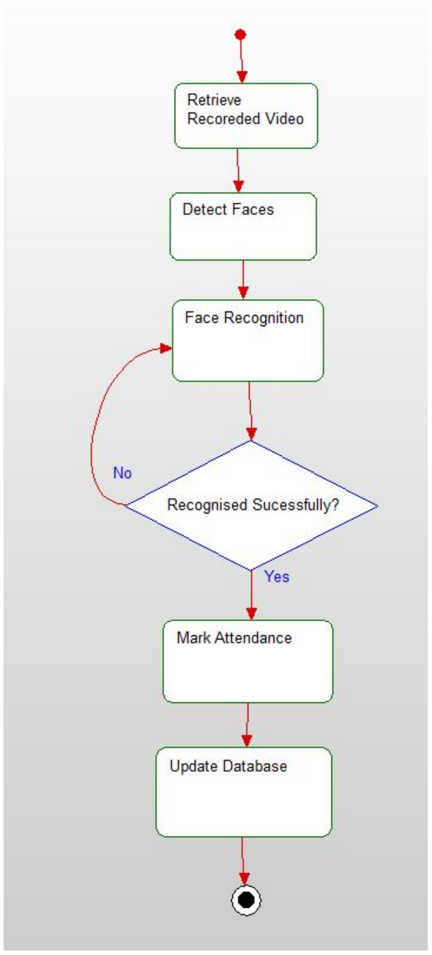


Fig 5: Server Activity Diagram

Sequence diagram:

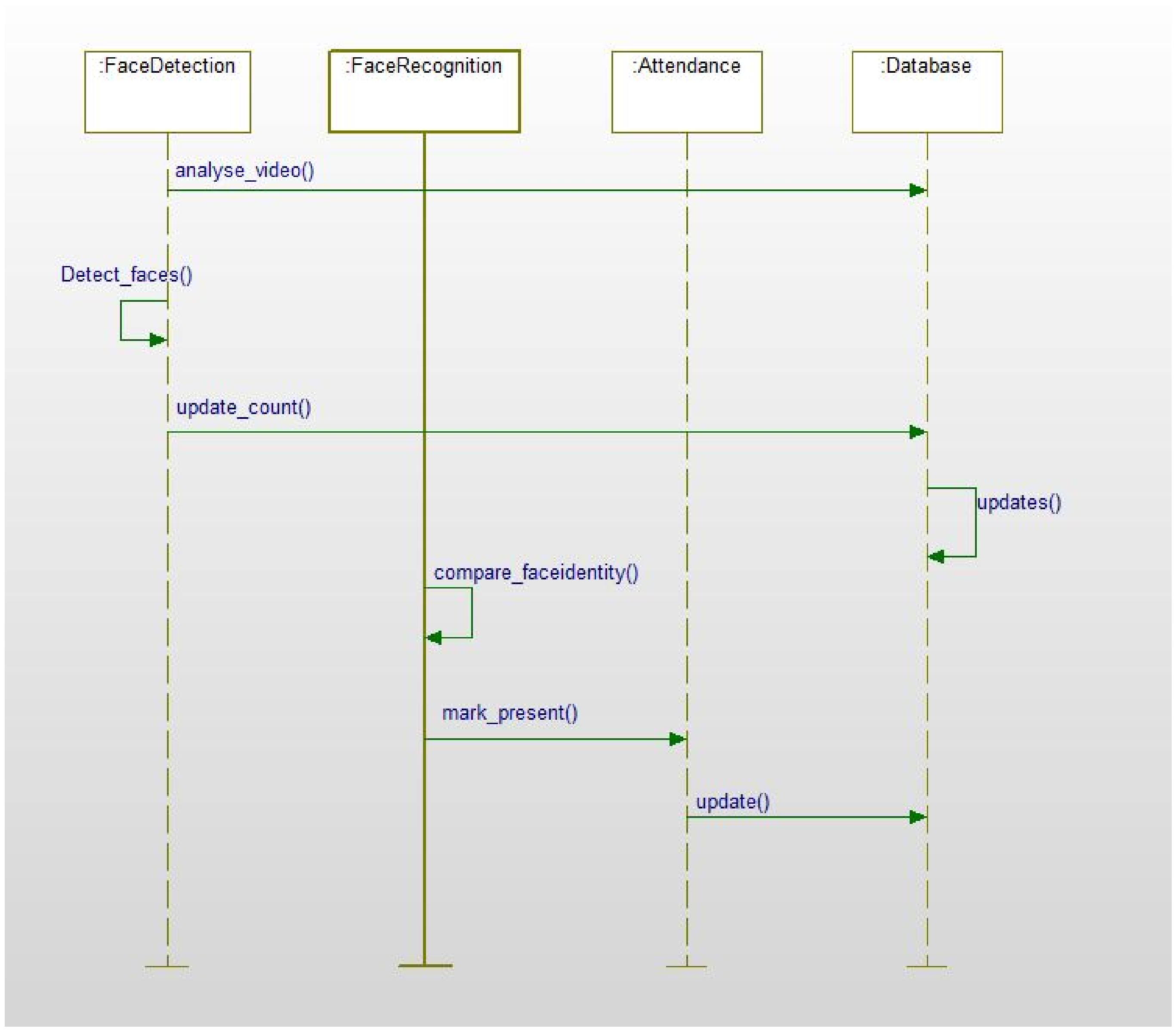


Fig 6: Server Sequence Diagram

Design of Mobile Client:

Activity Diagram:

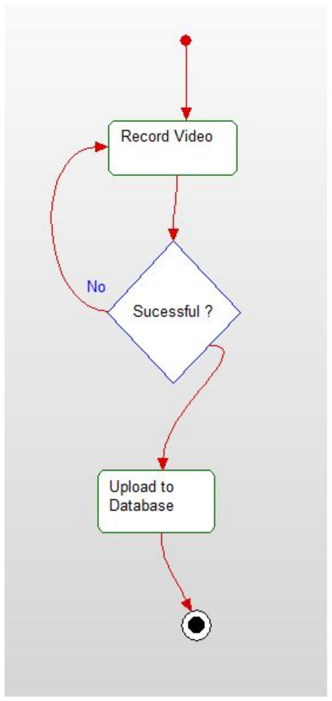


Fig 7: Client Activity Diagram

Sequence diagram:

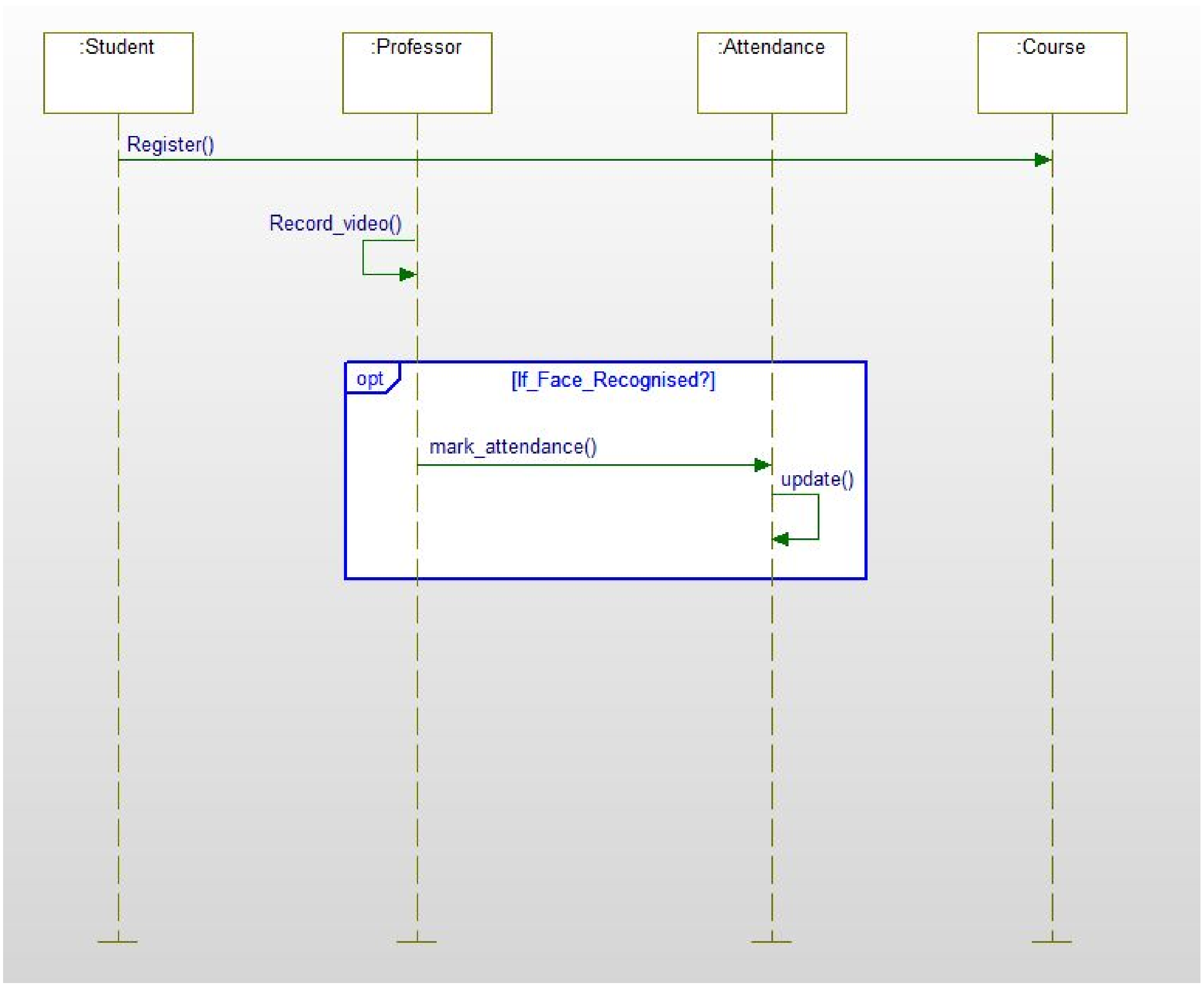


Fig 8: Client Sequence Diagram

<u>**Existing Services/APIs:**</u>

- Microsoft Cognitive Services - Face API
- Califai - Object Detection

5. IMPLEMENTATION:

Implementation of Sever using Spark and Storm:

Used OpenIMAJ (which has an inbuilt face detection and recognition algorithms) at the server side to detect the number of faces present per frame and also compare the detected faces with the list of images present in the training data, recognise them.

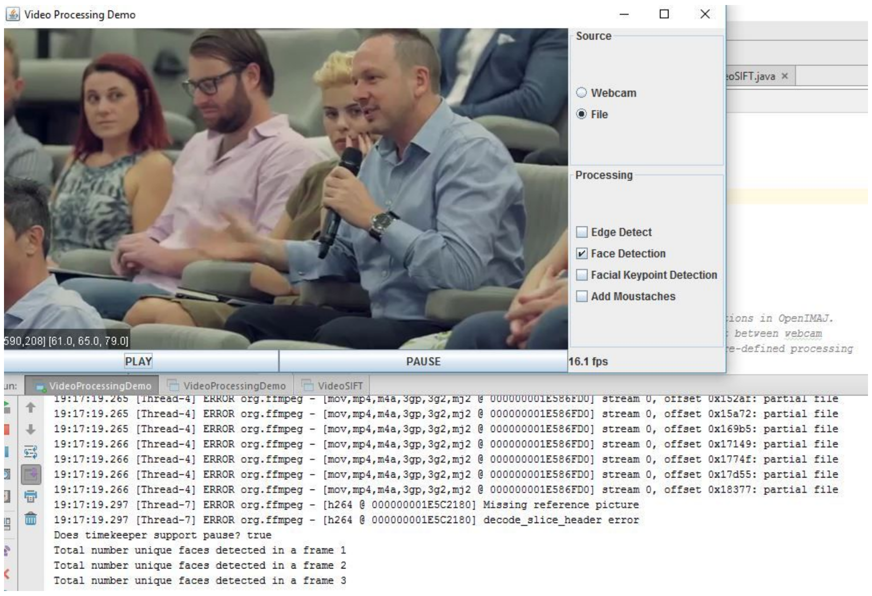
Implementation of Mobile Client:

The Client UI is developed using the HTML, CSS, Angular JS and Java Script, which provides the user with a option to record a live video to know the number attendees and recognise each individual person.

6. DOCUMENTATION:

RESULTS USING OPENIMAJ:

1. Static Classroom Video input - 3 faces detecting showing number of students count:

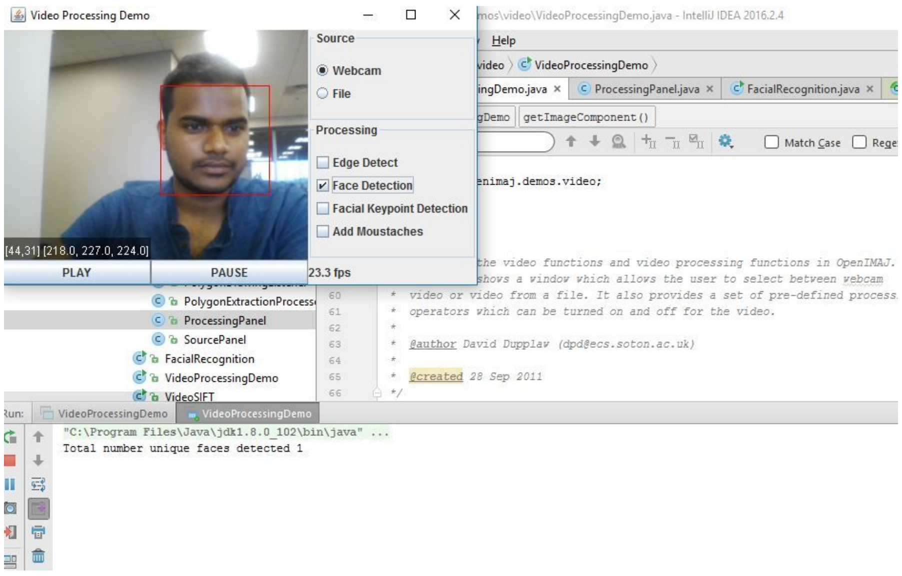


Source: [✓ File]

Processing: [✓ Face Detection]

```
19:17:19.265 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x152af: partial file
19:17:19.265 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x15a72: partial file
19:17:19.265 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x169b5: partial file
19:17:19.266 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x17149: partial file
19:17:19.266 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x1774f: partial file
19:17:19.266 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x17d55: partial file
19:17:19.266 [Thread-4] ERROR org.ffmpeg - [mov,mp4,m4a,3gp,3g2,mj2 @ 000000001E586FD0] stream 0, offset 0x18377: partial file
19:17:19.297 [Thread-7] ERROR org.ffmpeg - [h264 @ 000000001E5C2180] Missing reference picture
19:17:19.297 [Thread-7] ERROR org.ffmpeg - [h264 @ 000000001E5C2180] decode_slice_header error
Does timekeeper support pause? true
Total number unique faces detected in a frame 1
Total number unique faces detected in a frame 2
Total number unique faces detected in a frame 3
```

2. Live input using WebCam showing One face detected:


Video Processing Demo
mos\video\VideoProcessingDemo.java - IntelliJ IDEA 2016.2.4
Help
video 〉 VideoProcessingDemo 〉
ingDemo.java × ProcessingPanel.java × FacialRecognition.java ×
gDemo getImageComponent()
Match Case
Rege
PLAY
PAUSE
PolygonExtractionProcess
ProcessingPanel
SourcePanel
FacialRecognition
VideoProcessingDemo
VideoSIFT
Run: VideoProcessingDemo VideoProcessingDemo

Source
- [✓ Webcam]
- File

Processing
- [ ] Edge Detect
- [✓] Face Detection
- [ ] Facial Keypoint Detection
- [ ] Add Moustaches

[44,31] [218.0, 227.0, 224.0]

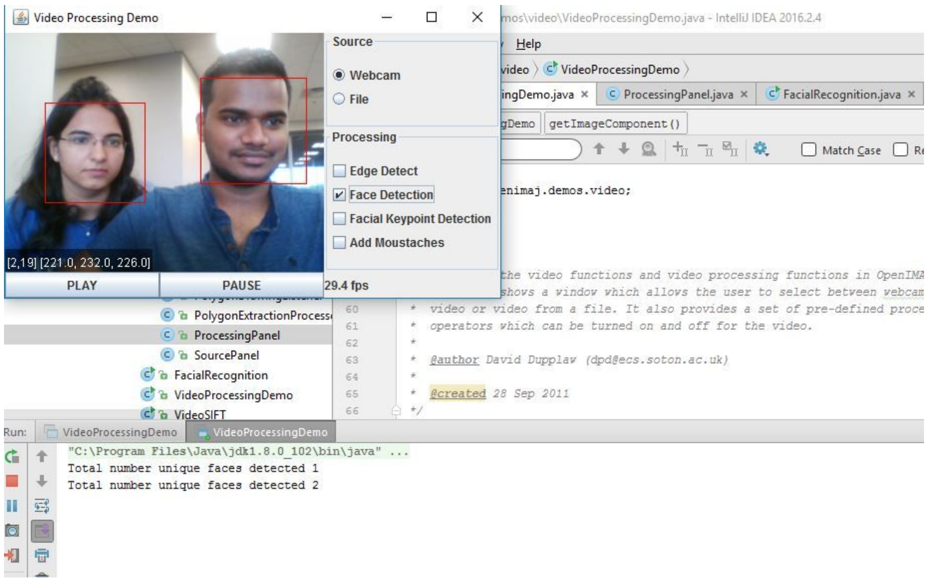
23.3 fps

```
enimaj.demos.video;

the video functions and video processing functions in OpenIMAJ.
shows a window which allows the user to select between webcam
 * video or video from a file. It also provides a set of pre-defined process
 * operators which can be turned on and off for the video.
 *
 * @author David Dupplaw (dpd@ecs.soton.ac.uk)
 *
 * @created 28 Sep 2011
 */
```

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Total number unique faces detected 1
```

3. Live WebCam input showing Two faces detected:


Video Processing Demo
mos\video\VideoProcessingDemo.java - IntelliJ IDEA 2016.2.4
Help
video 〉 VideoProcessingDemo 〉
ingDemo.java × ProcessingPanel.java × FacialRecognition.java ×
gDemo getImageComponent()
Match Case
Re
PLAY
PAUSE
PolygonExtractionProcess
ProcessingPanel
SourcePanel
FacialRecognition
VideoProcessingDemo
VideoSIFT
Run: VideoProcessingDemo VideoProcessingDemo

Source
- [✓ Webcam]
- File

Processing
- [ ] Edge Detect
- [✓] Face Detection
- [ ] Facial Keypoint Detection
- [ ] Add Moustaches

[2,19] [221.0, 232.0, 226.0]

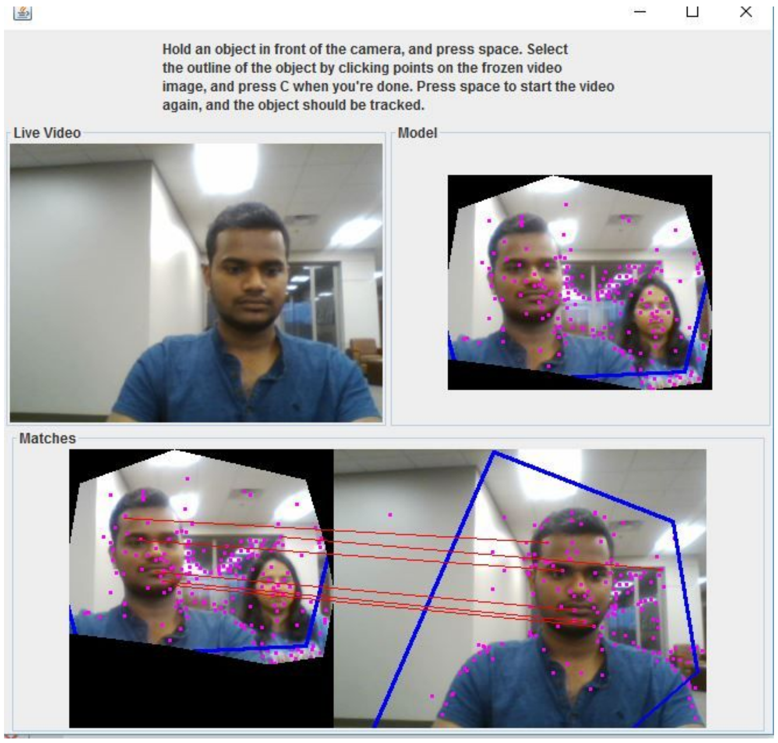
29.4 fps

```
enimaj.demos.video;

the video functions and video processing functions in OpenIMA
shows a window which allows the user to select between webcam
 * video or video from a file. It also provides a set of pre-defined proce
 * operators which can be turned on and off for the video.
 *
 * @author David Dupplaw (dpd@ecs.soton.ac.uk)
 *
 * @created 28 Sep 2011
 */
```

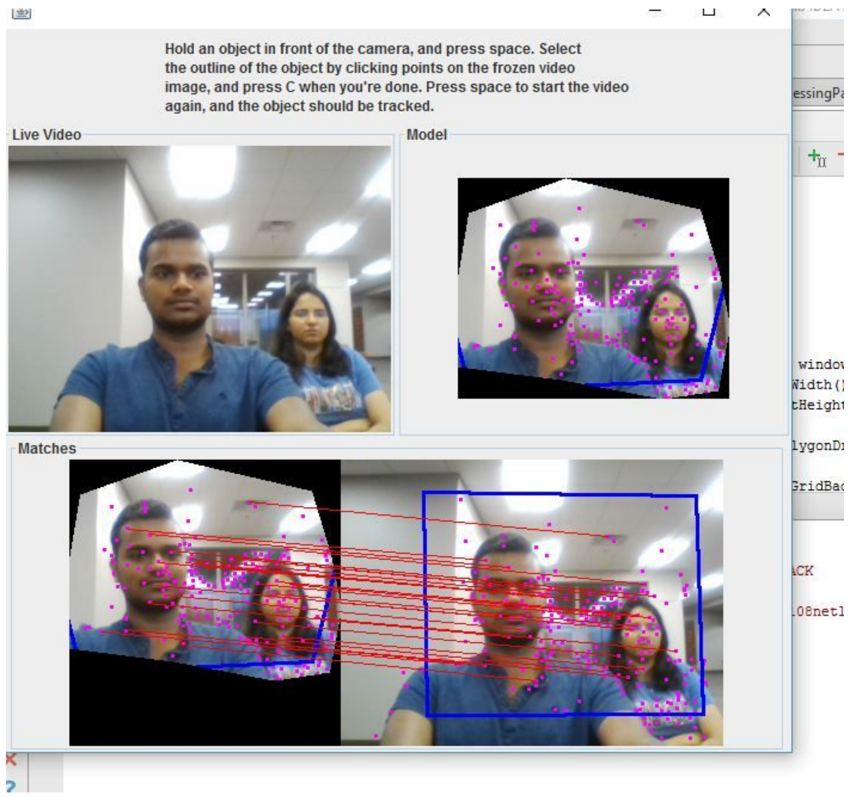
```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Total number unique faces detected 1
Total number unique faces detected 2
```

4. Live video input showing Face Recognition (Recognizing students facial features and matching them using SIFT). One match when one student present in class.



Hold an object in front of the camera, and press space. Select the outline of the object by clicking points on the frozen video image, and press C when you're done. Press space to start the video again, and the object should be tracked.

4. Live video input showing Face Recognition (Recognizing students facial features and matching them using SIFT). Two matches when two student present in class.



Hold an object in front of the camera, and press space. Select the outline of the object by clicking points on the frozen video image, and press C when you're done. Press space to start the video again, and the object should be tracked.

7. PROJECT MANAGEMENT:

Work Completed:

- Description:
    1. Successfully implemented Face identification and have count of number of students in the class using OpenIMAJ
    2. Used Microsoft Face API and Clarifai API to classify human faces and objects in a input image.
    3. Implemented Face recognition using OpenIMAJ by generating a model by selecting faces of the students.
- Responsibility:

  For this phase of project, the team collectively contributed for all the above implementations.
- Time Taken: 4 weeks
- Contributions:

  Nikhita Sharma - 25%
  Ganesh Taduri - 25%
  Aparna Pamidi - 25%
  Harsha Komalla - 25%

Work to be completed:

- Description:
    1. To generate a better model for Face recognition of students by using individual student images as training data.
    2. Use the new model to process real time video and mark a student as present, if recognized by the system.
    3. Design User Interface for Professor/Presenter and Students.
       - Professors would have functions of scanning the class, check a student's attendance history, attendance percentage, etc.
       - Students could check their daily attendance and attendance history, percentage, etc.
    4. Use Storm for accepting streaming video input and machine learning.
- Responsibility:

  We plan to share the responsibilities dynamically according to the workload.
- Time to be taken: 10 weeks

Issues/Concerns:

- While working with Microsoft Face API, most of the documentation of implementation was in C# or Python, which made it difficult to implement our modules in Java. Though we recieved some suggestions from Open Source support sites such as StackOverflow, we still had issues with the API keys.
- Most features provided by Clarifai API were not relevant to our project requirements. They were more generic to objects and not specific to Face Detection.

- Still working on the logic to generate a better model for classroom scenario.

  Project GitHub repository:
  https://github.com/nikhitasharma/KDM_Summer_2016_Cognitos

Future Work:

- Addition of additional features such as surveillance system for exit and entry of students in the class at various points of time during the class
- This project can also be extended to surveillance systems for public meetings or gatherings.

REFERENCES:

[1]. http://openimaj.org/openimaj-image/faces/dependencies.html
[2]. http://openimaj.org/tutorial/sift-and-feature-matching.html
[3]. http://openimaj.org/tutorial/eigenfaces.html
[4]. http://klresearch.org/IJMSTM/papers/v2i3_2.pdf
[5]. http://blog.mashape.com/list-of-10-face-detection-recognition-apis/
[6]. http://ieeexplore.ieee.org.proxy.library.umkc.edu/stamp/stamp.jsp?tp=&arnumber=6227479