iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.42229

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

Exam Proctoring System

Chaitanya Thombare[1], Kushank Sapate[2], Aniket Rane[3], Ankush Hutke[4]
[1, 2, 3, 4]*Dept. of Information Technology, Rajiv Gandhi Institute of Technology, Mumbai*

Abstract: Due to the rise of Coronavirus Pandemic, the majority of people were forced to follow a lot of protocols from masks and social distancing to online presence at work places. One of those places are educational institutions. The lectures could be easily conducted using online video conferences, but exams are very difficult to be conducted online. And the part that makes it hard is invigilation of those exams. Invigilation is a very important part of any examination. Even though it is evident that examinations are for the candidates to evaluate their performance, they still try to cheat to get good results which may get them benefits they do not deserve. We propose a system that proctors or invigilates students while giving online exams. The system collects candidates' behavior projected on the camera at browsers and extracts information from those images, like 'looking left', 'missing test' etc. and logs them to a SQL database. The system also has a threshold for fouls candidate commits and would submit the exam when threshold is crossed.

Keywords: Proctoring System, Head Pose Estimation, behaviour logging, Online Proctoring System, Anti-cheat System

I. INTRODUCTION

Digitization has affected mankind in different ways, from education, banking to a lot of things. Even though digitization has been around for a long while around the world, due to some reasons, the involvement of technology was not seen as much as it is now. The coronavirus pandemic forced most of the people to stay at their homes and put the whole world to a stop.

This is devastating to the whole system as work wasn't done, causing less revenue generation in the companies which in turn affected all the dues, be it salary, be it taxes. Hence to still continue with all the functioning, people were forced to go digital and work online from home.

Just like the whole world, students were also forced to come online. The courses and assignments submissions were easily achieved using video conferencing and file sharing platforms. But one major problem faced was an online exam. Online exam is difficult to conduct because it took a lot of control from the invigilators and gave some freedom to candidates. Candidates could do wrong things and go unnoticed. The only way invigilators had to was to hold video conferencing and check all student's video feeds, which is a very inconvenient and unreliable way of invigilation.

To overcome this problem, we have come up with a proctoring system, which is an automated approach to make sure students follow the rules. The solution does not stop students from doing anything wrong, which is impossible for an online system, but the system will observe everyone all the time and make sure no actions go unnoticed. The system would also submit the candidate's exams if they were found to cheat more often.

II. RELATED WORK

The paper [5] discusses using a 360-degree camera for proctoring. This is a really a great idea and really solves the problem of proctoring beyond computer vision, because this camera would cover everything in the surrounding. This is really a great approach, but 360-degree cameras as of now are expensive and not everyone can afford it. Our objective is to give a system that works on commonly existing hardware, and provide the best service in those parameters. A 360-degree camera for proctoring can be a more accurate approach for proctoring.

III.PROPOSED SYSTEM

There were two approaches we could use to solve this problem, first is having the proctoring model on client side, and other is having the proctoring model on the server side. The client-side approach would send the model to individual devices giving exams with the browser pages sent to them. These pages would collect the required data and extract information out of it and send this information to the backend server. The server-side approach would send browser pages to the device, which can only collect the data. The pages would send the raw data to the server, the server having the proctoring model that would extract information from the images. To choose the client-side approach, following factors are considered: -

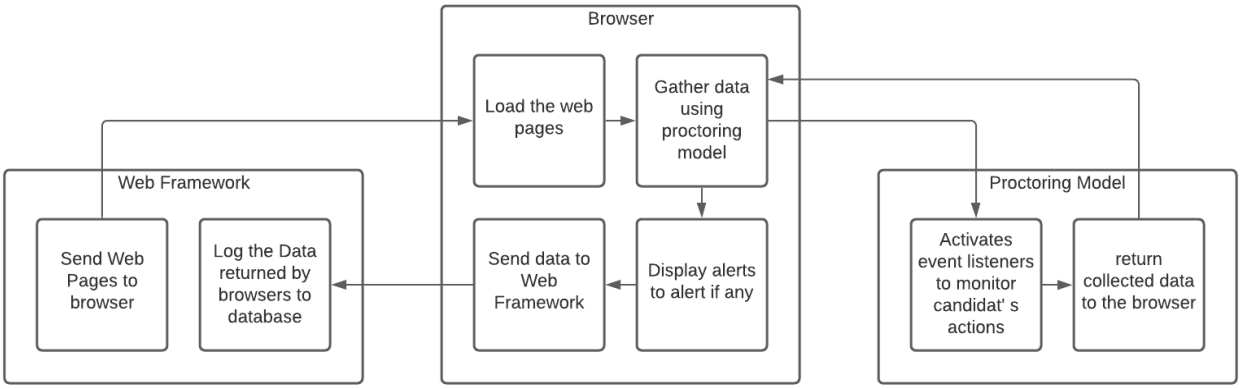
A. Bandwidth Requirement

The Client approach requires a small bandwidth as it has only a small amount of data that goes back and forth from server to web pages. This data is the extracted information from the raw data. Raw data would be the images and information extracted would be if a person is looking left, missing the test, etc. The size of information would be tremendously lower than the raw image data the web pages collect.

B. Volume of user Supported

The client approach would send a proctoring model to individual devices in the browser pages. So, when these pages load, every device would proctor their user. In short, every device that is involved would do the processing. In the server-side approach, the server would have to do the computation for all the images coming from all the devices. This becomes a tremendous number of images to process which would take a lot of time and huge processing power. Since the server also has some limitations, those limitations would be easily reached in few candidates using the system in server-side approach. But in the client-side approach, since a very small amount of data is sent, it would take a lot of candidates to make the server reach its limits.

System Architecture



.

Fig 1 System Architecture

1) *Flask Server:* Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies and several common framework related tools. In this system what we do is make the predictions at web page level. The predictions are done by the JavaScript part of the front-end. The JS would use the Media Pipe model to estimate head pose and send data as required. This would reduce the amount of data sent back and forth from web pages to server by a huge factor in comparison to the previous method. Next, it would also reduce a lot of loads of processing on the server as every device giving the exam would do the processing.
2) *Browser:* A web browser (commonly referred to as a browser) is application software for displaying web pages. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device. Web browsers commonly include an address bar or search bar, the ability to open multiple web pages in different tabs, and other user interface features for navigating the web. This system uses browsers as an interface for students to give exams. The web pages sent would request camera access and record everything via webcam. Further, the webpages would process those images using the proctoring model and extract information from those images. The web pages also have some event listeners to detect change in tabs, etc. and collect that information too.
3) *Proctoring Model:* The proctoring model is built using Media Pipe. Media Pipe is an open-source model for facial landmark detection. This model detects the 468 facial landmarks in 3D space. So, to estimate head pose, what we do is calculate the angle between the centre of head and the nose tip. Centre of the head would be the point in between the ears. Nose tip and Ears are some of those landmarks.

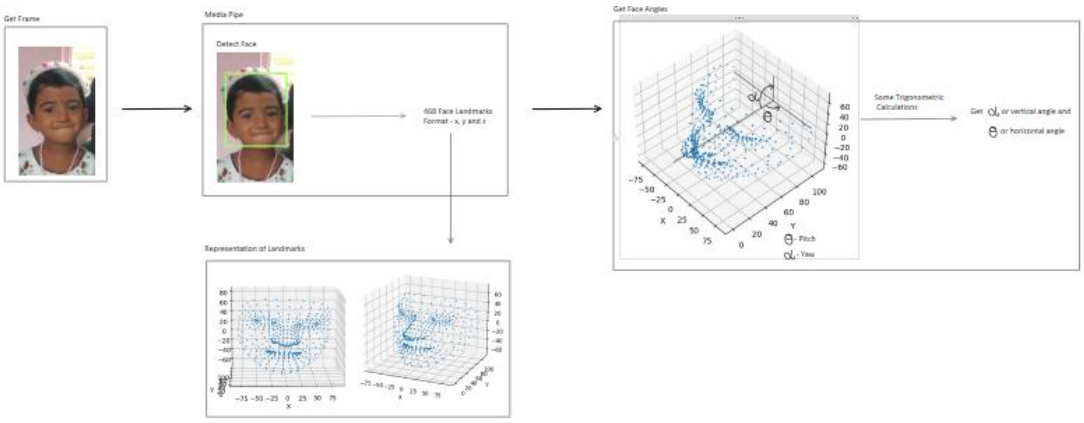


Fig 2 Face Mesh Model

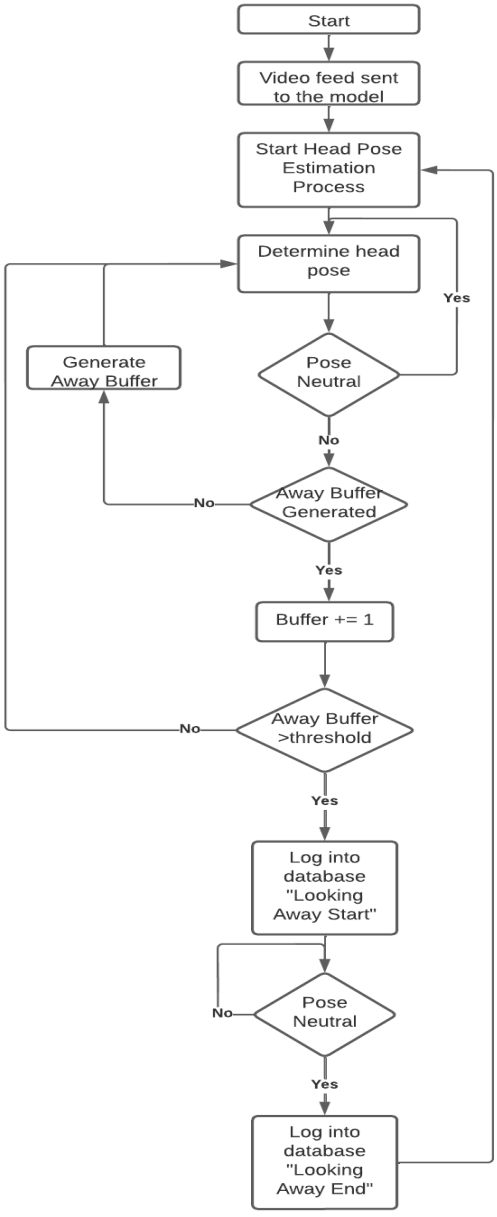


Fig 3 Head Pose Flow Chart

If the candidate tilts their head horizontally or vertically, beyond the defined threshold, a buffer would be started. The next time, if the angle is within threshold, then buffer is cancelled and system continues its further processes. If the angle is still beyond threshold, 1 would be added to buffer. This way if buffer fills up to 100 with takes about 2 seconds, then webpages send this data to the flask server that would be logged in the database. Now when the candidate looks back to the screen, this would again be logged in the database. So, in the database, we would see an 'looking away start' and 'looking away end' with proper time stamps. Same procedure is followed for the looking up or down

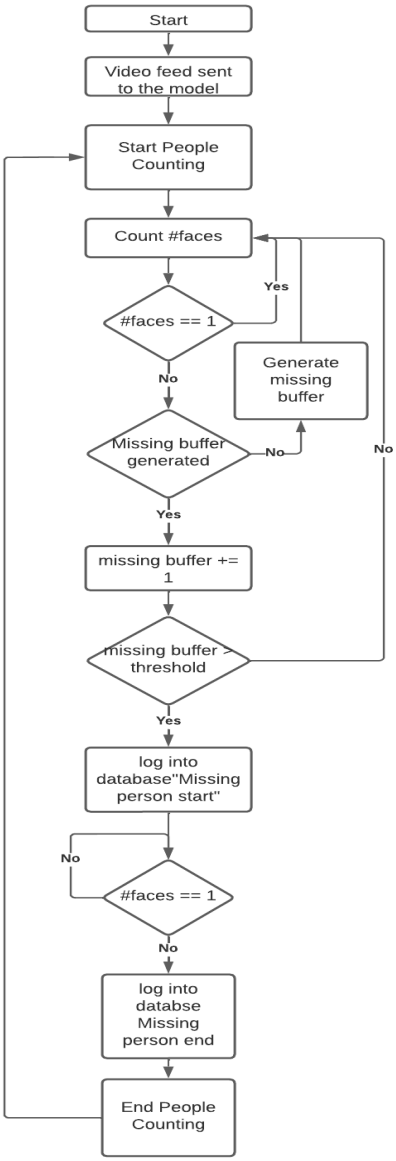


Fig 4 Missing Person detection model flow chart

If the candidate goes away from the webcam, a buffer would be started. The next time, if the candidate is back on the camera, then buffer is cancelled and system continues its further processes. If the candidate has not come yet, 1 would be added to buffer. This way if buffer fills up to 100, which takes about 2 seconds, then webpages send this data to the flask server. Now when the candidate looks back to the screen, this would again be logged in the database, which would be logged in the database. So, in the database, we would see an 'missing person start' and 'missing person end' with proper time stamps.

Event listeners are the procedure or function that waits for an event occur. This event can include click, hover, button press etc. The event listeners are usually a called with a function. The function may be predefined and defined while calling, and would be called when the event which triggers that event handler occurs

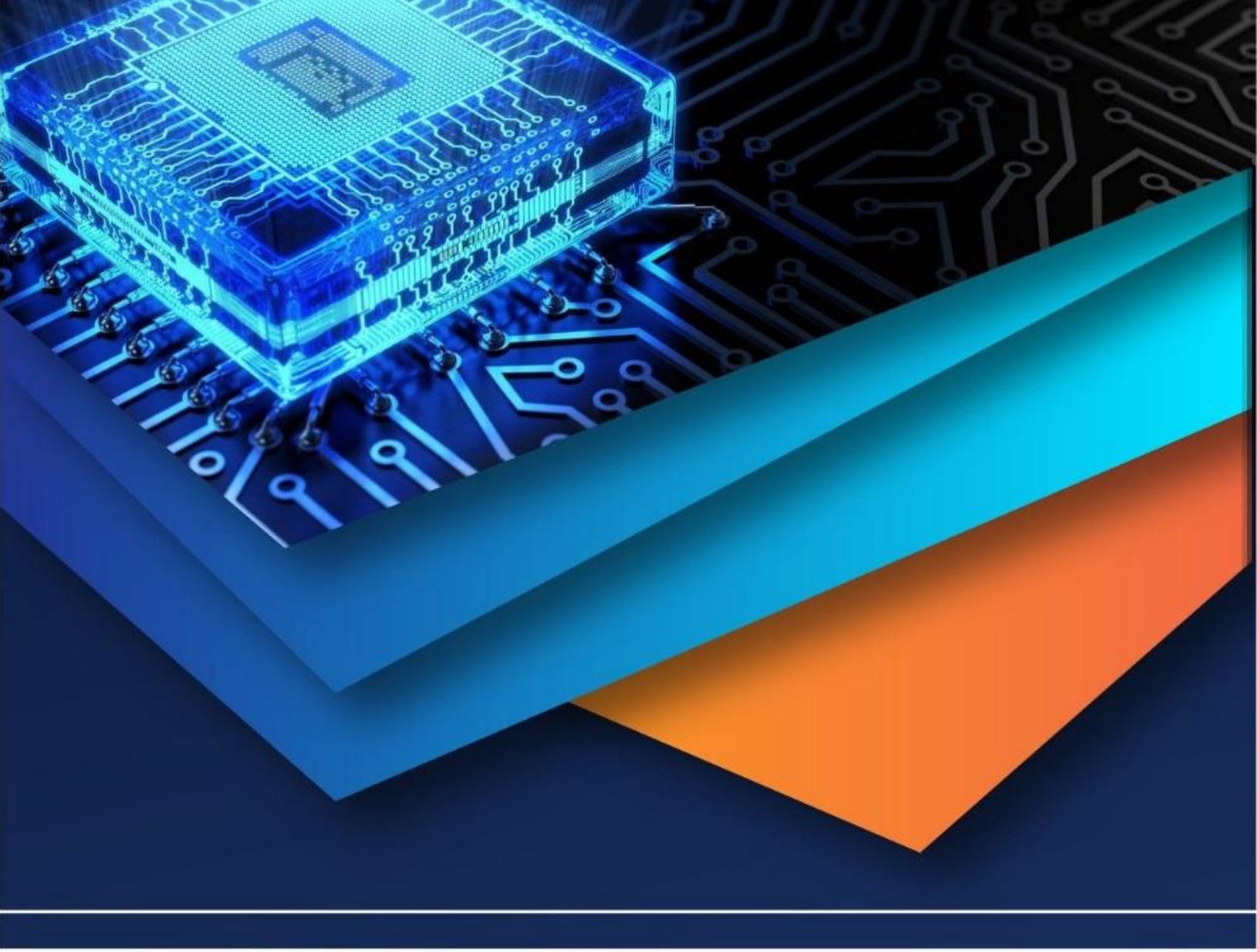
For proctoring, we have used tab change event listeners. If the candidate changes their exam tab and shifts to any new tab the event listeners would be triggered the function would send this data to flask server. The server would log 'tab change invisible' in the database. When the user comes back to the test page tab, again the event listener would be triggered and this time again it would send data to the flask server, which would log 'tab change visible' to the database.

IV. CONCLUSIONS

The whole model is managed by Flask. Starting from sending web pages to the client browser. These web pages are an interface for candidates to give their exam and allow the proctoring system to monitor the student behavior on camera and on browser (changing tabs and keys pressed). The detected behavior is recorded in the form of data (Looking Away Start and Looking Away End, Person Missing Start and Person Missing End) and sent to the Flask Server. The received data is logged into the database logging every action the candidate makes.

REFERENCES

[1] Praveen Kumar S , Kesava Jayendra Varma V, Subramanya V, Venkata Sai Harish A. 'A multiple face recognition system with dlib's resnet network using deep metric learning' . JCR. 2020; 7(6): 856-859. doi:10.31838/jcr.07.06.147
[2] T. Yang, Y. Chen, Y. Lin and Y. Chuang, "FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation From a Single Image," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1087-1096, doi: 10.1109/CVPR.2019.00118.
[3] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays and Fan Zhang, 'MediaPipe: A Framework for Building Perception Pipelines', (available at https://arxiv.org/abs/1906.08172)
[4] Ivan Grishchenko, Artsiom Ablavatski, Yury Kartynnik, Karthik Raveendran and Matthias Grundmann, 'Attention Mesh: High-Fidelity Face Mesh prediction in real-time', (available at https://arxiv.org/abs/2006.10962)
[5] A. A. Turani, J. H. Alkhateeb and A. A. Al Sewari, "Students Online Exam Proctoring: A Case Study Using 360 Degree Security Cameras," 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), 2020, pp. 1-5, doi: 10.1109/ETCCE51779.2020.9350872.
[6] E. Murphy-Chutorian and M. M. Trivedi, "Head Pose Estimation in Computer Vision: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, pp. 607-626, April 2009, doi: 10.1109/TPAMI.2008.106.
[7] Sanchit Mishra, Bhavin Patanwadia, Vineet Yadav, Shivam Kumar, Kalpesh Kubal and Shalmali Joshi, 'Online Examination System', International Research Journal of Engineering and Technology, 6(1), 2019, 1534-1535
[8] Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu and X. Liu, "Automated Online Exam Proctoring," in IEEE Transactions on Multimedia, vol. 19, no. 7, pp. 1609-1624, July 2017, doi: 10.1109/TMM.2017.2656064.
[9] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3D facial landmarks). In International Conference on Computer Vision, 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY