Multiparameter-based evaluation to identify the trustworthiness of the examinee during e-assessments

J. A. P. H. Perera

2023

Multiparameter-based evaluation to identify the trustworthiness of the examinee during e-assessments

A dissertation submitted for the Degree of Master of Science in Computer Science

J. A. P. H. Perera
University of Colombo School of Computing
2023

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: J. A. P. H. Perera

Registration Number: 2018/MCS/065

Index Number: 18440652

____

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. /Ms. ________________________ under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name:

____

Signature of the Supervisor & Date

ACKNOWLEDGEMENTS

Even though the cover of my dissertation just has my name on it, a large number of people helped in its development. I am grateful to everyone who contributed to the completion of this dissertation and made my graduate experience one I will never forget.

I'd like to begin by thanking my supervisor, Prof. K.P. Hewagamage, for his continuous guidance and support during this study. Without his great guidance, all of my efforts could have been worthless.

I'd also like to thank my family for their constant support and motivation as I work toward my goal.

Finally, I recognize that this research would not have been possible without the inspiration and assistance of each lecture at the University of Colombo School of Computing, who supported us whenever we needed it.

ABSTRACT

Monitoring and confirming the examinee's degree of trust during online exams without the assistance of live proctors or invigilators has become a significant concern for most Universities in Sri Lanka. There are currently feasible proctoring options for institutions in Sri Lanka, but they have flaws that make it possible for students to cheat on exams. This study primarily intends to address that issue by introducing a methodology to determine the examinee's trustworthiness during the e-assessment period. As a result, it is critical to comprehend the ideas of trust and trustworthiness in the context of assessment and there are numerous definitions of trust and trustworthiness in the literature. It is essential to highlight that the concept of trust and trustworthiness is context-sensitive, and several metrics are presented in the literature to assess the examinee's identification and behavior-related trustworthiness. Biometric parameters are emphasized in the literature as important factors that are employed for the examinee's identity verification or authentication and there are other non-biometric parameters as well.

The term "examinee trustworthiness" as defined in this study refers to the state that an examinee achieves by adhering to the established rules throughout the e-assessment time. Also, researchers have further defined the “trustworthiness parameter” as any action, occurrence, or environmental change that influences the variance in the examinee's level of trust during the online assessment time. In the context of an online e-assessment, this research attempts to construct a multiparameter-based model to measure the examinee's trustworthiness.

Using discrete image stream, and operating system event data collected during the online e-assessment, the researcher has conducted experiments and literature analysis in the first stage of this study to identify potential trustworthiness parameters. Through these experiments, the researcher has identified the examinee's environment's unacceptable background, the examinee's lack of visibility in front of the webcam, the presence of multiple people in front of the webcam, the examinee's false identity, and accessing unauthorized materials on the operating system as trustworthiness parameters in this study.

Through the parameters used during phase one of this research, the researcher defined the qualitative concept of "examinee trustworthiness" in the context of online e-assessment in a quantitative way. The researcher also determined the proper weights of each chosen parameter to the examinee's trustworthiness because the identified parameters affect the examinee's trustworthiness to varying degrees. The researcher presented the novel concept of the “Trust Index”, which is a quantitative representation of examinee trustworthiness, in an online e-assessment context as an intergraded output of selected parameters as a multiparameter base model to evaluate the examinee trustworthiness.

With the help of tooling the researcher had developed to capture each parameter of trustworthiness, the researcher qualitatively assessed the effectiveness of the proposed Trust Model using a simulated e-assessment with 15 participants. The researcher makes many recommendations for tools and the trust model that has been put in place for future research.

LIST OF PUBLICATIONS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CNN	**Convolutional Neural Network**
ML	**Machine Learning**
E-Learning	**Electronic Learning**
CPU	**Central Processing Unit**
LMS	**Learning Management System**

CHAPTER 1

1 INTRODUCTION

1.1 MOTIVATION

With the digital transformation, traditional educational systems were gradually replaced with e-learning systems. However, the Covid-19 epidemic has acted as a stimulus for this digital transformation in the educational sector. According to the statistics, globally over 1.2 billion children were unable to attend classroom and education systems, and educators were forced to shift to online methods rapidly ('The COVID-19 pandemic has changed education forever, 2020). When it comes to the local Sri Lankan context, most educational institutions, schools, and universities have started to move from traditional classroom-based approaches to online-based learning approaches to support students with their educational activities during the Covid-19 pandemic.

Assessment is critical in any educational system since it decides whether or not educational goals are reached ('Why Is Assessment Important?', 2008). However, with the Covid-19 pandemic, people were forced to maintain a social distance ('Coronavirus, Social and Physical Distancing and Self-Quarantine', 2020) and because of that educational institutions were forced to move to E-assessments as an alternative to traditional paper-based assessments. Unlike the traditional classroom and paper-based assessments, researchers have identified several issues that are unique to e-assessment such as

- Maintain fairness among all the students
- Methods that are unbiased for evaluating knowledge
- The degree to which kids can use technology
- Plagiarism
- The potential for internet fraud and impersonation ('Online Assesment in Schools-Challenges and Solutions', 2020)

Out of these points, online cheating during the e-assessment has been identified as a common challenge for all E-learning systems (Bawarith *et al.*, 2017). Hence measuring the trustworthiness of the examinee during and identifying the e-assessment has also become a challenge as well. Also, the magnitude of this challenge has increased with the Covid-19 pandemic.

1.2 PROBLEM STATEMENT

Without the involvement of live proctors or invigilators, monitoring and validating the trust level of the examinee during online examinations has become a critical challenge for most Universities in Sri Lanka. Though there are claimed advanced, automated proctoring systems available for commercial purposes, they are not affordable for universities in Sri Lanka due to financial restrictions. Since human resources are also limited, live proctoring also has many limitations in practice. Proctors, for instance, should be educated to check for authenticities and odd behaviors like eye or facial gestures and the appearance of any untrustworthy equipment that would suggest possible cheating. Additionally, it demands that the exams be planned at a specified time based on the proctor's availability at a given date and time and their technological proficiency (Foster and Layman, 2013).

The available current solutions which are feasible for proctoring for universities in Sri Lanka contain loopholes that enable the possibility of cheating during the examinations. As an example zoom proctoring with a live proctor is given several students to monitor in a single view which causes a limited focus in a single student, sometimes due to bad lighting conditions in the surrounding environment, the captured images are not clear enough to get a decision and sometimes unable to identify the student eye focus during the zoom proctoring as well.

As per the examples stated above, validating the trustworthiness of the examinee during an e-assessment has become a challenge and this reflects the importance of designing a mechanism to evaluate the trustworthiness of students in such an e-assessment context.

1.3 SIGNIFICANCE OF THE STUDY

Online assessments are now an integral part of our lives in a world where everything is being digitized. Exams administered online are becoming more prevalent as we adapt to the digital world, slowly replacing the old tradition of pen-and-paper testing. The primary benefit of an online examination system is the reduction of expenses and time for both the candidate and the evaluator. With the expanding usage of online assessment, the level of fraud committed has also increased significantly (Joshy *et al.*, 2018).

To benefit from this technological innovation and uphold academic integrity, it is crucial to design a technology to evaluate the examinee's trustworthiness during e-assessments. As a result, the primary aim of this research is to build a computational model that will quantify an examinee's trustworthiness during summative e-assessments and serve as a decision-supporting indication for evaluators when conducting mass e-assessments.

In the literature, trust and trustworthiness terms are defined in a variety of ways, and in this study, it will be expanded upon more in the literature review section. One crucial insight is that the concept of trust and trustworthiness is context-sensitive (Liu and Wu, 2010) and therefore, it is essential to define the trust and trustworthiness teams under consideration for this study.

In this study, the researcher has defined the term **examinee trust** in the context of e-assessments as adhering to the specified rules throughout the e-assessment period. From the perspective of the evaluator, if a person agrees and follows a set of rules, it is reasonable to conclude that the individual is upholding the rules. As a result, we may conclude that person is trustable. If the person violates the rules, he becomes a suspect in the e-assessment context. If the examinee performs any activity or any suspected activity which violates the defined e-assessment rules, then we can call that action a breach of trust and if the examinee performed the e-assessment adhering to the rules then we can call that action a maintained trust.

Also, the researcher has defined the term **examinee trustworthiness** in the context of e-assessment as the state examinee achieves by adhering to the specified rules throughout the e-assessment period. If the examinee performs the e-assessment without breaching the trust, we call him a trustworthy examinee. If he breaches the trust, we call him an untrustworthy examinee. Examinee trustworthiness is a qualitative concept that we can define within a time boundary.

The researcher has also defined the team **trustworthiness parameter** as any action, occurrence, or environmental change that influences the variance in the examinee's level of trust throughout the online assessment session.

In conclusion, educational institutions that have limited resources will benefit from the quantification of the examinee's trustworthiness during the e-assessment. Additionally, it will improve academic integrity and assist evaluators in their decision-making process in e-assessments.

1.4 RESEARCH QUESTION(S)

Develop a model for examinee trustworthiness that can be used to filter examinees of online e-assessment based on the given context and ruleset

- Analyze the discrete image stream of the examinee during the e-assessment to identify the most suitable trustworthiness evaluation parameters
- Analyze the operating system events of the examinee’s computer to identify the most suitable trustworthiness evaluation parameters

Integrate the outcome of the identified parameters to determine the trustworthiness of the examinee during an online e-assessment

1.5 RESEARCH AIMS AND OBJECTIVES

1.5.1 AIM

This study aims to develop a multiparameter-based model to measure the examinee's trustworthiness in an online e-assessment context. Based on the created model, the researcher will present a method for evaluating the examinee's trustworthiness in the context of an online e-assessment using a novel unit of measurement called the Trust Index, which is a quantitative representation of the examinee's trustworthiness.

1.5.2 OBJECTIVES

- To describe factors that should be considered to assess the examinee's trustworthiness in an online e-assessment context
- To develop an initial proctoring system to employ the proposed model to be used in online assessments
- To propose a model for assessing trustworthiness as a single unit of measurement during an e-assessment
- To evaluate the suitability of the proposed model in an e-assessment context

1.6 SCOPE

With the Covid-19 pandemic and social distance boundaries, to support schools and university students, most universities moved to online-based educational activities. Even though the university academic activities are carried out on an online basis, some of the universities postponed the examinations and some of them have conducted the examinations by maintaining the social distance including the University of Colombo School of Computing (UCSC) as well as attempts have been made to conduct the examinations online basis with the zoom real-time proctoring.

Hence, this research will primarily focus is to develop a model for the trustworthiness of the examinee during online an e-assessment context based on the parameters identified through the discreet image collection and operating system events captured during the e-assessment.

Suitable parameters will be selected based on the experiments and the literature review and prototype system will be implemented by using the identified parameters. This prototype system will be used for the data collection and model evaluation process.

When it comes to the target audience, this study will be conducted targeting postgraduate/undergraduate students and academic staff at the University of Colombo School of Computing as a supportive mechanism to conduct e-assessments with the help of automated proctoring. Also, this will be an individual research project as a submission for the MCS 3003 course which takes one year.

1.7 APPROACH TO THE PROBLEM

The first phase of this research will take a qualitative approach where the researcher will perform experiments and a literature review to identify possible trustworthiness parameters. Additionally, the researcher will implement software tools to capture these trustworthiness parameters in an optimum way. Using those implemented tools experimental data will be collected related to possible different parameters and qualitative analysis will be performed to identify the most feasible parameters for the study in the given context.

The second phase of this research will take a qualitative approach where the researcher is planning to quantitatively define the qualitative concept of examinee trustworthiness in the context of online e-assessment through the parameters identified during phase one of this research. Since the identified parameters contribute to the examinee's trustworthiness to different degrees, the researcher must identify the appropriate weight of each selected parameter to the examinee's trustworthiness (Raj, Narayanan and Bijlani, 2015). In this stage, the researcher aims to introduce the Trust Index, which is a quantitative representation of examinee trustworthiness, in an online e-assessment context as an intergraded output of selected parameters for the study to develop a multiparameter base model to evaluate the examinee trustworthiness.

A prototype system will be implemented, and simulation-based experiments will be conducted with uses to finetune the implemented model. During these experiments, students will be asked to simulate the cheating scenarios and trustable scenarios, and the Trust-Index output received by the model will be qualitatively analyzed in the study. Based on this researcher will introduce a Trust-Index Metrix which is a proposed scale of examinee trustworthiness that can be used in a similar online e-assessment context.

1.8 EXPECTED RESEARCH CONTRIBUTION

- This research will be a novel approach and attempt for the objective evaluation of examinee behavior during an online e-assessment based on the given context and ruleset. In this proposed approach, possible suspected activities will be monitored throughout the e-assessment process from the input received via a discreet collection of images/video stream, and operating system events.
- Trust index will be generated as the outcome of the proposed automated proctoring system and this proctoring system will be an initial framework for more cost-effective integrations as well.

CHAPTER 2

2 LITERATURE REVIEW

2.1 E-LEARNING & E-ASSESSMENT

With the digital transformation in the educational domain, e-learning has gained popularity, especially in higher educational institutions almost everywhere in the world. Musa and Othman define e-learning as all forms of learning delivered electronically that aid the teaching and learning processes (Musa and Othman, 2012). The scope of e-learning can have a vast range based on the involved practice, such as online learning where the learning process is based on digital media, blended learning where e-learning consists of both online as well as offline learning, ICT-mediated face-to-face learning where learning and teaching will be face to face based on online media and distance learning (Gikandi, Morrow and Davis, 2011)

With the introduction of E-learning systems, E-assessments also emerged as a digital way to conduct an assessment to measure learner outcomes. According to Gaytan (Gaytan, 2004), e-assessment is a system for assessing students' academic success that consists of numerous components that must be measured. Another study conducted by Alruwais, Wills, and Wald (Alruwais, Wills and Wald, 2018) explains E-assessment as a set of procedures based on the electronic form where start to end including designing, test implementation, and responding are conducted using ICT. In general, E-assessment can be defined as a technique to measure the learner's learning outcomes and performance with the support of digital technology. Like traditional assessments, E-assessments also can be categorized under the classification of the formative and summative assessment approach. Conducting formative assessments is comparatively easy in an online environment by using the facilities provided by the Learning Management systems (LMS) because formative assessments are conducted as a continuous learning activity during the course (Hewagamage and Wikramanayake, 2011). However, conducting a summative assessment in an online environment is different from conducting a formative assessment because summative assessments require high control and security measurements in the assessment process to ensure reliability and validity (Cassady and Gridley, 2005). Therefore, in most academic courses, summative assessments are conducted using the classroom-based approach and formative assessments are conducted using the help of available e-assessment methods in LMS.

In both classroom and e-assessment approaches, formative assessments are assessments defined to support learning (Gikandi, Morrow and Davis, 2011). The primary focus of them is to improve the student's performance and is not intended to audit it (Dixson and Worrell, 2016) and therefore the trustworthiness of the student is assumed during the assessment period. However, when it comes to summative assessment, is defined as validation and accreditation of the student leanings (Gikandi, Morrow and Davis, 2011). Summative evaluations are used to measure the effectiveness of the learning process and compare student performance to predetermined benchmarks (Dixson and Worrell, 2016) and therefore trust of the student during the assessment process is also validated, and measured with the help of invigilators.

2.2 TRUST AND TRUSTWORTHINESS

To gain a thorough understanding of trust and trustworthiness is crucial to continue this study. In general, there are many definitions of trust and trustworthiness that can be found in the literature. A paper published by Russell Hardin state that without any moral undertones, the trust may be entirely explicable as a skill or as the result of logical expectation (Hardin, 1996). Another study conducted by Margaret and Laura (Levi and Stoker, 2000) suggests, as a relational term, trust requires people to expose themselves to people, groups, or institutions that might injure them or betray them and the judgment of trust is inspired by the cause of actions. Furthermore, they explain that trustworthiness is a relational concept with a more limited sense and if a person has attributes of trustworthiness even though there is no call for trust, it will act as an assurance for potential trusters that the trusted party will not betray them (Levi and Stoker, 2000).

The concept of trust and trustworthiness is context-sensitive. To support this claim, Liu and Wu define trust as a relationship between the trustier and trustee in a certain situation and setting (Liu and Wu, 2010). They explain this with an example where the person can trust another person’s ability for computing but not the ability to write. Therefore, in the context related to computing, they can trust each other but not in the context of writing something.

Therefore in this study researcher must identify trustworthiness attributes in the e-assessment context so that they can be used to elaborate on the trustworthiness of the examinee during the e-assent. However, the question is, is it possible to measure trustworthiness in a given context, specifically in the computing domain? In the literature, the researcher has found several previous attempts to achieve this goal.

Marsh in his thesis on “formulating trust as a computational concept” (Marsh, 1994) introduces a formula for situational trust as,

$$\llbracket Tx\ (y,\alpha)\rrbracket\ \hat{}t = Ux\ \llbracket(\alpha)\rrbracket\ \hat{}\ t\ * Ix\ \llbracket(\alpha)\rrbracket\ \hat{}\ t\ * Tx\ \llbracket(y)\rrbracket\ \hat{}\ t$$

where α denotes the situation and $Ux\ \llbracket(\alpha)\rrbracket\ \hat{}\ t$ represents the utility x benefit from the situation α, $Ix\ \llbracket(\alpha)\rrbracket\ \hat{}\ t$ is the significance of the situation α for agent x and $Tx\ \llbracket(y)\rrbracket\ \hat{}t$ is the valuation of general trust after including all possibly relevant data concerning Tx (y,α)in the past. This study gives a clear indication to the researcher which shows the computational possibility of quantifying the trust concept related to a given situation.

Also, in the E-learning domain, the term trust has many definitions in terms of situation and context (Ben-Ner and Halldorsson, 2010). From a systems perspective, Liu and Wu mention that a reliable e-learning system is one that has reliable serving peers and effective learning resources (Liu and Wu, 2010). However, the focus of this study is on the examinee's trustworthiness during the assessment process, and, according to Ivanova et al., trust in the examinee's identity and trust in the submission of their original work are the two most important factors influencing that trust (Ivanova *et al.*, 2018).

When it comes to these two different perspectives, the authorship of the submissions can be validated through plagiarism detection tools which are widely available both online and offline. Therefore, it will not be considered in this study. Though early research and studies more focus on identity verification a recent study conducted by Senbo, Xiao, and Yingling explains the importance of the identification of the abnormal behaviors of the examinee during the examination process (Hu, Jia and Fu, 2018) along with the identity verification approach.

There are many definitions given for misconduct in the e-assessment domain. Hylton, Levy, and Dringus also define misconduct in the e-assessment as unethical and improper behavior in online exams, and dishonesty is defined as the inability to verify identity and the inability to behave within predetermined parameters. Additionally, they clarify that cheating on an exam happens when information is shared with or obtained from third parties or when prohibited materials are employed (Hylton, Levy and Dringus, 2016) .

As in the literature, different researchers have defined their definitions of trust and trustworthiness. Even though there are differences in these definitions, acceding to the purpose and the context they share the same general idea. Therefore researcher has to define the terms **Examinee Trust**, **Examinee Trustworthiness**, and **Trustworthiness Parameter** specific to this study under the introduction chapter of this study.

2.3 PARAMETERS USED TO DETERMINE THE TRUSTWORTHINESS IN THE E-ASSESSMENT CONTEXT

Different parameters are described in the literature to measure the trustworthiness of the examinee related to identity and behaviors. In literature, biometric parameters are highlighted as key parameters used for the identity verification or authentication of the examinee. The ability to identify and authenticate a person using one or more of their behavioral, physical, or chemical features is referred to as biometrics. (Ojo, Zuva and Ngwira, 2015). Following are some of the behavioral and physical biometric parameters used in the e-assessment context.

- **Kaystork Dynamics**

  To obtain the biometric data related to a person through the Keystore dynamics, it uses the features such as typing speed, keystroke seeks time, flight time, characteristic errors, and characteristic sequences (Ojo, Zuva and Ngwira, 2015). In the TeSLA system, keystroke dynamics are used to verify the identity as well as the authorship of writing assignments (Janssen *et al.*, 2019).

- **Mouse Dynamics**

  Mouse dynamics is another behavioral biometric parameter that uses features like mouse move, drag and drop, point and click, and no motion to collect data. (Ojo, Zuva and Ngwira, 2015). Mouse dynamic is useful for user identity authentication and it can be considered as an invasive method of capturing biometric data (Nazar, 2003).

- **Facial Biometrics**

  This is commonly referred to as facial recognition as well. Almost all the proctoring tools and systems use this parameter to retrieve the authentication data of the user. In these systems, a static image or recorded videos are used to extract the facial identity futures (Bertran, 2018). In the early days, 2D images were used to authenticate, and with the advancement of technology now 3D facial recognition is heavily used, and they are more accurate as well. Many algorithms for facial recognition have been developed by researchers, including the Kanade-Lucas-Tomasi (KLT) algorithm, which uses vector representation of images to establish the identity of the individual (Ojo, Zuva and Ngwira, 2015). Popular Proctoring tools such as TeSLA, ProctoreU, Procter C

am, etc. used this as a key parameter in their system to authenticate the user identity (Foster and Layman, 2013). This metric is also used to manually assess trust in existing proctoring systems.

- **Voice Recognition**

This is also another parameter used in the proctoring systems for identity verification and authentication of the user. In this approach, voice structures will be compared with the pre-generated learner mode. The TeSLA proctoring system also consists of this capability (Okada *et al.*, 2019). The recognition is primarily accomplished by comparing two voice segments; if the input voice is recognized from a known set of voice segments, this is referred to as an in-set scenario; otherwise, this is referred to as an out-set scenario (Hansen and Hasan, 2015). This function is useful for determining the examinee's trustworthiness during the. Unlike other biometric parameters, voice recognition is a performance-based biometric ic and therefore speech signals are prone to large variability and increase the complexity of the identification process. As an example, when a person is not well the voice is different than the normal situations (Hansen and Hasan, 2015).

- **Eye-gaze Tracking**

Eye-gaze tracking is also a parameter for user behavior tracking. Using eye-gaze tracking algorithms and implementations can determine the eye gaze point of a user as he or she looks around and the related coordinates are calculated related to the screen the person is looking at. These eye-gaze points are represented as (x,y) coordinates on the screen(Bawarith *et al.*, 2017). A study conducted by Cuong and Hoang has proposed a real-time eye-gaze detection system with the use of a webcam through a geometry feature extraction method (Cuong and Hoang, 2010). However, the angle of view and eye structure are vital elements of this tracking, and it is dependent on the user. To obtain more precise results from such gaze detections, a calibration stage is usually required and when users suffer eye abnormalities, this gaze tracking becomes more complex (Kar and Corcoran, 2017).

Apart from the biometric parameters, other parameters are also stated in the literature in the e-assessment and proctoring domain. Following are a few such parameters commonly used to determine the trustworthiness in the e-assessment context.

- **Browser Activity Monitoring**

  Browser locking is a common parameter/feature used in many proctoring tools such as Examus, Respondus, and TeSLA. Assessments are displayed full screen in this technique, preventing access to other apps such as chat, screen-sharing, virtual machines, and remote desktops (Foster and Layman, 2013). Safe Exam Browser is one such application that can be coupled with learning management systems (LMS), allowing examiners to design tests via the LMS by utilizing exciting features. A safe exam browser also provides a secure environment in which to conduct the assessments (Nigam *et al.*, 2021). This will impose examiner control over the examinee environment, which was not explored in this study.

- **OS Activity Monitoring**

  Operating system activity monitoring is another parameter commonly used in the e-assessment domain. Such systems can automatically identify open files and folders, unwanted applications, or suspicious OS events which can be concluded as misconduct behavior (Nigam *et al.*, 2021). In these systems, monitoring is configured depending on the examination's rules, and the system alerts when the rules are violated.

In general, by combining one or a few characteristics, it is feasible to confirm the identity of the user and identify inappropriate behavior of users during e-assessments.

2.4 PROCTORING

Compared to the human proctors and invigilators in traditional classroom-based assessments proctoring tools plays the same role in online e-assessments. According to the literature, online proctoring systems are classified into three types, as shown in the table below. (Nigam *et al.,* 2021).

Table 1: Proctoring Tools

Live Proctoring	Recorded Proctoring	Automated Proctoring
• System of real-time proctoring • The human proctor is present. • Suitable for theoretical exams lasting 2-3 hours • The human proctor can detect cheating and malpractice by tracking students' eye movements and recognizing their faces. • Competence in the application of technological advancements is required.	• Involves videotaping the candidate during the examination as well as other log details. • Post-proctoring entails tracking eye and face movements, detecting objects and faces, analyzing logs, and so on. • Human intervention is essential, but it is time-consuming and expensive.	• A more advanced version in which people do not proctor the entire time, but only review • Using various algorithms and technologies, the system detects fraud and cheating. • It is less expensive because no human proctors are used. • Such systems are more difficult to develop.

In the context of live proctoring, it depends on human resources and in mass examinations such as university examinations, it might be infeasible due to a lack of resources. However, because of the human involvement in this type of proctoring accuracy of decision-making is high due to the intelligent behavior of the human proctor.

There are many online proctoring toolings are available in the E-assessment domain such as ProctorU, Xproctor, TeSLA, Safe Exam Browser, and Examus (Nigam *et al.*, 2021). It is also observed that examinee trust is ensured in these proctoring systems through monitoring capabilities.

Following is a brief comparison of such proctoring tools and available features derived from the literature.

Table 2: Online Proctoring Tools

Tool/ Software Name	Description
ProctorU	• Make use of a microphone • Live Proctoring approach • Authenticate student identity • It is necessary to maintain an unbroken audio-visual link
Xproctor	• Use facial recognition, behavior streaming • Supports various LMSs • Authenticates students while constantly tracks monitoring them • Can be installed on Personal Computer
TeSLA	• The system involves authentication and authorship-checking instruments • Use a single tool for Face Recognition, Voice Recognition, Anti-Spoofing, Keystroke Dynamics, and Plagiarism Detection • Exam data is recorded and stored in an LMS server using cloud-based Software as a Service (SaaS)
Safe Exam Browser	• Software that is compatible with an • The device is locked and the user is unable to access other tabs or applications. • Disables shortcuts and copy/paste
Exams	• AI-based software • Capability to extract behavioral characteristics from students during online lectures

As per this table, these different proctoring tools and software work on different aspects and collects different type of data using different parameters. However, most of them are commercial products limited to one or a few aspects of proctoring.

Apart from these software-based solutions, there are several hardware-based solutions available for proctoring as well. As an example, Atom and his team have proposed wearable proctoring spectacles that are capable of user verification, text detection, voice detection, active window detection, gaze estimation, and phone detection (Atoum *et al.*, 2017). Also, solutions such as TeSLA are aimed to use biometric verifications as proctoring approach as well (Bertran, 2018). However, these solutions are somewhat advanced and hard to afford by every student in the institutions especially universities that resides in developing countries.

Also, in most cases, a final human should be available to decide on the student's worthiness by analyzing all these different kinds of collected data. However, fewer researchers have been conducted on thinking about aggregating the data collected through these different types of results into one single unit of measurement. Such research was conducted by Vishnuraj and his team at the E-learning research lab in India, where they propose a heuristic-based automatic proctoring system for E-assessments. They collect data from different imputes such as webcam, audio tracking, and, user screen detection and implemented an inference system to find out the malpractices. However, this system is only capable of detecting malpractice in each of these different scenarios and only intended to implement as a replacement for human proctored in E-assessments (Raj, Narayanan and Bijlani, 2015).

Also, a recent study conducted by Maniar *et al* has implemented a proctoring system with a combination of multiple parameters such as eye gaze tracking, mouth open or close detection, object identification, and head posture estimate using facial landmarks and face detection. However, in this system, there is a proctor to take the final decision on the examinee's trustworthiness with all the captured data (Maniar *et al.*, 2021) and it can be considered a real-time trust monitoring approach.

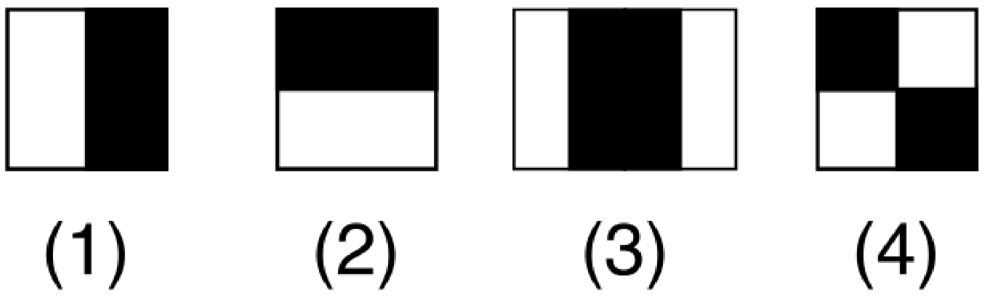
Therefore, this indicates the need for an objective trust measurement approach in online e-assessments in large-scale examinations and the researcher is the focus to fill this research gap in this research study.

2.5 FACE DETECTION TECHNIQUES

OpenCV Method

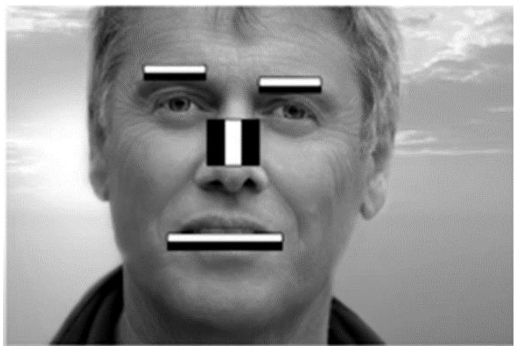
The OpenCV method is highly used for face detection in the computer vision domain. This is a feature-based face detection method and it discovers faces by extracting structural features of the face (Dwivedi, 2018). These features are commonly called **Haar Wavelets** or **Haar Features** and it is used to detect the features in the human face such as eyes, eyebrows, nose, and lips. Alfred Harr in 1909 suggested harr features as a sequence of rescaled square shape functions and the Viola-Jones Algorithm is used to detect the harr-like features in the image (Adakane, 2019).

Figure 1: Haar Wavelets or Haar Features



in the above image 1 and 2 consider edge features, 3 consider a line feature and 4 consider a four-rectangle feature. To determine the human face, these harr features are applied to all relevant parts of the image.

Figure 2: Four-rectangle feature



As in the above image, it is possible to represent the most relevant features by using these harr features. In this process, a large sample set is generated by extracting feature images and then the AdaBoost algorithm is used to detect the faces (Lu and Yan, 2021).

MediaPipe Method

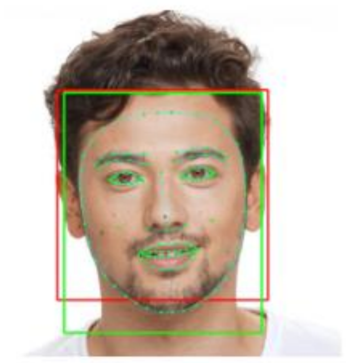
MediaPipe face detection is a quick face detection system that is supported by Google machine learning and focuses on live and streaming data. This solution consists of 6 facial landmarks and multi-face detection ability (‘MediaPipe Face Detection’, 2020). This solution contains three face detection models such as,

- Short-range model (best for faces up to 2 meters away from the camera)
- Full-range model (dense, best for faces up to 5 meters away from the camera)
- Full-range model (sparse, best for faces up to 5 meters away from the camera)

Depending on the application user can select the preferred model to optimize this task.

The BlazeFace algorithm, which is a lightweight and high-performance face detector adapted from the Single Shot Multibox Detector (SSD) framework, is the basic theory behind this approach (Bazarevsky *et al.*, 2019). Researchers trained this model with 66K images and evaluated it on a private geographically diversified dataset of 2K images, achieving an average precision of 97.95% (Bazarevsky *et al.*, 2019). Therefore this can be ranked as one of high accurate face detection models in the computer vision domain.

Figure 3: BlazeFace algorithm output



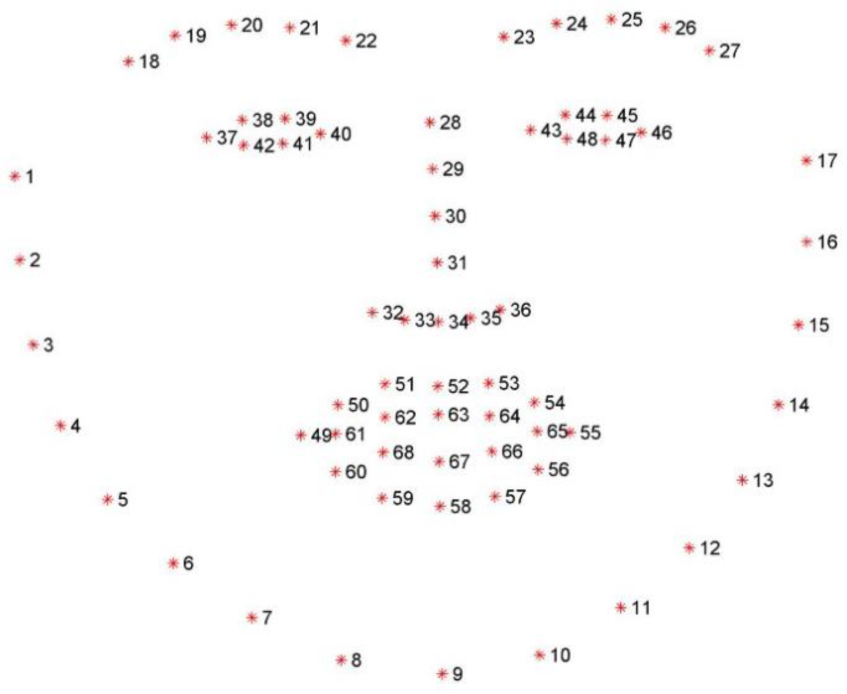
This is an example of BlazeFace output in red color and task-specific output in green color.

2.6 GAZE DETECTION

There are three types of eye-tracking techniques in the computer vision domain. The first is electro-oculography (which involves inserting electrodes near the eye skin), the second is contact lens-based eye coil systems, and the third is image-based tracking of eye movement. (Cuong and Hoang, 2010). The image-based method is a noninvasive method and therefore it has gained more popularity over time. s

In this study, the researcher has studied the Dlib library ('Dlib', 2022) which is a modern C++ machine learning algorithm toolkit. To detect 68 facial key points it uses a pre-trained network which was proposed by Kazemi and Sullivan in their research (Kazemi and Sullivan, 2014). The researcher has used Dlib in the study because it can give predictions on a real-time basis.

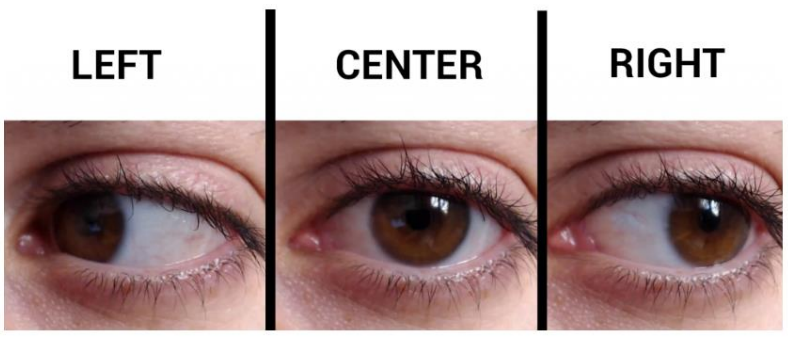
Figure 4: Facial landmarks with Dlib



By using the facial keypoint detecter it is possible to find the eyes in the given image and it is the first step of gaze detection.

After identifying the eye, the gaze can be predicted using the pupil, iris, and sclera positions. The following image illustrates the position of each eye component when gazing in different directions.

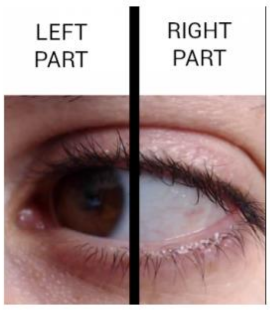
Figure 5: Eye component illustration



As shown in the image, the sclera (white portion of the eye) occupies the right part of the eye when looking to the left, and the sclera fills the left part of the eye when looking to the right. When staring at the middle, the sclera is balanced between the left and right.

Using this fact, the term Eye gaze ratio (Canu, 2019) is introduced, with the concept being to divide the eye into two sections and determine which of the two has a more visible sclera.

Figure 6: Eye gaze ratio



In that scenario, the sclera is more visible on the right side, indicating that the eye is looking to the left, and the sclera is more visible on the left side, indicating that the eye is looking to the right. To compute the gaze ratio, after locating the eye in a given image, the image is converted to grayscale, and the white and black pixels in the image can be counted by defining a threshold (Canu, 2019).

```
          gaze_ratio = left_side_white / right_side_white
```

2.7 IMAGE CLASSIFICATION

Image classification is the process through which a computer analyzes the properties of an image and determines which class it belongs to. This class is commonly known as the label. Image classification calculates the probability of being related to a particular class. The generic classification methods are classified into two types: supervised classification and unsupervised classification. (Gavali and Banu, 2019).

In this research, the researcher is mainly focused on the supervised classification where the user defines a training set that contains homogeneous items based on prior knowledge, and then once an item is given the classification model determines the best-fitted class for the given item.

Convolutional neural networks, or CNNs, are frequently used in image classification with machine learning. (‘what is image classification in deep learning?’, 2021). All neural networks, in general, are made up of three layers: an input layer, hidden layers, and an output layer. There are various varieties of neural networks, with the convolutional neural network standing out due to its hidden layer. Convolution neural networks consist of hidden layers such as

- **Convolutional layers** - Convolutional layers are used to identify and extract information from images.
- **Pooling layers –** Using pooling, objects can be detected regardless of where they are in the image.
- **ReLU layers -** ReLU layers enable the computer to process more complex, non-linear data images.
- **Fully connected layers -** This is where the convolutional neural network's retrieved features are merged. (O’Shea and Nash, 2015)

In practice, CNNs are best used for facial recognition, handwriting recognition, and image classification (Li *et al.*, 2021).

CHAPTER 3

3 METHODOLOGY

The researcher will address the research methodology in this section, which discusses the nature of this study, research methods, data gathering procedure, experiments, and prototype architecture used in this study.

3.1 EXPERIMENTAL DATA SET

To understand the trust parameters researcher has conducted several experiments based on a private secondary dataset. The dataset contains 39,168 images of students, which were captured during moodle proctored examinations. All the images are randomized and anonymized by masking the student identity information.

3.2 EXPERIMENTS

3.2.1 EXPERIMENT 01 – FACE DETECTION

Background

The purpose of this experiment is to detect human faces in the image. The availability of the examinee in the camera frame is a critical aspect of the trust evaluation process in an automated protected e-assessment. If the examiner is unavailable in the given image frame it gives us an indication that there is a violation of trust trough out the given period. This might be a deliberate action by the examinee or might be due to a technical issue as well.

Problem

The problem of this experiment is, how we can identify whether the examinee available in the image captured in the automated proctored e-assessment.

Experimentation Hypothesis

Implemented OpenCV-based image detection program (See Appendix B: OpenCV Based Face Detection) will be able to give high accuracy in detecting faces in the given image.

Experiment Design

To conduct this experiment researcher has implemented OpenCV based python program to filter the images with faces in the data set. The program has used the pre-trained Haar cascade models to detect faces and eyes in an image (‘OpenCV’, 2022). Also, the program was executed with different scaleFactor (settings indicating the amount by which the image size is reduced at each image scale) to optimize the output.

Results and Observations

The following results have been obtained from the python filtering program with scaleFactor=2

> 01 Number of images where the program **can detect a face** in the image using the used classifier - 19, 633 images (50.12 %)
>
> 02. Number of images where the program **cannot detect a face** in the image using the used classifier - 19, 535 images (49.87 %)

Flowing results have been obtained from the python filtering program with scaleFactor= 1.05

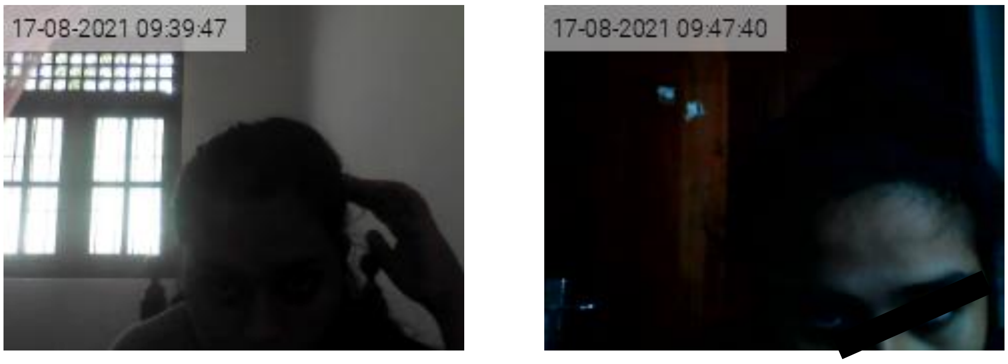
> 01. Number images where the program **can detect a face** in the image using the used classifier -26, 566 images (67.82 %)
>
> 02. Number of images where the program **cannot detect a face** in the image using the used classifier - 12, 602 images (32.17 %)

By setting the scaleFactor= 1.05 researcher was able to gain the maximum throughput of this program.

When analyzing the identified images qualitatively, the researcher has observed that even though there is a face in the image, the program was unable to identify the face in the image, due to the following reasons

- Dark background

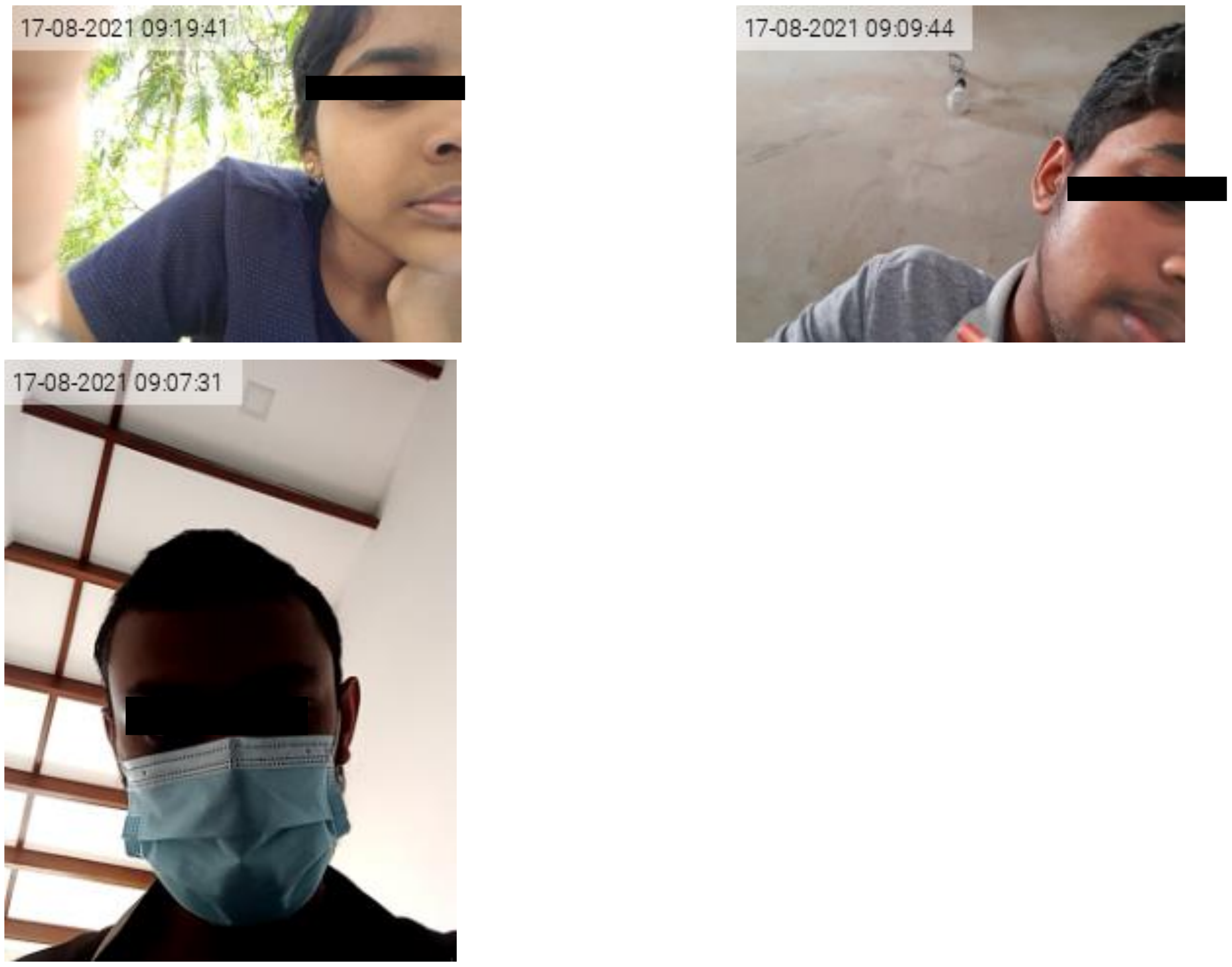
Figure 7: Experiment 01: Dark Background



[redacted]

- Images containing half of the face

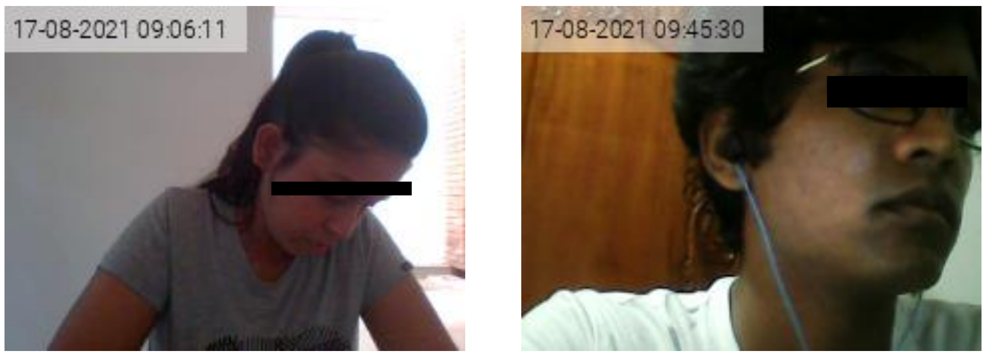
Figure 8: Experiment 01: Images containing half of the face



[redacted]
[redacted]
[redacted]

- Images containing a side view of the face

Figure 9: Experiment 01: Images containing a side view of the face



[redacted]
[redacted]

Conclusion

With this experiment, the researcher has concluded that with OpenCV-based face detection approach best works with a scale Factor= 1.05.

In a protected e-assessment the correct placement of the camera and the background selection is a key responsibility of the examinee, and therefore images with dark backgrounds, and half-face images are generated due to a lack of attention to details by the examinee. This might happen accidentally as well as deliberately. Therefore, they can be considered trust-breaching parameters during a proctored e-assessment.

However, the side face images are acceptable, and since the program was unable to identify them, the researcher needs to find a mechanism to detect them with a different filter approach.

3.2.2 EXPERIMENT 02 – MULTIPLE FACE DETECTION 01

Background

The purpose of this experiment is to detect multiple human faces in the given image. In an protected e-assessment multiple faces in the image indicate a breach of trust in the assessment process. Therefore, the detection of multiple faces in the given image is important to implement a reliable e-assessment process.

Problem

The problem of this experiment is, how we can identify multiple faces available in the image captured during the automated proctored e-assessment.

Experimentation Hypothesis

Implemented OpenCV-based multiple-face image filter (See Appendix B: OpenCV Based Multiple Face Detection) will be able to give high accuracy in detecting multiple faces in the given image.

Experiment Design

To conduct this experiment researcher has implemented OpenCV based python program to filter the images with multiple faces in the data set. During the previous experiment, the filtered dataset containing the faces was used as the input dataset for this experiment. This program has also used the pre-trained Haar cascade models to detect faces and eyes in an image (‘OpenCV’, 2022)

Results and Observations

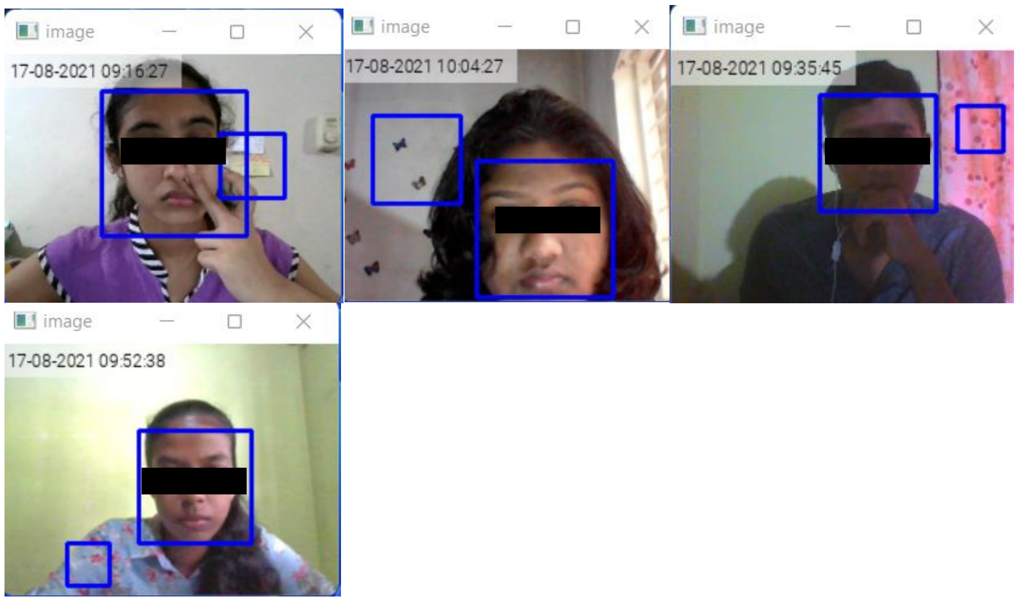
Out of 26, 566 images (filtered by the previous experiment) containing faces, the program was able to label 792 images as suspected images that contain multiple faces. However, the following points has observer when the researcher went through these images manually

- Only one image contained real two faces
- 15 images with ID card photos and the student's face (Logically two faces in the image)
- Apart from that, all the others were wrong predictions

Also, the researcher has observed that the background and dress matter for this identification. As an example, if the background or dress contains face-like objects then the program model will predict them as faces.

following are some of the images captured through the experiment

Figure 10: Images with a face like objects in the background/ dress



[redacted]
[redacted]
[redacted]
[redacted]

Figure 11: Images with an Identity card

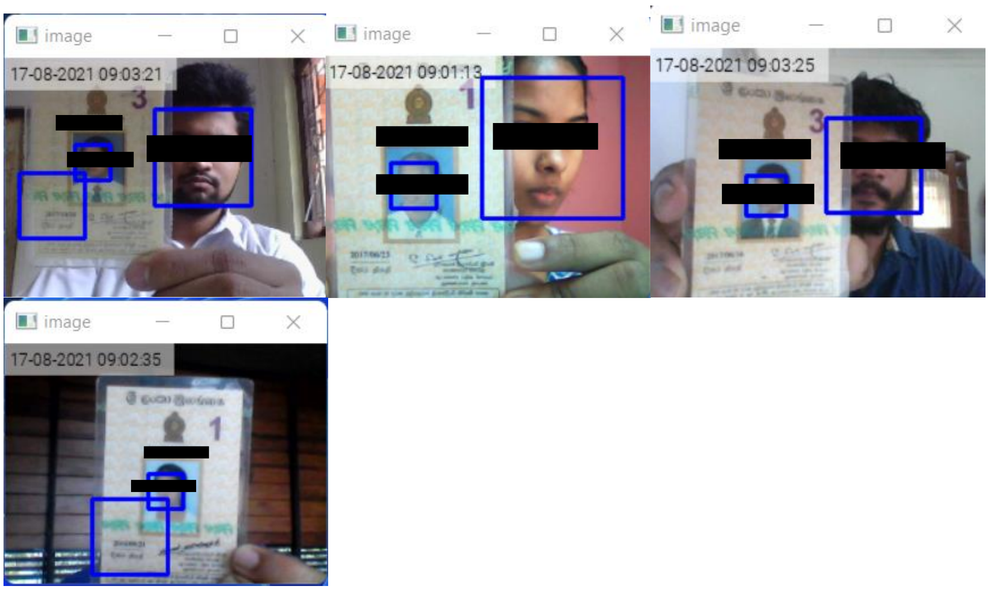
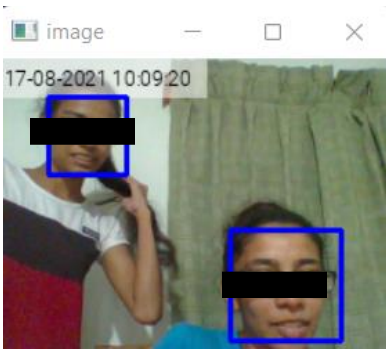


Figure 12: Correctly predicted Image



Conclusion

The reason for the high error rate in this experiment is the used OpenCV face detection model is based on **Haar Wavelets or Haar Features** (Adakane, 2019) as explained in the literature review. If the model finds a harr feature in the given image it classifies it as a face and it will lead to a wrong prediction. As a result of the high error rate, the researcher has decided that the chosen OpenCV multi-face detection model is unsuitable for this study.

3.2.3 EXPERIMENT 03 – MULTIPLE FACE DETECTION 02

Background

The purpose of this experiment is to detect multiple human faces in the given image. In an automated protected e-assessment multiple faces in the image indicate a breach of trust in the assessment process with higher accuracy than the OpenCV-based approach.

Problem

The problem of this experiment is, how we can identify multiple faces available in the image captured during the automated proctored e-assessment with higher accuracy than OpenCV based approach.

Experimentation Hypothesis

Implemented mediaPipe-based multiple-face image filters will be able to give high accuracy in detecting multiple faces in the given image.

Experiment Design

To conduct this experiment researcher has implemented a media pipe-based python program (See Appendix B: MediaPipe Based Multiple Face Detection) to filter the images with multiple faces in the data set. During the first experiment, the filtered dataset containing the faces was used as the input dataset for this experiment.

Results and Observations

following results have been obtained from the media pipe-based python filtering program with the data set of 26, 566 images.

01. Number of images where the program **can detect a single face** in the image - 25, 127 images (94.58 %)

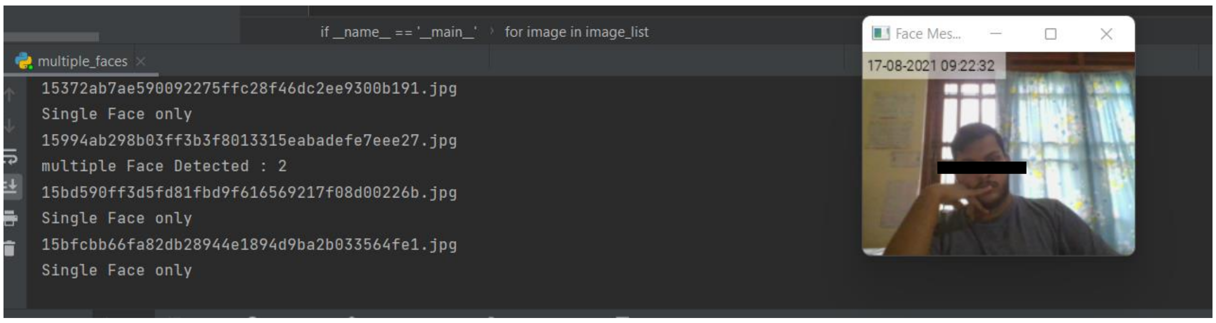
02. Number of images where the program **cannot detect a face** in the image using the used classifier – 1 415 images (5.32 %)

03. Number of images where the program **can detect multiple faces** in the image using the used classifier - 01 image

03. Number of images where the program **incorrectly detects multiple faces** in the image using the used classifier - 23 image

Following are some of the images captured through the experiment

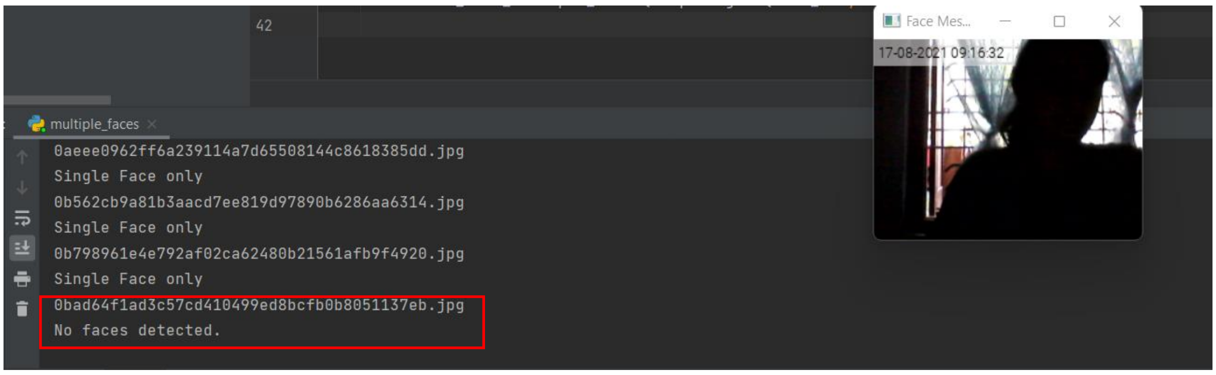
Figure 13 Correct prediction even with face-like background



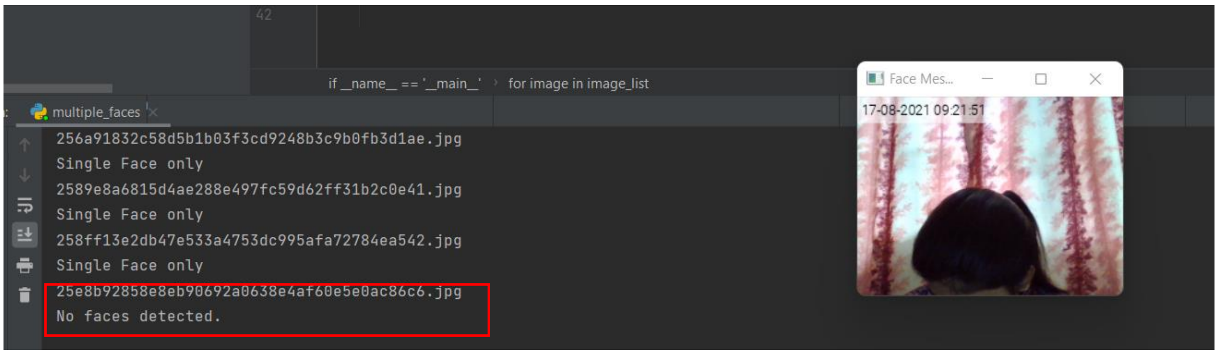
[redacted]

```
15372ab7ae590092275ffc28f46dc2ee9300b191.jpg
Single Face only
15994ab298b03ff3b3f8013315eabadefe7eee27.jpg
multiple Face Detected : 2
15bd590ff3d5fd81fbd9f616569217f08d00226b.jpg
Single Face only
15bfcbb66fa82db28944e1894d9ba2b033564fe1.jpg
Single Face only
```

Figure 14: Unable to identify a face or multiple faces

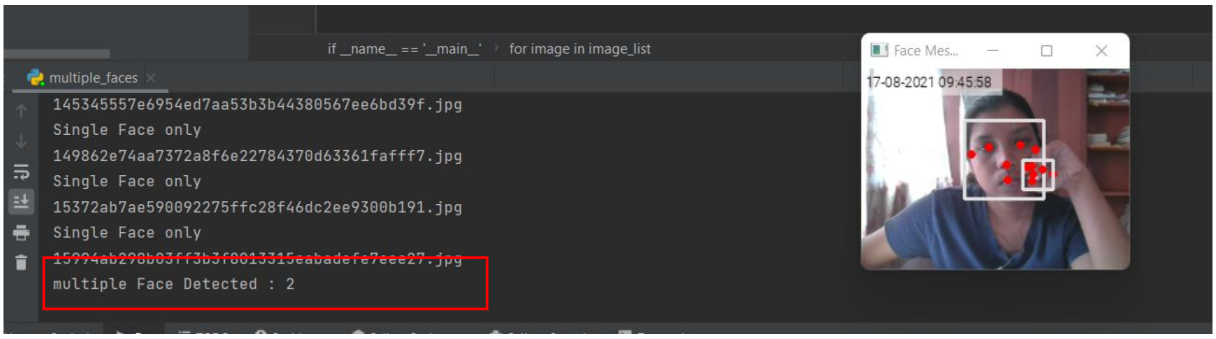


```
0aeee0962ff6a239114a7d65508144c8618385dd.jpg
Single Face only
0b562cb9a81b3aacd7ee819d97890b6286aa6314.jpg
Single Face only
0b798961e4e792af02ca62480b21561afb9f4920.jpg
Single Face only
0bad64f1ad3c57cd410499ed8bcfb0b8051137eb.jpg
No faces detected.
```

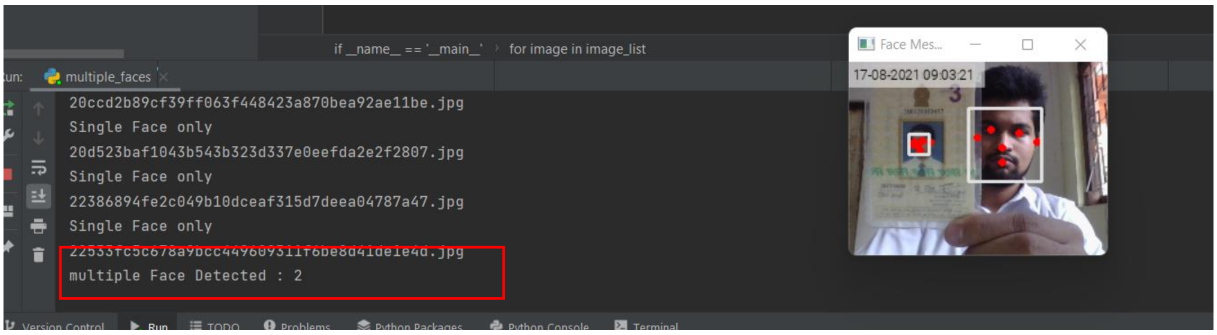


```
256a91832c58d5b1b03f3cd9248b3c9b0fb3d1ae.jpg
Single Face only
2589e8a6815d4ae288e497fc59d62ff31b2c0e41.jpg
Single Face only
258ff13e2db47e533a4753dc995afa72784ea542.jpg
Single Face only
25e8b92858e8eb90692a0638e4af60e5e0ac86c6.jpg
No faces detected.
```

Figure 15: Incorrect prediction with multiple faces including identity images

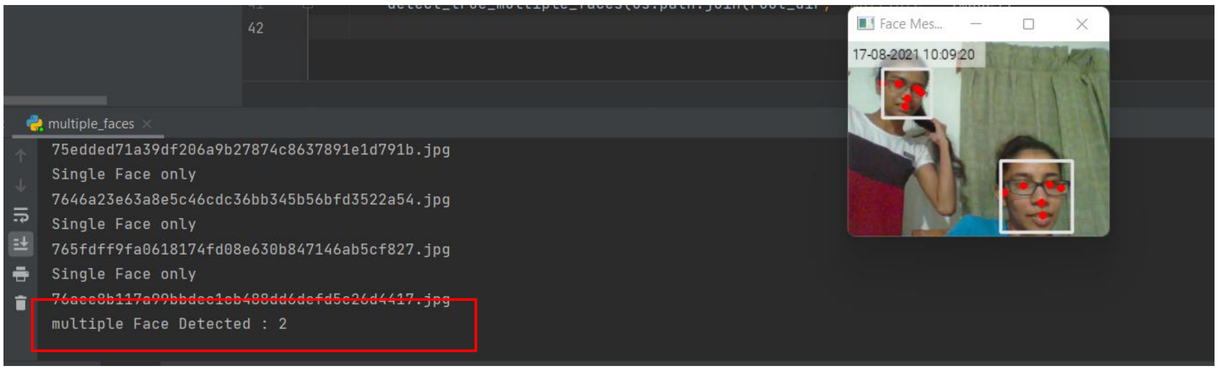


```
145345557e6954ed7aa53b3b44380567ee6bd39f.jpg
Single Face only
149862e74aa7372a8f6e22784370d63361fafff7.jpg
Single Face only
15372ab7ae590092275ffc28f46dc2ee9300b191.jpg
Single Face only
15994ab298b03ff3b3f8013315eabadefe7eee27.jpg
multiple Face Detected : 2
```

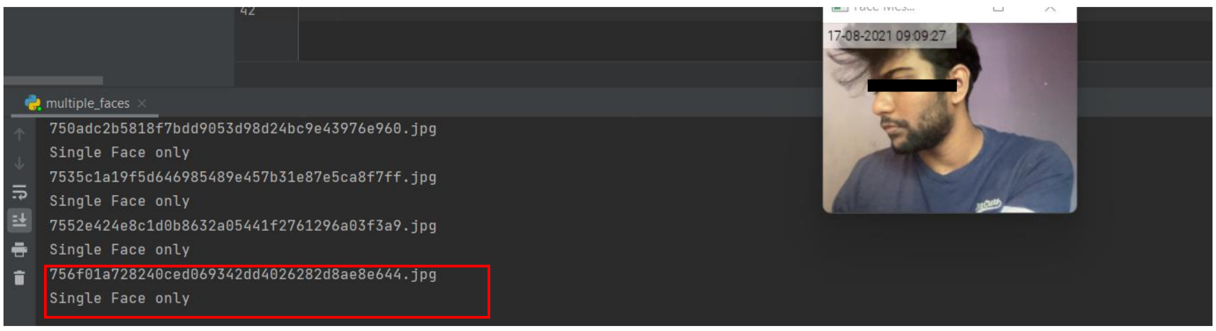


```
20ccd2b89cf39ff063f448423a870bea92ae11be.jpg
Single Face only
20d523baf1043b543b323d337e0eefda2e2f2807.jpg
Single Face only
22386894fe2c049b10dceaf315d7deea04787a47.jpg
Single Face only
22533fc5c678a9bcc449609311f6be8d41de1e4d.jpg
multiple Face Detected : 2
```

Figure 16: Correct prediction even with the side of the face



```
75edded71a39df206a9b27874c8637891e1d791b.jpg
Single Face only
7646a23e63a8e5c46cdc36bb345b56bfd3522a54.jpg
Single Face only
765fdff9fa0618174fd08e630b847146ab5cf827.jpg
Single Face only
76ace8b117a99bbdec1cb488dd6defd5e26d4417.jpg
multiple Face Detected : 2
```



```
750adc2b5818f7bdd9053d98d24bc9e43976e960.jpg
Single Face only
7535c1a19f5d646985489e457b31e87e5ca8f7ff.jpg
Single Face only
7552e424e8c1d0b8632a05441f2761296a03f3a9.jpg
Single Face only
756f01a728240ced069342dd4026282d8ae8e644.jpg
Single Face only
```

Conclusion

Compared to the resulting gain through the OpenCV-based face detection program, the media-pipe-based python program was able to gain a low error rate in face prediction. This is due to the performance efficiency of the BlazeFace algorithm (Bazarevsky *et al.*, 2019) used in the media pipe model implementation. Therefore, the researcher decided to implement the single and multiple face detection filters with media-pipe-based implementation. One of the advantages of this implementation was it was able to detect the faces with the side view of the face as well. Also, it is observed that if this model is unable to detect the face, which emphasizes that there is a serious issue with background and lighting which is a clear indication of a violation of trust with the examination proctoring setup.

3.2.4 EXPERIMENT 04 – ML APPROACH TO DETECT ACCEPTED AND UNACCEPTED IMAGES

Background

Image classification is one approach that can be used to distinguish whether the given image is accepted or not in a proctoring e-assessment. With the previous experiments, the researcher was able to manually identify a good set of acceptable and unacceptable images from the initial dataset (39,168 images). The goal of this experiment is to investigate the potential use of Convolutional Neural Network-based image classification in the context of proctored e-assessment.

Problem

The problem of this experiment is, how we can identify a captured image during a proctored e-assessment is acceptable or not using CNN-based image classification.

Experimentation Hypothesis

Implemented CNN-based image classifier (See Appendix B: ML Image Classifier) will be able to give high accuracy in detecting unaccepted images in the given data set.

Experiment Design

To conduct this experiment researcher has used the filtered data set from the previous experiment. The researcher has built the following experimental data set manually to experiment.

01. Training acceptable images

This contains 376 images labeled as ‘accepted’. These images contain images with acceptable poses, clear backgrounds, and only one face available in the given image frame.

02. Testing acceptable images

This contains 151 images labeled as ‘accepted’. These images contain images with acceptable poses, clear backgrounds, and only one face available in the given image frame.

03. Training unacceptable images

This contains 376 images labeled as ‘rejected. These images contain images with unacceptable poses, dark backgrounds, images with half of the faces or side of the face visible in the frame, and multiple faces available in the given image frame.

04. Testing unacceptable images

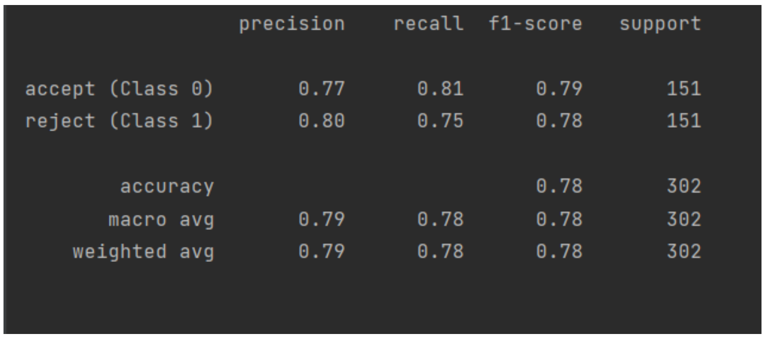
This contains 151 images labeled as ‘rejected. These images contain images with unacceptable poses, dark backgrounds, images with half of the faces or side of the face visible in the frame, and multiple faces available in the given image frame.

The researcher created a Python-based image classification tool that uses a simple CNN model with three Convolutional layers and a max-pooling layer. To avoid overfitting, a dropout layer is inserted after the third max pool operation. Due to the extremely low learning rate that was chosen, a reduced learning rate of 0.000001 was also used, and 500 epochs were used to train the model.

Results and Observations

Following is the result obtained from the experiment related to training and validation accuracy and training and validation loss. The researcher was able to gain 0.78 accuracies in this experiment.

Figure 17: Training & Validation Accuracy



```
                    precision    recall  f1-score   support

accept (Class 0)         0.77      0.81      0.79       151
reject (Class 1)         0.80      0.75      0.78       151

        accuracy                             0.78       302
       macro avg         0.79      0.78      0.78       302
    weighted avg         0.79      0.78      0.78       302
```

Figure 18: Training & Validation Accuracy Graph

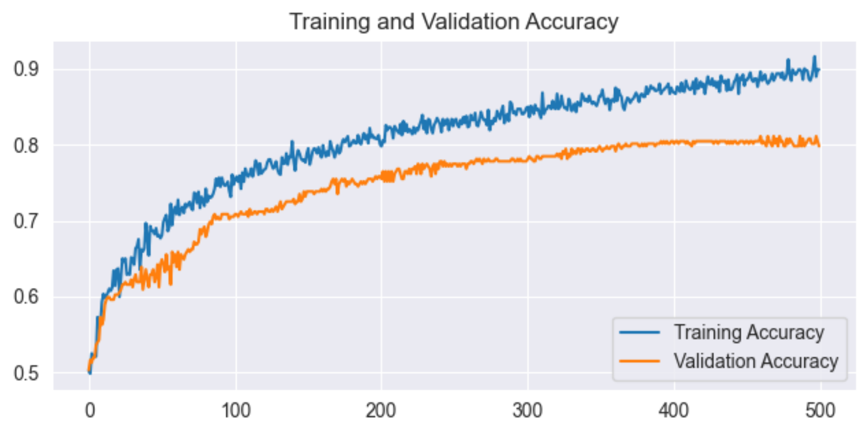
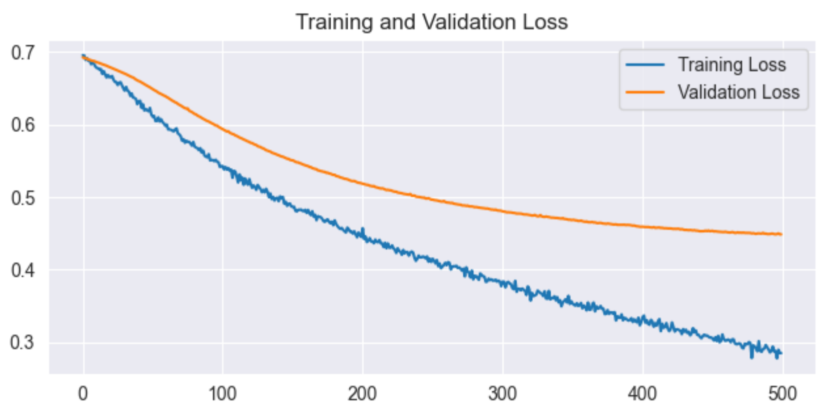


Figure 19: Training & Validation Loss Graph



Conclusion

This model achieved good accuracy with the testing data set. Because of the trained model's mobility and ease of integration with other applications, the researcher chose to employ this trained ML model to filter suspicious images.

3.2.5 EXPERIMENT 05 – EYE GAZE DETECTION

Background

During the online proctored e-assessments, if a laptop webcam is used the front face of the examinee should be visible to the image frame. If the examinee looks away continuously or intentionally it will become a breach of trust or suspicious activity. Therefore, there should be a mechanism to track this behavior of the examinee though out the online e-assessment process.

Problem

The problem of this experiment is, how we can detect suspicious gazes with high accuracy.

Experimentation Hypothesis

Implemented python-based eye gaze detecter (See Appendix B: Eye Gaze Detection) will be able to give high accuracy in detecting suspicious eye gazes

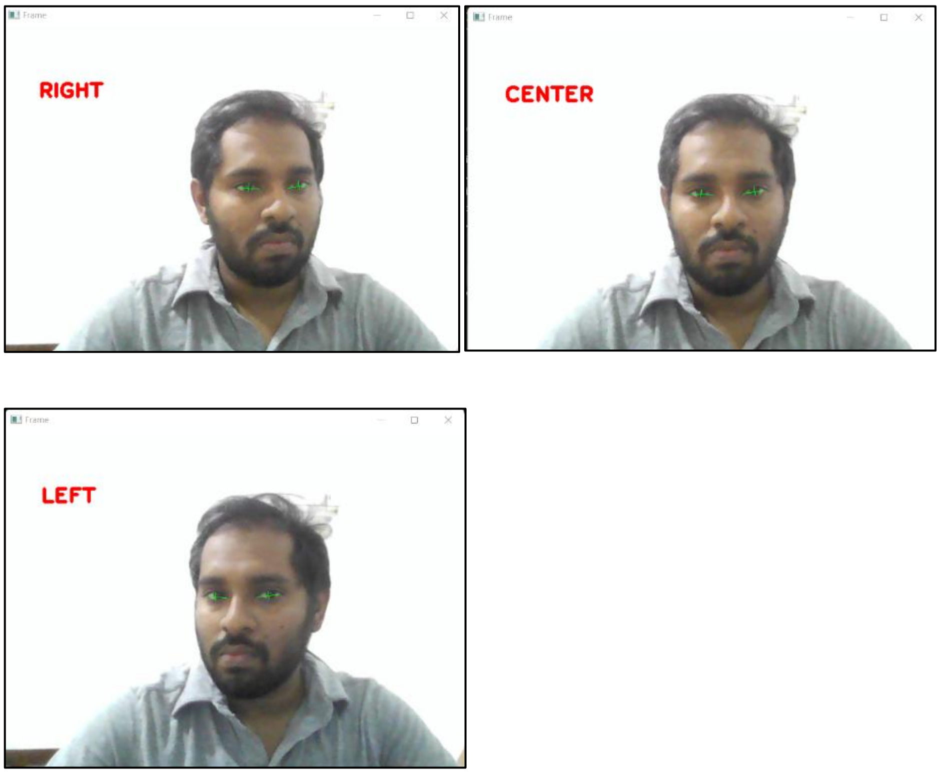
Experiment Design

To conduct this experiment researcher has implemented an OpenCV(‘OpenCV’, 2022) and Dlib(‘Dlib’, 2022) based python program to detect the examinee's gaze. Eye gaze is calculated based on the gaze ratio (Canu, 2019) of human eyes.

Results and Observations

As in the following images, the program was able to detect the gaze of the human eye. However, it is observed that the program is not detecting all the pauses correctly.

Figure 20: Gaze Detection



Conclusion

When analyzing the observation researcher noticed that the default gaze ratio varies because of the shape of the eye. Therefore, the person's default gaze ratios should be calculated in advance to obtain more accurate results during the detection period. Also, it is difficult to detect the gaze using this method if a person has eye orientation issues or is wearing eyeglasses. Due to these drawbacks, the researcher decided not to use this approach in the study.

3.2.6 EXPERIMENT 06 –IDENTITY DETECTION

Background

It's possible that someone else will perform the exam in place of the examinee during the online proctored e-assessments, or that they will swap places at some point. Another trust parameter required to evaluate this study is determining the examinee's identity.

Problem

The problem of this experiment is, how we can detect the examinee’s identity during the e-assessment.

Experimentation Hypothesis

Implemented python-based identity detector (See Appendix B: Identity Detection) will be able to give high accuracy in detecting the examinee’s identity

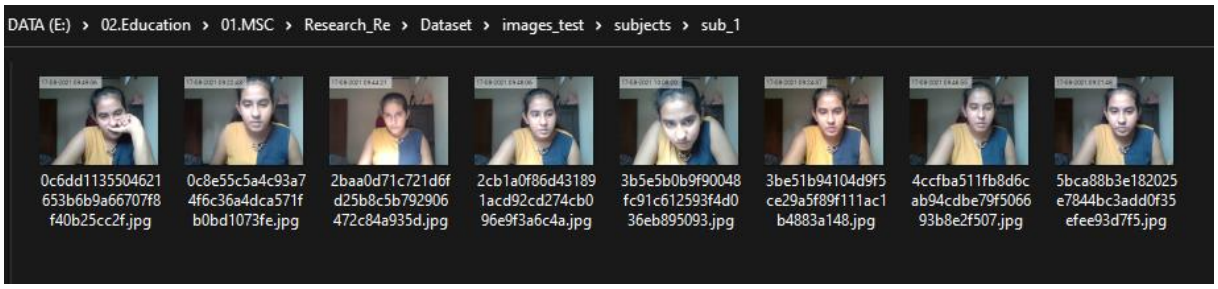
Experiment Design

To conduct this experiment researcher has implemented a python based program for identity verification. The Implemented program was executed in two stages such as training and validating. An exportable facial encoding model will be developed specifically for the examinee once a series of facial images are acquired using the program during the training stage. Then, a series of webcam images will be taken during the testing stage and they will be compared with the created facial model to determine the identity of the examinee. And if it detects any identity mismatch it will log the details.

Results and Observations

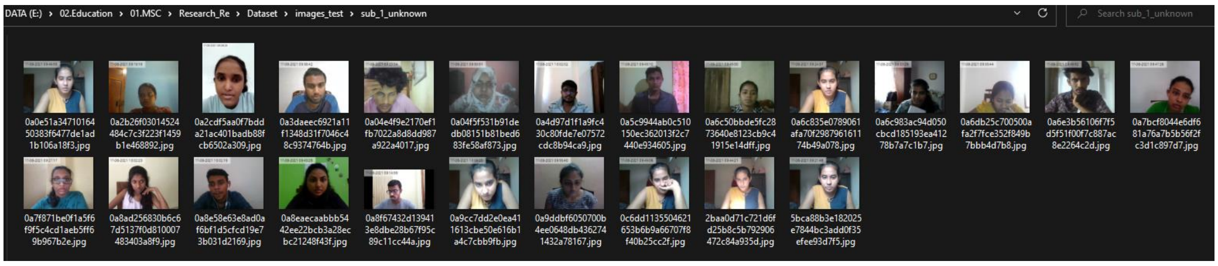
Training input for encoding model

Figure 21: Experiment 06; Training input for encoding model


DATA (E:) › 02.Education › 01.MSC › Research_Re › Dataset › images_test › subjects › sub_1
0c6dd1135504621653b6b9a66707f8f40b25cc2f.jpg
0c8e55c5a4c93a74f6c36a4dca571fb0bd1073fe.jpg
2baa0d71c721d6fd25b8c5b792906472c84a935d.jpg
2cb1a0f86d431891acd92cd274cb096e9f3a6c4a.jpg
3b5e5b0b9f90048fc91c612593f4d036eb895093.jpg
3be51b94104d9f5ce29a5f89f111ac1b4883a148.jpg
4ccfba511fb8d6cab94cdbe79f506693b8e2f507.jpg
5bca88b3e182025e7844bc3add0f35efee93d7f5.jpg

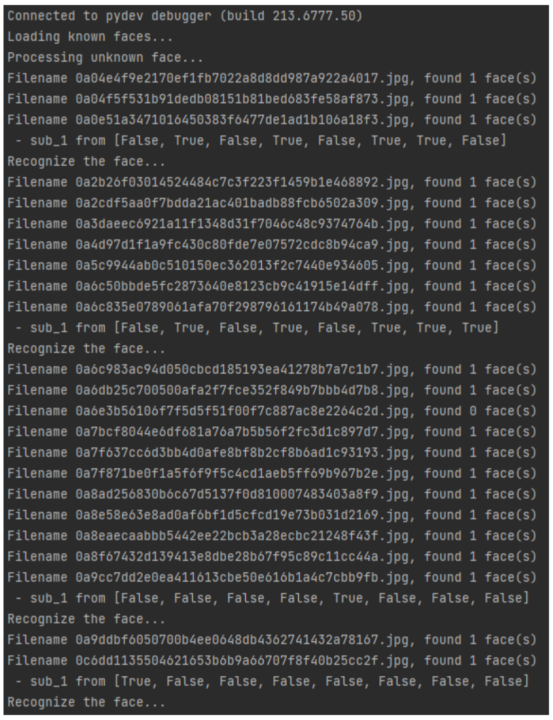
Random images for recognition

Figure 22: Experiment 06; Random images for recognition


DATA (E:) › 02.Education › 01.MSC › Research_Re › Dataset › images_test › sub_1_unknown
Search sub_1_unknown
0a0e51a3471016450383f6477de1ad1b106a18f3.jpg
0a2b26f03014524484c7c3f223f1459b1e468892.jpg
0a2cdf5aa0f7bdda21ac401badb88fcb6502a309.jpg
0a3daeec6921a11f1348d31f7046c48c9374764b.jpg
0a04e4f9e2170ef1fb7022a8d8dd987a922a4017.jpg
0a04f5f531b91dedb08151b81bed683fe58af873.jpg
0a4d97d1f1a9fc430c80fde7e07572cdc8b94ca9.jpg
0a5c9944ab0c510150ec362013f2c7440e934605.jpg
0a6c50bbde5fc2873640e8123cb9c41915e14dff.jpg
0a6c835e0789061afa70f298796161174b49a078.jpg
0a6c983ac94d050cbcd185193ea41278b7a7c1b7.jpg
0a6db25c700500afa2f7fce352f849b7bbb4d7b8.jpg
0a6e3b56106f7f5d5f51f00f7c887ac8e2264c2d.jpg
0a7bcf8044e6df681a76a7b5b56f2fc3d1c897d7.jpg
0a7f871be0f1a5f6f9f5c4cd1aeb5ff69b967b2e.jpg
0a8ad256830b6c67d5137f0d810007483403a8f9.jpg
0a8e58e63e8ad0af6bf1d5cfcd19e73b031d2169.jpg
0a8eaecaabbb5442ee22bcb3a28ecbc21248f43f.jpg
0a8f67432d139413e8dbe28b67f95c89c11cc44a.jpg
0a9cc7dd2e0ea411613cbe50e616b1a4c7cbb9fb.jpg
0a9ddbf6050700b4ee0648db4362741432a78167.jpg
0c6dd1135504621653b6b9a66707f8f40b25cc2f.jpg
2baa0d71c721d6fd25b8c5b792906472c84a935d.jpg
5bca88b3e182025e7844bc3add0f35efee93d7f5.jpg

Recognition output

Figure 23: Experiment 06; Recognition output



```
Connected to pydev debugger (build 213.6777.50)
Loading known faces...
Processing unknown face...
Filename 0a04e4f9e2170ef1fb7022a8d8dd987a922a4017.jpg, found 1 face(s)
Filename 0a04f5f531b91dedb08151b81bed683fe58af873.jpg, found 1 face(s)
Filename 0a0e51a3471016450383f6477de1ad1b106a18f3.jpg, found 1 face(s)
 - sub_1 from [False, True, False, True, False, True, True, False]
Recognize the face...
Filename 0a2b26f03014524484c7c3f223f1459b1e468892.jpg, found 1 face(s)
Filename 0a2cdf5aa0f7bdda21ac401badb88fcb6502a309.jpg, found 1 face(s)
Filename 0a3daeec6921a11f1348d31f7046c48c9374764b.jpg, found 1 face(s)
Filename 0a4d97d1f1a9fc430c80fde7e07572cdc8b94ca9.jpg, found 1 face(s)
Filename 0a5c9944ab0c510150ec362013f2c7440e934605.jpg, found 1 face(s)
Filename 0a6c50bbde5fc2873640e8123cb9c41915e14dff.jpg, found 1 face(s)
Filename 0a6c835e0789061afa70f298796161174b49a078.jpg, found 1 face(s)
 - sub_1 from [False, True, False, True, False, True, True, True]
Recognize the face...
Filename 0a6c983ac94d050cbcd185193ea41278b7a7c1b7.jpg, found 1 face(s)
Filename 0a6db25c700500afa2f7fce352f849b7bbb4d7b8.jpg, found 1 face(s)
Filename 0a6e3b56106f7f5d5f51f00f7c887ac8e2264c2d.jpg, found 0 face(s)
Filename 0a7bcf8044e6df681a76a7b5b56f2fc3d1c897d7.jpg, found 1 face(s)
Filename 0a7f637cc6d3bb4d0afe8bf8b2cf8b6ad1c93193.jpg, found 1 face(s)
Filename 0a7f871be0f1a5f6f9f5c4cd1aeb5ff69b967b2e.jpg, found 1 face(s)
Filename 0a8ad256830b6c67d5137f0d810007483403a8f9.jpg, found 1 face(s)
Filename 0a8e58e63e8ad0af6bf1d5cfcd19e73b031d2169.jpg, found 1 face(s)
Filename 0a8eaecaabbb5442ee22bcb3a28ecbc21248f43f.jpg, found 1 face(s)
Filename 0a8f67432d139413e8dbe28b67f95c89c11cc44a.jpg, found 1 face(s)
Filename 0a9cc7dd2e0ea411613cbe50e616b1a4c7cbb9fb.jpg, found 1 face(s)
 - sub_1 from [False, False, False, False, True, False, False, False]
Recognize the face...
Filename 0a9ddbf6050700b4ee0648db4362741432a78167.jpg, found 1 face(s)
Filename 0c6dd1135504621653b6b9a66707f8f40b25cc2f.jpg, found 1 face(s)
 - sub_1 from [True, False, False, False, False, False, False, False]
Recognize the face...
```

the implemented program was able to detect the identity of the examinee in real-time. However, it is observed that some of the partial face images will not get detected in this program.

Conclusion

The identity of the examinee can be quickly and accurately determined using this application. As a result, the researcher decided to incorporate this python-based facial recognition filter into to final tooling implementation.

3.2.7 EXPERIMENT 07 – ACTIVITY DETECTION

Background

During the online proctored e-assessments, the examinee might open unauthorized materials and cheat during the examination process. Detecting whether the examinee is accessing unauthorized materials is another trust parameter that is needed to assess this study.

Problem

The problem of this experiment is, how we can detect unauthorized material access with high accuracy.

Experimentation Hypothesis

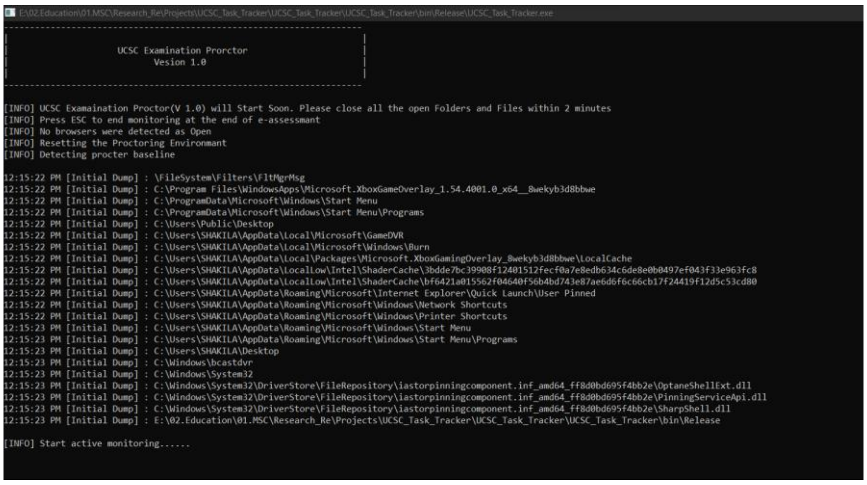
Implemented .net-based activity tracker will be able to give high accuracy in detecting suspicious material access

Experiment Design

To conduct this experiment researcher has implemented a .net base program (See Appendix B: OS Activity Detection) to detect the windows file opening operations. The program tracks the directories which open in real-time and logs into a file with the access details. This program initially closes all open file handlers in the windows system and also asks students to close all the files as well. Once the monitoring stared it will log all the suspicious directory open operations in real time.

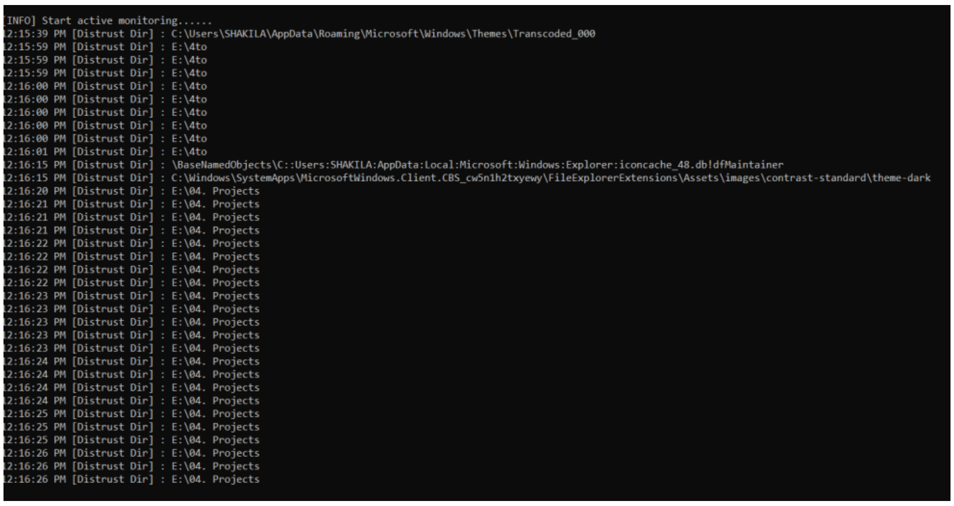
Results and Observations

Figure 24: Experiment 07; Activity Detection Startup



```
E:\02.Education\01.MSC\Research_Re\Projects\UCSC_Task_Tracker\UCSC_Task_Tracker\UCSC_Task_Tracker\bin\Release\UCSC_Task_Tracker.exe
--------------------------------------------------------------------
|                                                                  |
|                    UCSC Examination Prorctor                     |
|                           Vesion 1.0                             |
|                                                                  |
--------------------------------------------------------------------

[INFO] UCSC Examaination Proctor(V 1.0) will Start Soon. Please close all the open Folders and Files within 2 minutes
[INFO] Press ESC to end monitoring at the end of e-assessmant
[INFO] No browsers were detected as Open
[INFO] Resetting the Proctoring Environmant
[INFO] Detecting procter baseline

12:15:22 PM [Initial Dump] : \FileSystem\Filters\FltMgrMsg
12:15:22 PM [Initial Dump] : C:\Program Files\WindowsApps\Microsoft.XboxGameOverlay_1.54.4001.0_x64__8wekyb3d8bbwe
12:15:22 PM [Initial Dump] : C:\ProgramData\Microsoft\Windows\Start Menu
12:15:22 PM [Initial Dump] : C:\ProgramData\Microsoft\Windows\Start Menu\Programs
12:15:22 PM [Initial Dump] : C:\Users\Public\Desktop
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Local\Microsoft\GameDVR
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Local\Microsoft\Windows\Burn
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Local\Packages\Microsoft.XboxGamingOverlay_8wekyb3d8bbwe\LocalCache
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\LocalLow\Intel\ShaderCache\3bdde7bc39908f12401512fecf0a7e8edb634c6de8e0b0497ef043f33e963fc8
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\LocalLow\Intel\ShaderCache\bf6421a015562f04640f56b4bd743e87ae6d6f6c66cb17f24419f12d5c53cd80
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch\User Pinned
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Windows\Network Shortcuts
12:15:22 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Windows\Printer Shortcuts
12:15:23 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Windows\Start Menu
12:15:23 PM [Initial Dump] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
12:15:23 PM [Initial Dump] : C:\Users\SHAKILA\Desktop
12:15:23 PM [Initial Dump] : C:\Windows\bcastdvr
12:15:23 PM [Initial Dump] : C:\Windows\System32
12:15:23 PM [Initial Dump] : C:\Windows\System32\DriverStore\FileRepository\iastorpinningcomponent.inf_amd64_ff8d0bd695f4bb2e\OptaneShellExt.dll
12:15:23 PM [Initial Dump] : C:\Windows\System32\DriverStore\FileRepository\iastorpinningcomponent.inf_amd64_ff8d0bd695f4bb2e\PinningServiceApi.dll
12:15:23 PM [Initial Dump] : C:\Windows\System32\DriverStore\FileRepository\iastorpinningcomponent.inf_amd64_ff8d0bd695f4bb2e\SharpShell.dll
12:15:23 PM [Initial Dump] : E:\02.Education\01.MSC\Research_Re\Projects\UCSC_Task_Tracker\UCSC_Task_Tracker\UCSC_Task_Tracker\bin\Release

[INFO] Start active monitoring......
```

Figure 25: Experiment 07; Activity Detection Execution



```
[INFO] Start active monitoring......
12:15:39 PM [Distrust Dir] : C:\Users\SHAKILA\AppData\Roaming\Microsoft\Windows\Themes\Transcoded_000
12:15:59 PM [Distrust Dir] : E:\4to
12:15:59 PM [Distrust Dir] : E:\4to
12:15:59 PM [Distrust Dir] : E:\4to
12:16:00 PM [Distrust Dir] : E:\4to
12:16:00 PM [Distrust Dir] : E:\4to
12:16:00 PM [Distrust Dir] : E:\4to
12:16:00 PM [Distrust Dir] : E:\4to
12:16:00 PM [Distrust Dir] : E:\4to
12:16:01 PM [Distrust Dir] : E:\4to
12:16:15 PM [Distrust Dir] : \BaseNamedObjects\C::Users:SHAKILA:AppData:Local:Microsoft:Windows:Explorer:iconcache_48.db!dfMaintainer
12:16:15 PM [Distrust Dir] : C:\Windows\SystemApps\MicrosoftWindows.Client.CBS_cw5n1h2txyewy\FileExplorerExtensions\Assets\images\contrast-standard\theme-dark
12:16:20 PM [Distrust Dir] : E:\04. Projects
12:16:21 PM [Distrust Dir] : E:\04. Projects
12:16:21 PM [Distrust Dir] : E:\04. Projects
12:16:21 PM [Distrust Dir] : E:\04. Projects
12:16:22 PM [Distrust Dir] : E:\04. Projects
12:16:22 PM [Distrust Dir] : E:\04. Projects
12:16:22 PM [Distrust Dir] : E:\04. Projects
12:16:22 PM [Distrust Dir] : E:\04. Projects
12:16:23 PM [Distrust Dir] : E:\04. Projects
12:16:23 PM [Distrust Dir] : E:\04. Projects
12:16:23 PM [Distrust Dir] : E:\04. Projects
12:16:23 PM [Distrust Dir] : E:\04. Projects
12:16:23 PM [Distrust Dir] : E:\04. Projects
12:16:24 PM [Distrust Dir] : E:\04. Projects
12:16:24 PM [Distrust Dir] : E:\04. Projects
12:16:24 PM [Distrust Dir] : E:\04. Projects
12:16:24 PM [Distrust Dir] : E:\04. Projects
12:16:25 PM [Distrust Dir] : E:\04. Projects
12:16:25 PM [Distrust Dir] : E:\04. Projects
12:16:25 PM [Distrust Dir] : E:\04. Projects
12:16:26 PM [Distrust Dir] : E:\04. Projects
12:16:26 PM [Distrust Dir] : E:\04. Projects
12:16:26 PM [Distrust Dir] : E:\04. Projects
```

The implemented program was able to detect the open directories and access folders in real time. However, it is observed that some of the windows operating system files are by default, and they also get captured in this program.

Conclusion

This program is useful for capturing the open directories on the Windows operating system. However, since it captured the default folders they are logged as an initial dump. All the other directory file operations are considered suspicious activities. If the model detects suspicious directory open activity then we can assume it is a breach of trust.

3.3 DEFINED TRUSTWORTHINESS PARAMETERS FOR THE STUDY

The researcher has defined the following trustworthiness parameters for this study based on the literature review and experiments. Based on the discrete image stream inputs and operating system events, these parameters were defined. These parameters will be used to evaluate the examiner's trustworthiness in the scope of this study.

- **Dark lightning condition**

During an e-assessment, it is required to have a clear image of the examinee in the captured image. It is the responsibility of the examinee to maintain an appropriate lightning condition during the e-assessment and if the program is undetectable in the captured image due to bad lighting conditions, it imposes a negative impact on the trustworthiness of the examinee.

- **Unaccepted backgrounds** (The accepted background types have to be defined in the rule set)

During an e-assessment, the examinee's room background has an impact on identifying the examinee in the captured image. The examinee has to adhere to the given rule set and if the examinee violates it by having an accepted background it has an impact on the examinee's trustworthiness.

- **Lack of face visibility in front of the webcam**

The examinee must be available in front of the webcam during the e-assessment period, and it is the examinee's responsibility to provide a clear face view to the webcam during the e-assessment period. If the examinee is not there or has a partial facial image, the examinee's trustworthiness declines.

- **False identity of the examinee**

The identification of the examinee is very important during an e-assessment, and if the identity of the person who takes the assessment does not match the identity of the examinee, it has a significant impact on the examinee's trustworthiness.

- **Multiple people in the image**

Having multiple people around the e-assessment area is a direct violation of the examination code of conduct and if the captured image contains multiple faces it has a huge impact on the trustworthiness of the examinee.

- **Suspicions file, directory opening**

During an assessment, it is not allowed to use unauthorized materials or unauthorized programs. Therefore, if the examinee attempts to perform such an activity it has an impact on the trustworthiness of the examinee.

3.4 DATA COLLECTION

To gather data to test and assess the efficacy of the suggested trust model, the researcher decided to conduct an informative simulated online exam. An online e-assessment comprised of several multiple-choice IQ questions was prepared by the researcher to be used for the e-assessment. During the e-assessment participants, some were asked to attend an IQ test as in real e-assessment and some of them were asked to purposefully imitate cheating and misbehavior. The implemented tooling was executed in the background to collect data during the e-assessment and later a post-feedback questionnaire was given to participants to explain their cheating, and misconduct scenario.

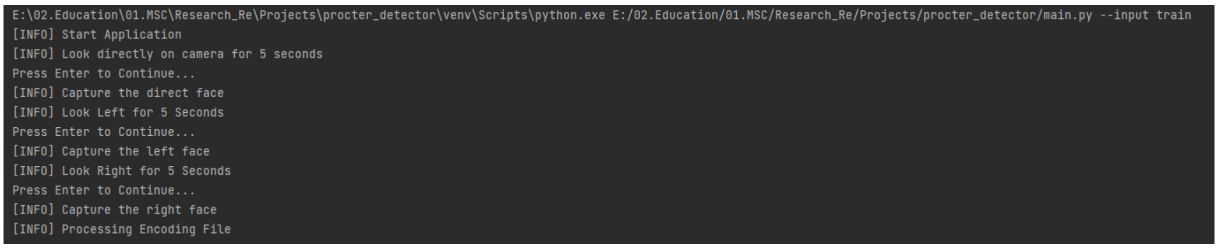
3.4.1 TOOLING AND EQUIPMENT

The researcher developed two software tools to capture the examinee's behavior during an e-assessment that focuses on the stated trustworthiness parameters as a result of the experiments.

1. *The python-based face analysis module*

The first tool (See Appendix B: Python-based Image analysis module) uses picture frame analysis to find suspicious actions. The tool can be used in training and testing modes. Before the e-assessment, the examinee's facial images are captured using a webcam, and a facial encoding model that is unique to the examinee is built. This model will be used during the running mode to perform examinee recognition.

Figure 26: Python-Based Image Analysis Model Training



```
E:\02.Education\01.MSC\Research_Re\Projects\procter_detector\venv\Scripts\python.exe E:/02.Education/01.MSC/Research_Re/Projects/procter_detector/main.py --input train
[INFO] Start Application
[INFO] Look directly on camera for 5 seconds
Press Enter to Continue...
[INFO] Capture the direct face
[INFO] Look Left for 5 Seconds
Press Enter to Continue...
[INFO] Capture the left face
[INFO] Look Right for 5 Seconds
Press Enter to Continue...
[INFO] Capture the right face
[INFO] Processing Encoding File
```

During the running mode, this program uses the web camera to collect a stream of continuous picture frames and then processes them using several trust parameters filters. There are three primary stages in the filtering mechanism: the ML Filtering layer, the Face Detection layer, and the Face Recognition layer.

ML Filtering Layer

As the output of experiment no 04, the researcher has created a convolutional neural network-based model that can distinguish between acceptable proctoring images and unaccepted images taken during the e-assessment process using python. The model was trained with 39,168 real proctoring images which were taken during the different moodle e-assessments.

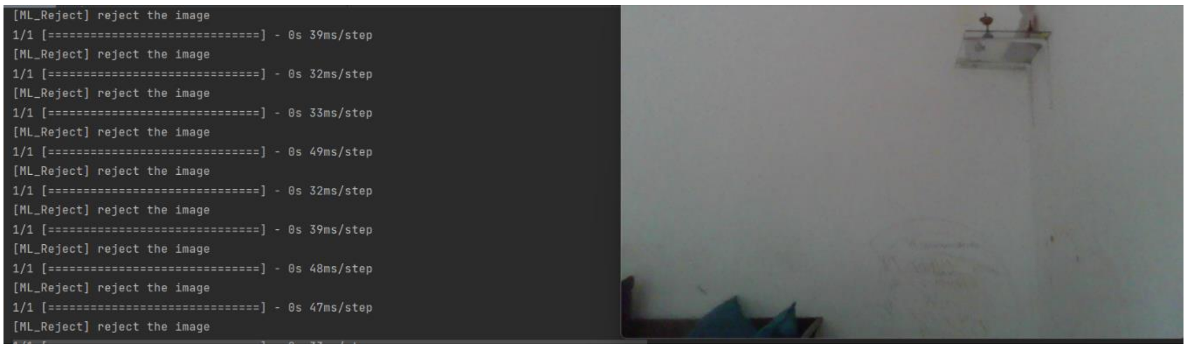
The training unaccepted images contained images such as

- Images contained a part of the face
- Images that do not contain any face
- Images containing multiple faces
- Image with unfocused faces
- Images with faces that overexposed to light
- Images with faces that underexposed to light
- Images with unaccepted backgrounds

And later this model was saved to be used in the developed tool. It is preferable to utilize a pre-trained model since it provides cutting-edge performance without needing GPUs to train the network.

As the first layer of filtering, this module examines each captured image to determine if it should be approved or not. If it is approved the image will be passed through the rest of the filters and if it is unapproved it will be considered a suspicious image.

Figure 27: Python-Based Image Analysis Model ML Filtering

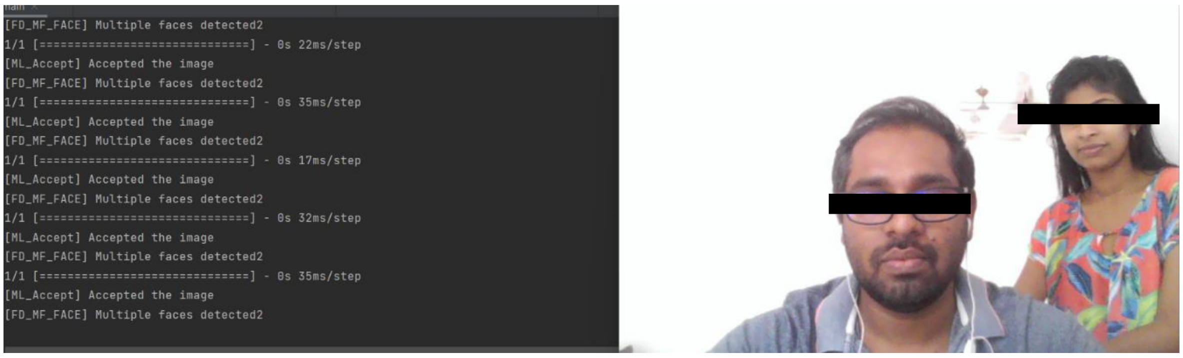


```
[ML_Reject] reject the image
1/1 [==============================] - 0s 39ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 32ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 33ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 49ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 32ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 39ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 48ms/step
[ML_Reject] reject the image
1/1 [==============================] - 0s 47ms/step
[ML_Reject] reject the image
```

Face and multi-face Detection layer

The researcher has successfully constructed a face and multi-face detection layer utilizing a google media pipe-based methodology based on the findings and observations of experiments 1, 2, and 3. This filter looks for faces in the picture that was recorded, and if it finds more than one face, it flags the image as suspicious. Only the approved photos from the first ML filtering layer are applied to this filtering. The next filtering layer will only be reached if this layer finds a single face in the captured image. When many faces are found in a taken image, the image is flagged as suspicious.

Figure 28: Python-Based Image Analysis Model Multiple Face Detection



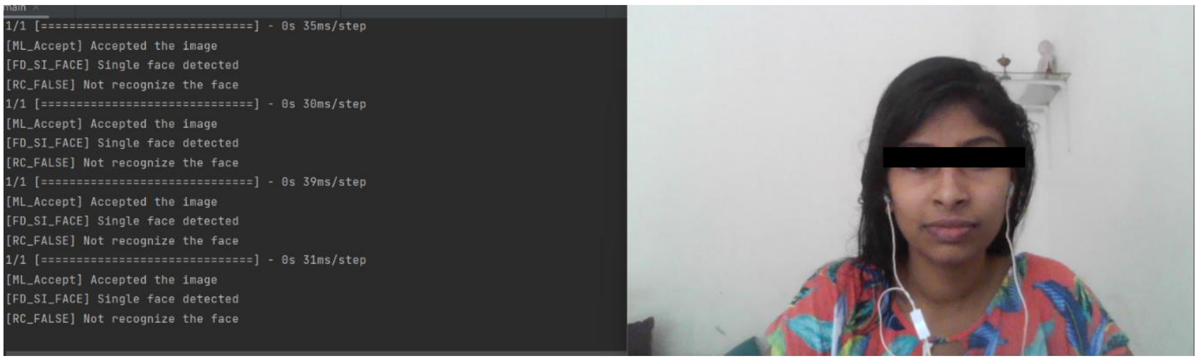
[redacted] [redacted]

```
[FD_MF_FACE] Multiple faces detected2
1/1 [==============================] - 0s 22ms/step
[ML_Accept] Accepted the image
[FD_MF_FACE] Multiple faces detected2
1/1 [==============================] - 0s 35ms/step
[ML_Accept] Accepted the image
[FD_MF_FACE] Multiple faces detected2
1/1 [==============================] - 0s 17ms/step
[ML_Accept] Accepted the image
[FD_MF_FACE] Multiple faces detected2
1/1 [==============================] - 0s 32ms/step
[ML_Accept] Accepted the image
[FD_MF_FACE] Multiple faces detected2
1/1 [==============================] - 0s 35ms/step
[ML_Accept] Accepted the image
[FD_MF_FACE] Multiple faces detected2
```

Facial recognition layer

As per experiment 07, the researcher has implemented a facial recognition layer using python. The researcher has created another Python-based application to capture the face images and build the subject facial encoding model, which is necessary for facial recognition. This facial encoding model is maintained locally on the computer so that the face recognition layer can use it as a source of reference facial encoding. A captured image is flagged as suspicious if the face on it doesn't fit the facial encoding model.

Figure 29: Python-Based Image Analysis Model Facial Recognition



[redacted]

```
1/1 [==============================] - 0s 35ms/step
[ML_Accept] Accepted the image
[FD_SI_FACE] Single face detected
[RC_FALSE] Not recognize the face
1/1 [==============================] - 0s 30ms/step
[ML_Accept] Accepted the image
[FD_SI_FACE] Single face detected
[RC_FALSE] Not recognize the face
1/1 [==============================] - 0s 39ms/step
[ML_Accept] Accepted the image
[FD_SI_FACE] Single face detected
[RC_FALSE] Not recognize the face
1/1 [==============================] - 0s 31ms/step
[ML_Accept] Accepted the image
[FD_SI_FACE] Single face detected
[RC_FALSE] Not recognize the face
```

2. The .net-based OS activity tracking module

The implemented second tool (See Appendix B: .net based OS activity tracking module) uses windows operating system file handling analysis to find suspicious directory access and file open operation in real-time. Initially, the tool asks the examinee to close any open files and directories and it reset the windows operating system file explorer to make sure no file handlers are in place. After that, it starts keeping track of how the OS handles file operations, and if the examinee accesses a suspected directory, the event is immediately flagged in the log and a screenshot will be stored for future reference.

It can be concluded that if there is a flagged item in the log it indicates that the examinee had attempted to perform misconduct. level of academic misconduct is proportional to the number of entries in the log during the given period.

3.5 TRUST MODLE DESIGN

Though out the e-assessment process each captured image will go through defined layers and the operating system file handling monitoring also will get captured as well. At the end of the e-assessment, the log details will be passed through the analysis layer and it will generate the following metrics as the result.

The following rates are calculated based on the **ML filtering layer**

$$\boxed{\boldsymbol{Accepted\ Image\ Rate} = \frac{\sum number\ of\ accepted\ image\ frames}{\sum number\ of\ captured\ image\ frames}}$$

$$\boxed{\boldsymbol{Rejected\ Image\ Rate} = \frac{\sum number\ of\ rejected\ image\ frames}{\sum number\ of\ captured\ image\ frames}}$$

The following rates are calculated based on the **Face and multi-face Detection layer**

$$\boxed{\boldsymbol{Single\ Face\ Detection\ Rate} = \frac{\sum number\ of\ Single\ face\ image\ frames}{\sum number\ of\ accepted\ image\ frames}}$$

$$\boldsymbol{Multiple\ Face\ Detection\ Rate} = \frac{\sum number\ of\ multiple\ face\ image\ frames}{\sum number\ of\ accepted\ image\ frames}$$

$$\boldsymbol{Empty\ Face\ Detection\ Rate} = \frac{\sum number\ of\ empty\ face\ image\ frames}{\sum number\ of\ accepted\ image\ frames}$$

The following rates are calculated based on the **Facial recognition layer**

$$\boldsymbol{True\ Face\ Detection\ Rate} = \frac{\sum number\ of\ true\ face\ image\ frames}{\sum number\ of\ single\ face\ image\ frames}$$

$$\boldsymbol{False\ Face\ Detection\ Rate} = \frac{\sum number\ of\ false\ face\ image\ frames}{\sum number\ of\ single\ face\ image\ frames}$$

$$\boldsymbol{Error\ Face\ Detection\ Rate} = \frac{\sum number\ of\ error\ face\ image\ frames}{\sum number\ of\ single\ face\ image\ frames}$$

The following rates are calculated based on the **OS activity monitoring layer**

$$\boldsymbol{OS\ Activity\ Detection\ Rate} = \frac{\sum number\ oumber\ of\ rounds\ open\ directories\ and\ files\ action\ has\ occered}{\sum number\ of\ validation\ rounds\ for\ open\ directories\ and\ files\ action}$$

From the academic perspective, any action or behavior which violates the e-assessment code of conduct considers an academic offense. Therefore any value greater than zero for Multiple Face Detection Rate(MFDR), False Face Detection Rate(FFDR), and OS Activity Detection Rate(OADR) can be considered a direct violation of the examinee's trust.

The examinee's trust is affected by the Rejected Image Rate (RIR) and Empty Face Detection Rate (RFDR), but their weight in this regard is less than the MFDR, FFDR, and OADR since it may be impacted by the examination environment factors including poor lighting and background objects and not having a person in front of the webcam.

Empty Face Detection Rate(EFDR) also should be considered though because it is capable of capturing empty faces which are not captured through the ML filtering layer.

Error Face Detection Rate(ErFDR) also should be considered though it does not generate much impact on the overall trust because this can happen due to tooling detection errors.

Based on experiment observations and expert opinion researcher came up with a flowing novel Trust Index Model to present trustworthiness as a single unit of measurement by combining the selected multiple parameters.

The weight of each trust parameter corresponding to each filter was determined qualitatively based on the expert opinion and the affecting rules and regulations of the e-assessment process. The following diagram illustrates the high-level design of the Trust Model.

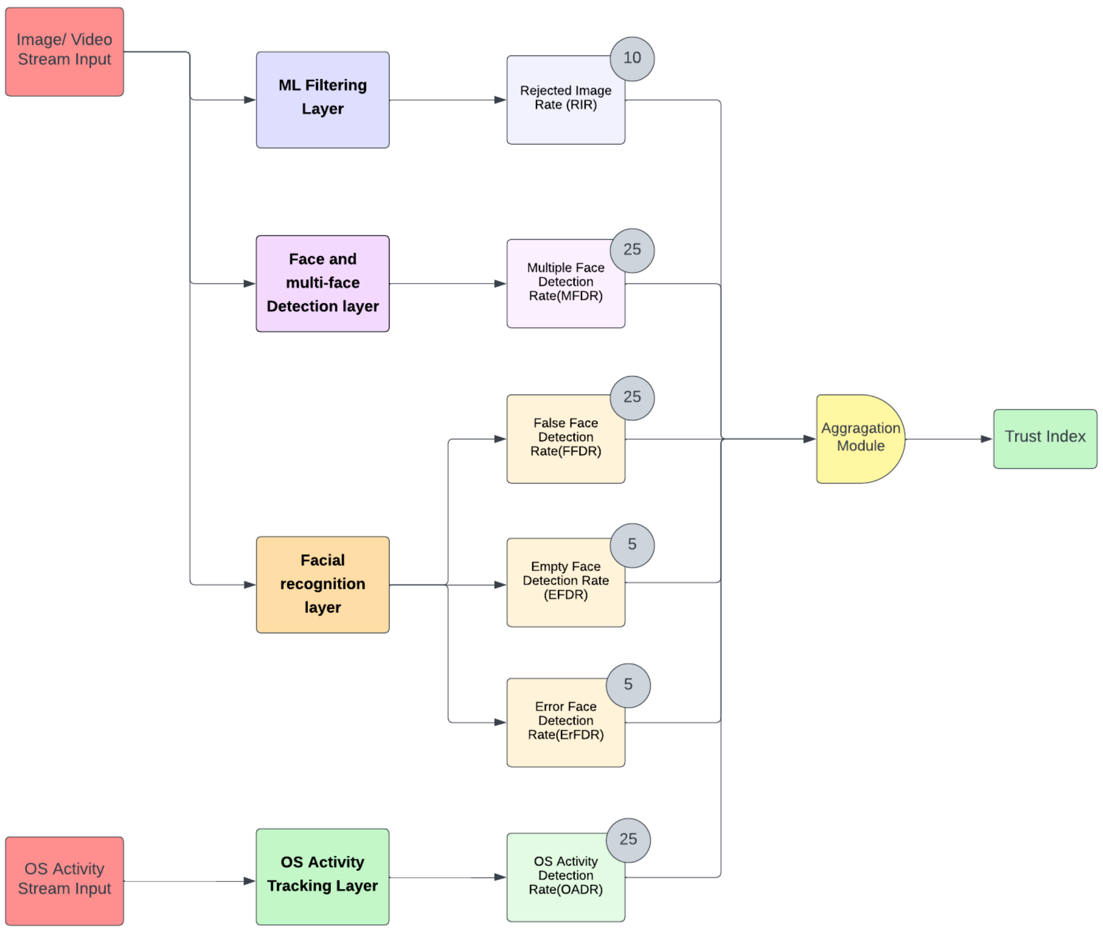
The proposed novel Trust Index is based on a scale from 0 to 100 points, with weights assigned to each of the chosen parameters.

Parameter	Wight	Reason for selecting the wight
Multiple Face Detection Rate(MFDR)	25	Direct violation of serious academic misconduct
False Face Detection Rate(FFDR)	25	Direct violation of serious academic misconduct
OS Activity Detection Rate(OADR)	25	Direct violation of serious academic misconduct
Rejected Image Rate (RIR)	10	Not direct academic misconduct. This happens because the captured image is not matched with the developed ML mode and this model was able to predict unaccepted images with an accuracy of 78% with the used training data set.
Empty Face Detection Rate (EFDR)	5	It is expected to the user to sit in front of the webcam properly. But this is not serious academic misconduct.
Error Face Detection Rate(ErFDR)	5	This might be due to tooling detection errors.

The trust index can be determined using the following equation based on the weights already defined.

$$\boxed{\boldsymbol{Trust\ Index} = 100 - ((MFDR * 25) + (FFDR * 25) + (OADR * 25) + (RIR * 10) + (ErFDR * 5) + (EFDR * 5))}$$

Figure 30: Trust Model Design


Image/ Video Stream Input
ML Filtering Layer
Rejected Image Rate (RIR)
10
Face and multi-face Detection layer
Multiple Face Detection Rate(MFDR)
25
False Face Detection Rate(FFDR)
25
Aggragation Module
Trust Index
Facial recognition layer
Empty Face Detection Rate (EFDR)
5
Error Face Detection Rate(ErFDR)
5
OS Activity Stream Input
OS Activity Tracking Layer
OS Activity Detection Rate(OADR)
25

3.6 MOCK E-ASSESSMENT - IQ TEST

To simulate the assessment researcher created a mock e-asset based on 15 multiple-choice IQ questions (See Appendix A: IQ Test) and examinees were asked to complete it within 15 minutes time duration. Before starting the test, a student facial encoding model was generated using the implemented python based tool. The student was asked to run both a python-based program and a .net-based program in the background and continue with e-assessment. Once they are done with the e-assessment they were asked to exit both programs.

3.7 PARTICIPANTS

During the data-collecting phase of this study, information from fifteen individuals was gathered. All of them participated in this study as volunteers. This research included males and females between the ages of 20 and 35. The researcher has requested participants to complete a volunteer sign-up form that contains the description of the study and the objective of collecting the data since this study involves face data. Before the e-assessment, instructions were provided on how to set up the camera, and how to execute the implemented tooling before starting the e-assessment.

3.8 DATA COLLECTION PROCEDURE

Following suspicious scenarios were identified and different participants were asked to simulate each of these scenarios. Since the e-assessment duration was 15 minutes researcher asked all the participants not to engage in any suspicious activity during the first 5 minutes and then to perform their simulated scenarios. The flowing table shows the simulation scenarios used for the data collection procedure

Table caption: E-assessment Simulation Scenarios

Trust Parameter #	Activity Type	Sub #1	Sub #2	Sub #3	Sub #4	Sub #5	Sub #6	Sub #7	Sub #8	Sub #9	Sub #10	Sub #11	Sub #12	Sub #13
Parameter #1	Proper background	10 min	10 min	10 min	5 min	2 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min
	Dark/ Light background				3 min	8 min								
	No person infront of web cam						3 min	8 min						
Parameter #2	Multiple people in front of the webcam								3 min	8 min				
	A single person in front of the webcam	10 min	10 min	10 min	10 min		5 min	2 min	5 min	2 min	10 min	10 min	10 min	10 min
Parameter #3	Real subject in front of the webcam	10 min	10 min	10 min	10 min		10 min	10 min	10 min	10 min	5 min	2 min	10 min	10 min
	Fake subject in front of the webcam										3 min	8 min		
Parameter #4	Accessing unauthorized materials												3 min	8 min
	Not accessing unauthorized materials	10 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min	10 min	5 min	2 min

According to this matrix subjects, 1,2, and 3 ask to perform the e-assessment genuinely without performing any malpractices.

To test the effects of parameter 01, which had been picked up by the tooling's ML filtering layer, subjects 4 and 5 were instructed to adjust the surrounding lighting. Even though this machine learning filtering layer may identify a variety of background situations dependent on the model that was used, the researcher chose to validate the lightning aspect in this study.

Subjects 6 and 7 simulated the effects of parameter 02 which is a single and multiple face detection filters, with no face in front of the cam scenario, and subjects 8 and 9 simulated the multiple faces in front of the cam scenario.

The impacts of parameter 03, which is obtained from the tool's facial recognition layer, were asked to be simulated by subjects 11 and 12.

Subjects 12 and 13 were asked to simulate parameter 4 which is unauthorized access to the os file operations which is captured from the implemented .net-based application.

Finally, subjects 14 and 15 were asked to randomly perform any suspicious activities during that time and the results were recorded for analysis.

CHAPTER 4

4 EVALUATION

This chapter evaluates the suggested trust model's ability to detect academic dishonesty using selected trust parameters. Multiple Face Detection Rate (MFDR), False Face Detection Rate (FFDR), OS Activity Detection Rate (OADR), Rejected Image Rate (RIR), Rejected Image Rate (RIR), Empty Face Detection Rate (EFDR), Error Face Detection Rate(EFDR) and Trust Index was calculated and the results were qualitatively evaluated with the simulated scenario.

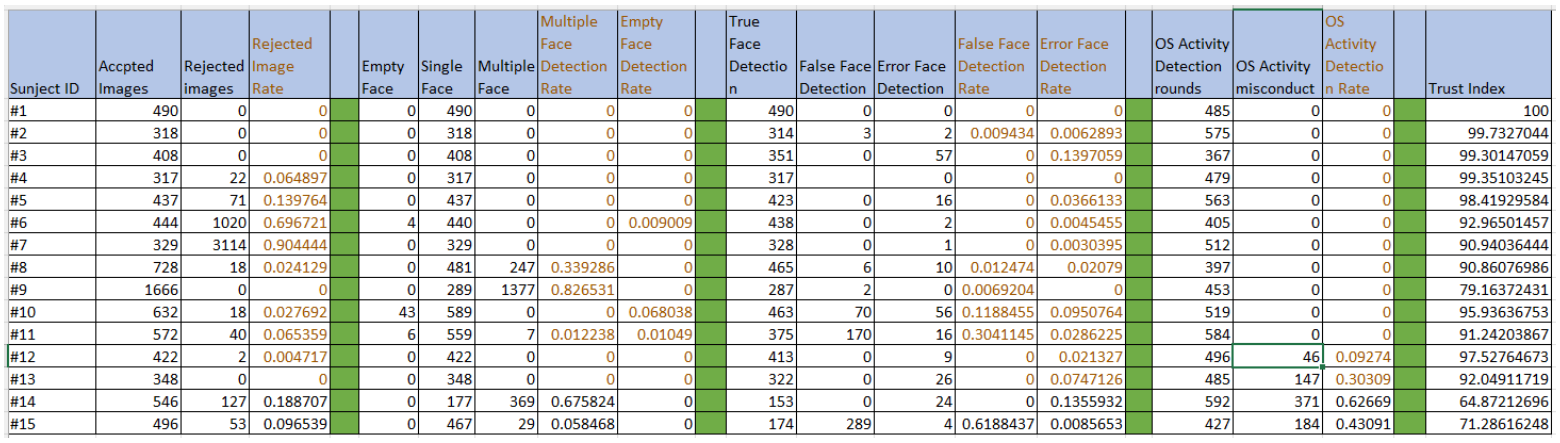
4.1 EVALUATION RESULTS

Following is the information captured by executing the analysis layer of the implemented tool. Then the proposed trust index was calculated for each participant based on the data received from the simulated e-assessment scenarios.

4.2 ANALYSIS

In this study, there were 15 participants, and although 13 of them engaged in simulated e-assessment situations involving various parameters, two of them (Subject IDs 14 and 15) were randomly assigned to engage in academic misconduct during the e-assessment.

Table Caption: Evaluation Results

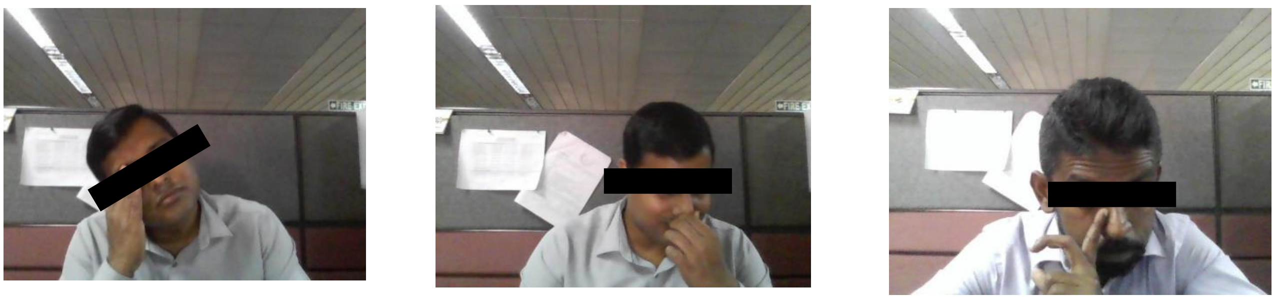


Sunject ID	Accpted Images	Rejected images	Rejected Image Rate		Empty Face	Single Face	Multiple Face	Multiple Face Detection Rate	Empty Face Detection Rate		True Face Detectio n	False Face Detection	Error Face Detection	False Face Detection Rate	Error Face Detection Rate		OS Activity Detection rounds	OS Activity misconduct	OS Activity Detectio n Rate		Trust Index
#1	490	0	0	🟩	0	490	0	0	0	🟩	490	0	0	0	0	🟩	485	0	0	🟩	100
#2	318	0	0	🟩	0	318	0	0	0	🟩	314	3	2	0.009434	0.0062893	🟩	575	0	0	🟩	99.7327044
#3	408	0	0	🟩	0	408	0	0	0	🟩	351	0	57	0	0.1397059	🟩	367	0	0	🟩	99.30147059
#4	317	22	0.064897	🟩	0	317	0	0	0	🟩	317		0	0	0	🟩	479	0	0	🟩	99.35103245
#5	437	71	0.139764	🟩	0	437	0	0	0	🟩	423	0	16	0	0.0366133	🟩	563	0	0	🟩	98.41929584
#6	444	1020	0.696721	🟩	4	440	0	0	0.009009	🟩	438	0	2	0	0.0045455	🟩	405	0	0	🟩	92.96501457
#7	329	3114	0.904444	🟩	0	329	0	0	0	🟩	328	0	1	0	0.0030395	🟩	512	0	0	🟩	90.94036444
#8	728	18	0.024129	🟩	0	481	247	0.339286	0	🟩	465	6	10	0.012474	0.02079	🟩	397	0	0	🟩	90.86076986
#9	1666	0	0	🟩	0	289	1377	0.826531	0	🟩	287	2	0	0.0069204	0	🟩	453	0	0	🟩	79.16372431
#10	632	18	0.027692	🟩	43	589	0	0	0.068038	🟩	463	70	56	0.1188455	0.0950764	🟩	519	0	0	🟩	95.93636753
#11	572	40	0.065359	🟩	6	559	7	0.012238	0.01049	🟩	375	170	16	0.3041145	0.0286225	🟩	584	0	0	🟩	91.24203867
#12	422	2	0.004717	🟩	0	422	0	0	0	🟩	413	0	9	0	0.021327	🟩	496	46	0.09274	🟩	97.52764673
#13	348	0	0	🟩	0	348	0	0	0	🟩	322	0	26	0	0.0747126	🟩	485	147	0.30309	🟩	92.04911719
#14	546	127	0.188707	🟩	0	177	369	0.675824	0	🟩	153	0	24	0	0.1355932	🟩	592	371	0.62669	🟩	64.87212696
#15	496	53	0.096539	🟩	0	467	29	0.058468	0	🟩	174	289	4	0.6188437	0.0085653	🟩	427	184	0.43091	🟩	71.28616248

4.2.1 Simulation Scenario 01 - Subject ID 1,2 and 3

Subject IDs 1, 2, and 3 were instructed to complete the online assessment truthfully without engaging in any type of academic dishonesty. They earned scores of 100, 99.73, and 99.3 on the Trust Index, respectively. Subject IDs 2 and 3 showed error detection and false face detection rates, which are regarded to represent the tooling's error outputs. Some of the erroneous picture instances that were recorded throughout the test are included below.

Figure 31: Error Face Detections


[redacted] [redacted] [redacted]

This shows that the facial recognition layer expects the full face image and therefore it is expected to discourage maintaining different facial poses during the e-assessment to gain the best results. The selection of a low weight for the Error Face detection rate parameter is justifiable because this kind of caption cannot be viewed as academic dishonesty.

4.2.2 Simulation Scenario 02 - Subject ID 4 and 5

Subject IDs 4 and 5 were asked to simulate the dark background scenarios. Subject ID 4 was in the dark background for 3 minutes and it received a Rejected Image Rate of 0.06 and Subject ID 4 was in a dark background for 8 minutes and gave a Rejected Image Rate of 0.13 respectively. The generated trust index for Subject ID 4 was 99.35 and for Subject ID 5 it was 98.41.

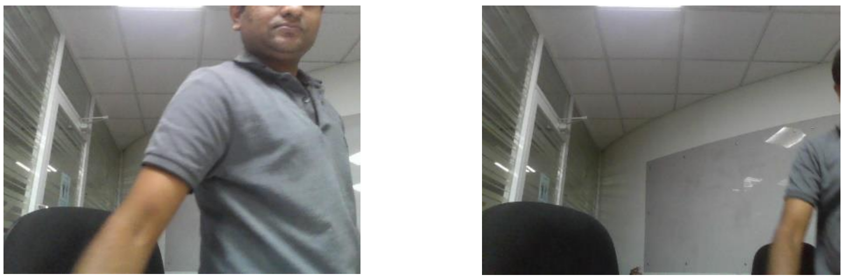
There is a drawback with this rejected image listing, though, in that we cannot say with certainty that this image was disqualified for having a dark background in actual, practical e-assessment scenarios. This is because the filtering is done through an ML filtering layer based on a built-in ML model, and this ML model did not train exclusively using images with dark backgrounds.

Additionally, the reduction of the Trust Index value with the Increase of the Reject image rate happened due to the increased dark background time showing the possibility of quantitatively measuring the trust parameter.

4.2.3 Simulation Scenario 03 - Subject ID 6 and 7

Subject IDs 6 and 7 asked to simulate the No person in front of the webcam scenario. Subject ID 6 was asked to leave the chair for 3 minutes and Subject ID 7 was asked to leave the chair for 8 minutes respectively. Since this is also expected to be captured as part of parameter 01 during the ML filtering layer, we can observe an increase in the Reject Image Rate in both scenarios, and it is directly proportional to the amount of time spent leaving the chair. Additionally, it is shown that the events like getting up from the chair are also captured in the frames that follow.

Figure 32: Leaving the chair



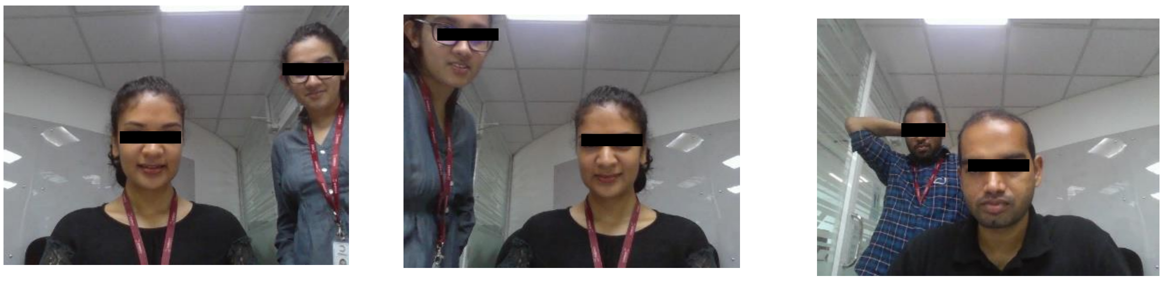
However, we have the same drawback we had on the dark background scenario is also present in this case, because we can not exactly say that the Reject Image rate is increased due to the no face scenario since it used the ML filtering layer for this validation as well. Based on this fact, it is plausible to infer that having many layers trained separately for different attributes will improve the trust index's accuracy.

In this simulation subject ID 6 gained a Trust Index value of 92.96 and Subject ID 7 gained a Trust Index of 90.94 respectively. Although Trust Parameter 1 weighs 10, the final trust index value has changed significantly as a result of the scenario, showing that even though the supplied weight is relatively low, it can have a big impact on the trust value.

4.2.4 Simulation Scenario 04 - Subject ID 8 and 9

In this simulated e-assessment, Subjects ID 8 and 9 were asked to simulate the multiple people in front of the webcam scenario. This simulation's parameter 2, which relates to multiple face detection, weighs 25 because it can be considered serious academic dishonesty. Subject Id 8 was asked to get help from a second person during the e-assessment for 3 minutes and Subject ID 9 for 8 minutes respectively. Following are some of the frames captured during the monitoring process as proofs.

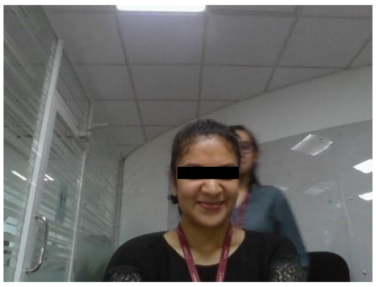
Figure 33: Multiface Detections



When receiving 3-minute help throughout the entire 15-minute e-assessment length during this simulation, Subject ID 8 decreased its Trust Index to 90.86. The Subject ID also decreased its Trust index to 79.16 after obtaining an 8-minute support session for a 15-minute e-assessment. Therefore as in this simulation, with the given weight of 25, it is possible to expect a drastic reduction in the Trust Index in multiple faces in front of a webcam and it is useful for detecting this serious academic misconduct in large-scale examination by evaluating the Trust Index value of each participant.

However, The researcher has observed a False face detection in this simulation in both Subjects. The following image was captured during false face detection.

Figure 34; Second person behind the examinee



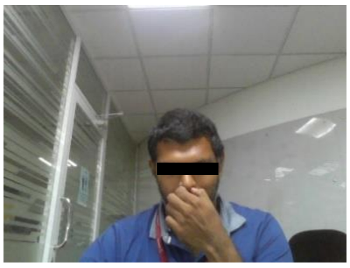
In this scenario, the second person is behind the examinee and the multi-face detection layer was not able to capture it. But the false face detection layer was able to capture it and the false face detection rate was 0.00. but still, it is a false detection it had an impact on the overall trust value. This is an advantage of having multiple parameter layers in the program.

Additionally, it can be seen from the simulation results that the Trust index decline is proportional to the length of academic misbehavior.

4.2.5 Simulation Scenario 05 - Subject ID 10 and 11

In this Simulated e-assessment Subject IDs 10 and 11 were asked to perform the False face scenario which was captured from parameter 3 related to the face recognition layer in the tooling. Subject ID 10 was asked to switch places with a different person for 3 minutes and subject id 11 was asked to switch places with a different person for 8 minutes. During this study Subject ID, 10 gained a Trust Index of 95.93, and Subject ID 11 gained a Trust Index of 91.24. The result shows that the False Face detection rate is proportional to the cheating time duration. The researcher has also observed that in one instance that the real face was detected as a face in the following frame.

Figure 35: Detecting a real face as a false face


[redacted]

This happens because the captured image didn’t have any facial encoding match with the training image set taken properly to the e-assessment. This is a false positive instance and this happened only once during the simulation. However, this can be reduced by increasing the number of frames taken during the training phase by increasing the number of facial encodings.

During this Simulation researcher also observed that there is an Error Face detection rate in the simulation. This happened due to the tooling comparison errors and we have given a weight of 5 to Error face detections in the Trust Index equation. However, the researcher discovered certain image frames that fall into the false face image category when the researcher examined the image frames connected to tooling defects. The trust index may have fallen even lower if those images had been classified as false face images with the weight provided to the false face image rate, which is 25.

The Trust index for Subject ID 10 was dropped from 4.06 even with a 15-minute e-assessment while running a 3-minute cheating simulation, it was noted. Having such a large difference will be beneficial during mass assessments since examiners will be able to identify individuals who have engaged in questionable behavior by looking at the Trust Index.

4.2.6 Simulation Scenario 06 - Subject ID 12 and 13

During this simulation Subject IDs, 12 and 13 were asked to simulate the unauthorized material access scenario which is related to trust parameter 04. Subject ID 12 was asked to access the unauthorized directory for 3 minutes and Subject ID 13 for 8 minutes. Subject ID 12 gained a Trust Index of 97.52 with an OS Activity Detection rate of 0.092 and Subject ID 13 gained a Trust Index of 92.04 respectively with an OS Activity Detection rate of 0.30. We can detect a decline in the Trust Index as a result of unauthorized material access as parameter 4 has been given a weight of 25 due to the seriousness of the academic misconduct. Additionally, it has been noted that the OS Activity detection rate correlates with the amount of time the misconduct has been committed.

4.2.7 Simulation Scenario 08 - Subject ID 14 and 15

Irrespective of the Simulated Scenarios Subject IDs 14 and 15 were asked to perform random misconduct scenarios. Subject ID 14 engaged in academic misconduct during this random experiment by taking breaks during the e-assessment, asking for assistance, and visiting illegal directories on the computer. At the end of the e-assessment Subject ID, 14 gained a Trust Index value of 64.87. According to the findings, the Multiple Face Detection rate and OS activity detection, which we regard to be serious academic misconduct, made the highest contributions to this reduction. Subject Id 15, using the same random situation, decrease its Trust Index to 71.28, with the primary factors being the False Face detection rate and OS activity detection, which were brought on by behaviors like asking someone else for help and accessing prohibited resources. The decrease in the Trust Index is visible in the finished product because they are generally regarded as major academic misconduct.

4.2.8 Overall Simulation Analysis

In this simulated e-assessment, during the first 5 minutes, all the participants were asked to genuinely attend the e-assessment and then perform the cheating scenario. Therefore this result contains both positive and negative behavior during the easement.

Subject IDs 1, 2, and 3 completed the online evaluation without engaging in any improper behavior, earning an average Trust Index of 99.67. This indicates that the errors in the monitoring tooling that was built caused a drop in the Trust Index of 0.32 on average. It is therefore possible to conclude from this data that the tooling error rate is ± 0.32 with this data set. However, it is possible to increase the accuracy of this value by collecting more data from users. Therefore, based on this simulation, it is possible to conclude that the examinee has made an effort to take the online assessment professionally if the Trust Index is higher than 99.67.

As previously stated, the ML filtering layer just records changes in the environment throughout the e-assessment and does not detect any major misconduct. As in the simulation result Subject ID, 4 and 5 simulated this scenario, and according to the given weight value, the minimum value that can be taken for Trust Index is 90 and this means that for the entire duration of the e-assessment the subject is either not in front of the webcam or the background setup is not feasible for an e-assessment. But the likelihood of happening this scenario is low. On the other hand, if the Trust Index is between 90 and 99.67, we can infer that academic dishonesty has occurred.

As in the Simulation results, if it is less than 90, the examinee has engaged in serious academic dishonesty. Examiners can determine the specific type of academic dishonesty by further examining the Rejected Image Rate, Multiple Face Detection Rate, False Face Detection Rate, Os Activity Detection Rates, and Error Face Detection Rates.

Through the use of this model and various threshold values, the examiners will be able to identify those who engaged in examination misconduct during the mass e-examination.

4.3 CONSTRAINTS

To operate implemented tooling, at least two 2.20GHz processors and 8 GB of RAM are required and this tooling only supports Windows Operating systems. On the PC, Python version 3.8 or higher must be installed, and.net version 4.2 is the bare minimum needed.

Due to the constrained timeline and available resources, the simulated E-assessment was performed with 15 participants. However, it is possible to obtain more accurate results by increasing the number of participants in this study.

CHAPTER 5

5 CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

As a result of the digital transformation, e-learning systems gradually replaced traditional educational systems, and e-assessment has also evolved into a crucial component of it. To attain high academic integrity, it is crucial to maintain the examinee's trustworthiness during the e-assessment setting. There are different behavioral and biometric parameters available to detect the examinee's trustworthiness during proctored e-assessments. The evaluation of trustworthiness by combining multiple parameters, however, has only been the subject of a small number of studies in the e-assessment domain.

In a summary, in this study, the researcher has proposed a Trust Index model to quantitatively evaluate the trustworthiness of the examinee during e-assessment with a selected set of trustworthiness parameters. The researcher has experimented with multiple parameters and has determined the optimal parameters that can be used with the least amount of computer power. Also, the researcher has built a set of tools for capturing academic misconduct by using the selected parameters. The Implemented tooling had 4 different filtering layers. The first layer is based on the Machine Learning model built by the researcher by using 39,168 images which is capable of filtering accepted proctoring images captured through the webcam. The second layer is capable of identifying multiple user presence during the assessment. The researcher has built a mechanism to create a facial encoding model before the e-assessment that can be utilized for validation in the filtering process, and the third layer is capable of authenticating the examinee's identification throughout the e-assessment using this encoding model. The fourth layer is capable of identifying unauthorized files and directory access during the assessment. Then an analysis was performed using collected data to generate the trust index according to the proposed model. Finally, a simulated e-assessment with multiple cheating scenarios was undertaken and the results were assessed to assess the validity of the suggested Trust Index model.

The initial goal of this study was to develop a multiparameter-based model to measure examinee trustworthiness in the context of an online e-assessment. The researcher used two input streams to implement this multiparameter model: a discrete image stream and an operating system event. The researcher was able to implement the model using these two inputs by identifying the possible trustworthiness parameters which can be derived from them. However, due to time constraints, other alternative inputs, such as voice steam and video streams, which can be acquired during the e-assistant in a noninvasive manner, were not addressed in this work. Using such input streams, the researcher may have been able to derive more trustworthiness parameters, increasing the model's accuracy.

Though the researcher conducted experiments in this study to establish a trustworthiness measure related to eye gaze detection, it was not included in the final mode construction due to its subjective nature. According to the experiment, a researcher determined that initial calibration is required to perform eye gaze detection and therefore separate data collection related to eye gaze movements is required. Due to the time constraints of this study, the researcher did not incorporate that parameter into the final model. However, future work including this trustworthiness component in the model will boost the model's accuracy.

Due to time constraints, the researcher assigned different weight values to different trustworthiness parameters during model construction. This was done using the subjective opinion of fewer subject matter experts based on the level of academic dishonesty captured through the trustworthiness parameter. As an improvement, these weight values can be derived by polling numerous parties, including students and subject matter experts.

The proposed model was evaluated using a simulated e-assessment with 15 participants. In general, the simulated e-assessment outcome does not reflect the genuine e-assessment conditions. As an improvement, it is beneficial to conduct a model evaluation with an actual e-assessment with a larger number of participants, which will increase the model's reliability.

5.2 FUTURE WORK

Since there aren't many studies undertaken in this area, this study will pave the way for a quantitative assessment of the examinee's trustworthiness in the e-assessment process.

One of the major improvements that can be done in this study is to improve the ML filtering layer to detect multiple attributes separately. The ML filtering layer can presently identify situations like dark or light backdrops, no one in front of the webcam, and hazy backgrounds. This happened because the training dataset contains all these attributes. However, it is possible to construct many ML layers with datasets tailored to each feature, and doing so will improve the Trust Index's accuracy.

At the movement, the captured data analysis and trust index generation both take place after the e-assessment. To conduct a mass e-assessment, it might be possible to do this in real-time during the e- assessment by sending those details to proctors in real-time.

Even though this model was tested on 15 participants, it is possible to increase accuracy by adding more participants and simulating test scenarios as possible future work.

Support for more operating systems and a range of devices, including mobile phones and tablets, would be another future task.

6 REFERENCES

Adakane, D. (2019) ‘What are Haar Features used in Face Detection ?’, 12 November. Available at: https://medium.com/analytics-vidhya/what-is-haar-features-used-in-face-detection-a7e531c8332b (Accessed: 12 September 2022).

Alruwais, N., Wills, G. and Wald, M. (2018) ‘Advantages and Challenges of Using e-Assessment’, *International Journal of Information and Education Technology*, 8(1), pp. 34–37. Available at: https://doi.org/10.18178/ijiet.2018.8.1.1008.

Atoum, Y. *et al.* (2017) ‘Automated Online Exam Proctoring’, *IEEE Transactions on Multimedia*, 19(7), pp. 1609–1624. Available at: https://doi.org/10.1109/TMM.2017.2656064.

Bawarith, R. *et al.* (2017) ‘E-exam Cheating Detection System’, *International Journal of Advanced Computer Science and Applications*, 8(4). Available at: https://doi.org/10.14569/IJACSA.2017.080425.

Bazarevsky, V. *et al.* (2019) ‘BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs’, p. 4.

Ben-Ner, A. and Halldorsson, F. (2010) ‘Trusting and trustworthiness: What are they, how to measure them, and what affects them’, *Journal of Economic Psychology*, 31(1), pp. 64–79. Available at: https://doi.org/10.1016/j.joep.2009.10.001.

Bertran, A. (2018) ‘TeSLA project - Towards an evaluation system reliable online Moodle Moot Spain 2018’, p. 40.

Canu, S. (2019) ‘Eye Gaze detection 2 – Gaze controlled keyboard with Python and Opencv p.4’. Available at: https://pysource.com/2019/01/17/eye-gaze-detection-2-gaze-controlled-keyboard-with-python-and-opencv-p-4/#.

Cassady, J.C. and Gridley, B.E. (2005) ‘The Effects of Online Formative and Summative Assessment on Test Anxiety and Performance’, p. 31.

‘Coronavirus, Social and Physical Distancing and Self-Quarantine’ (2020) *Coronavirus, Social and Physical Distancing and Self-Quarantine*, 15 July. Available at: https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/coronavirus-social-distancing-and-self-quarantine (Accessed: 7 December 2020).

Cuong, N.H. and Hoang, H.T. (2010) ‘Eye-gaze detection with a single WebCAM based on geometry features extraction’, in *2010 11th International Conference on Control Automation Robotics & Vision. Vision (ICARCV 2010)*, Singapore, Singapore: IEEE, pp. 2507–2512. Available at: https://doi.org/10.1109/ICARCV.2010.5707319.

Dixson, D.D. and Worrell, F.C. (2016) ‘Formative and Summative Assessment in the Classroom’, *Theory Into Practice*, 55(2), pp. 153–159. Available at: https://doi.org/10.1080/00405841.2016.1148989.

‘Dlib’ (2022) *Dlib*. Available at: http://dlib.net/.

Dwivedi, D. (2018) ‘Face Detection For Beginners’, *Face Detection For Beginners*, 28 April. Available at: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9 (Accessed: 12 September 2022).

Foster, D. and Layman, H. (2013) ‘Online Proctoring Systems Compared’, p. 12.

Gavali, P. and Banu, J.S. (2019) ‘Deep Convolutional Neural Network for Image Classification on CUDA Platform’, in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Elsevier, pp. 99–122. Available at: https://doi.org/10.1016/B978-0-12-816718-2.00013-0.

Gaytan, J. (2004) ‘Effective Assessment Techniques for Online Instruction’, p. 10.

Gikandi, J.W., Morrow, D. and Davis, N.E. (2011) ‘Online formative assessment in higher education: A review of the literature’, *Computers & Education*, 57(4), pp. 2333–2351. Available at: https://doi.org/10.1016/j.compedu.2011.06.004.

Hansen, J.H.L. and Hasan, T. (2015) ‘Speaker Recognition by Machines and Humans: A tutorial review’, *IEEE Signal Processing Magazine*, 32(6), pp. 74–99. Available at: https://doi.org/10.1109/MSP.2015.2462851.

Hardin, R. (1996) ‘Trustworthiness’, p. 17.

Hewagamage, K.P. and Wikramanayake, G.N. (2011) ‘Designing Formative e-Assessments to Prepare Studentsfor the Summative Assessment in Massive Online Courses’, *International Journal of Information and Education Technology*, pp. 286–291. Available at: https://doi.org/10.7763/IJIET.2011.V1.46.

Hu, S., Jia, X. and Fu, Y. (2018) ‘Research on Abnormal Behavior Detection of Online Examination Based on Image Information’, in *2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Hangzhou: IEEE, pp. 88–91. Available at: https://doi.org/10.1109/IHMSC.2018.10127.

Hylton, K., Levy, Y. and Dringus, L.P. (2016) ‘Utilizing webcam-based proctoring to deter misconduct in online exams’, *Computers & Education*, 92–93, pp. 53–63. Available at: https://doi.org/10.1016/j.compedu.2015.10.002.

Ivanova, M. *et al.* (2018) ‘Enhancing Trust in eAssessment - the TeSLA System Solution’, p. 18.

Janssen, J. *et al.* (2019) ‘E-assessment model & framework’, p. 12.

Joshy, N. *et al.* (2018) ‘MULTI-FACTOR AUTHENTICATION SCHEME FOR ONLINE EXAMINATION’, (3), p. 8.

Kar, A. and Corcoran, P. (2017) ‘A Review and Analysis of Eye-Gaze Estimation Systems, Algorithms and Performance Evaluation Methods in Consumer Platforms’, *IEEE Access*, 5, pp. 16495–16519. Available at: https://doi.org/10.1109/ACCESS.2017.2735633.

Kazemi, V. and Sullivan, J. (2014) ‘One millisecond face alignment with an ensemble of regression trees’, in *2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH: IEEE, pp. 1867–1874. Available at: https://doi.org/10.1109/CVPR.2014.241.

Levi, M. and Stoker, L. (2000) ‘Political Trust and Trustworthiness’, *Annual Review of Political Science*, 3(1), pp. 475–507. Available at: https://doi.org/10.1146/annurev.polisci.3.1.475.

Li, Z. *et al.* (2021) ‘A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects’, *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21. Available at: https://doi.org/10.1109/TNNLS.2021.3084827.

Liu, Y. and Wu, Y. (2010) ‘A Survey on Trust and Trustworthy E-learning System’, in *2010 International Conference on Web Information Systems and Mining. 2010 International Conference on Web Information Systems and Mining (WISM)*, Sanya, China: IEEE, pp. 118–122. Available at: https://doi.org/10.1109/WISM.2010.62.

Lu, D. and Yan, L. (2021) ‘Face Detection and Recognition Algorithm in Digital Image Based on Computer Vision Sensor’, *Journal of Sensors*. Edited by H. Lv, 2021, pp. 1–16. Available at: https://doi.org/10.1155/2021/4796768.

Maniar, S. *et al.* (2021) ‘Automated Proctoring System using Computer Vision Techniques’, in *2021 International Conference on System, Computation, Automation and Networking (ICSCAN). 2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, Puducherry, India: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICSCAN53069.2021.9526411.

Marsh, S.P. (1994) *Formalising Trust as a Computational Concept*.

‘MediaPipe Face Detection’ (2020). Available at: https://google.github.io/mediapipe/solutions/face_detection.html#resources (Accessed: 25 September 2022).

Musa, M.A. and Othman, M.S. (2012) ‘CRITICAL SUCCESS FACTOR IN E-LEARNING: AN EXAMINATION OF TECHNOLOGY AND STUDENT FACTORS’, 3(2), p. 10.

Nazar, A. (2003) *Synthesis & Simulation of Mouse Dynamics*.

Nigam, A. *et al.* (2021) ‘A Systematic Review on AI-based Proctoring Systems: Past, Present and Future’, *Education and Information Technologies*, 26(5), pp. 6421–6445. Available at: https://doi.org/10.1007/s10639-021-10597-x.

Ojo, S.O., Zuva, T. and Ngwira, S.M. (2015) ‘Survey of biometric authentication for e-assessment’, in *2015 International Conference on Computing, Communication and Security (ICCCS). 2015 International Conference on Computing, Communication and Security (ICCCS)*, Pointe aux Piments, Mauritius: IEEE, pp. 1–4. Available at: https://doi.org/10.1109/CCCS.2015.7374150.

Okada, A. *et al.* (2019) ‘Pedagogical approaches for e-assessment with authentication and authorship verification in Higher Education’, *British Journal of Educational Technology*, 50(6), pp. 3264–3282. Available at: https://doi.org/10.1111/bjet.12733.

‘Online Assesment in Schools-Challenges and Solutions’ (2020). Available at: https://www.skoolbeep.com/blog/online-assessment-in-schools-challenges-and-solutions/ (Accessed: 16 December 2021).

‘OpenCV’ (2022) *Cascade Classifier Training*. Available at: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (Accessed: 24 February 2022).

O’Shea, K. and Nash, R. (2015) ‘An Introduction to Convolutional Neural Networks’. arXiv. Available at: http://arxiv.org/abs/1511.08458 (Accessed: 17 November 2022).

Raj, R.S.V., Narayanan, S.A. and Bijlani, K. (2015) ‘Heuristic-Based Automatic Online Proctoring System’, in *2015 IEEE 15th International Conference on Advanced Learning Technologies*. *2015 IEEE 15th International Conference on Advanced Learning Technologies (ICALT),* Hualien, Taiwan: IEEE, pp. 458–459. Available at: https://doi.org/10.1109/ICALT.2015.127.

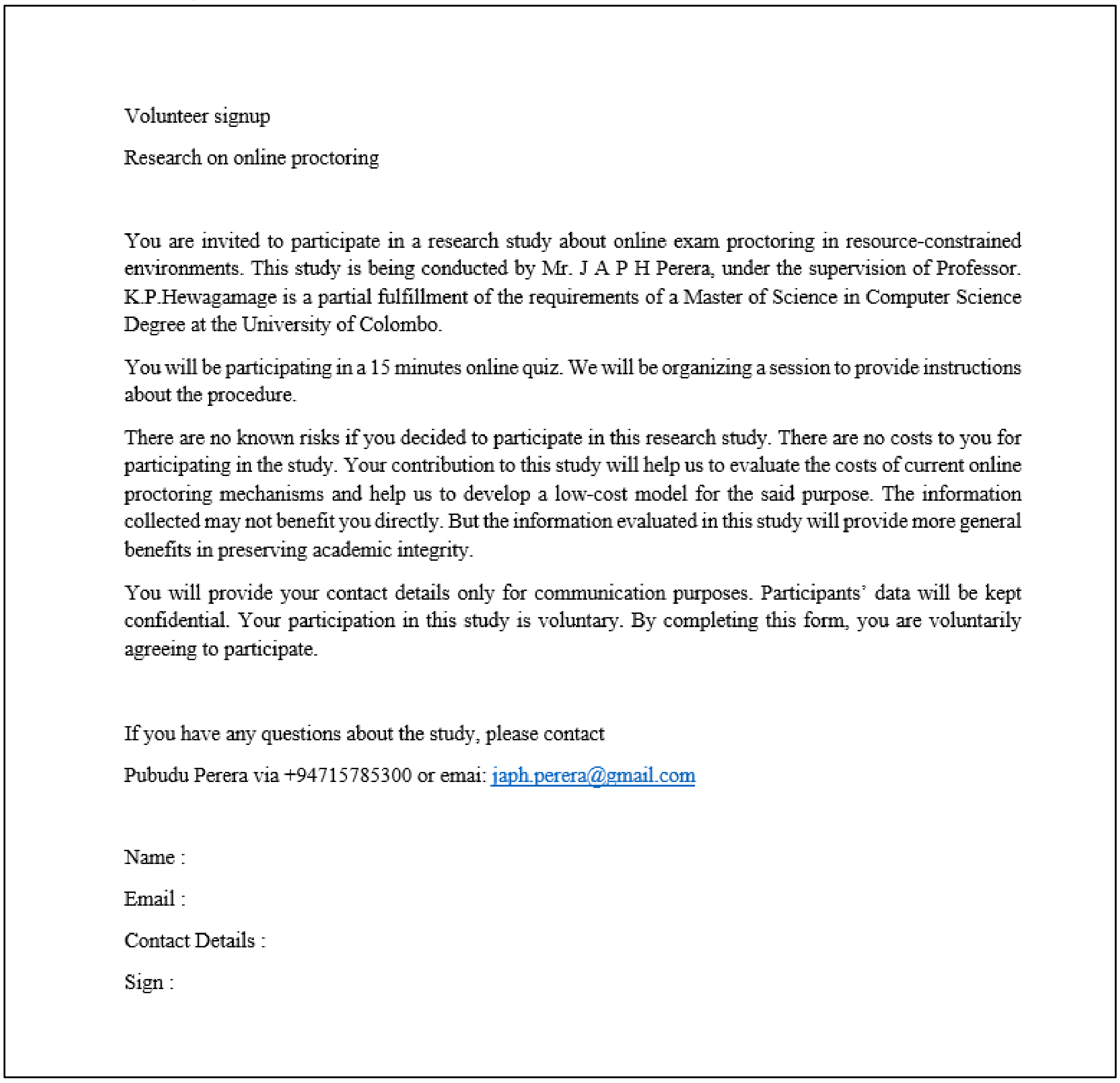
‘The COVID-19 pandemic has changed education forever’ (2020), 29 April. Available at: https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning (Accessed: 7 September 2020).

‘what is image classification in deep learning?’ (2021). Available at: https://www.thinkautomation.com/eli5/eli5-what-is-image-classification-in-deep-learning/.

‘Why Is Assessment Important?’ (2008). Available at: https://www.edutopia.org/assessment-guide-importance (Accessed: 13 October 2021).

APPENDIX A: Data Collection

Volunteer Signup Form



Volunteer signup

Research on online proctoring

You are invited to participate in a research study about online exam proctoring in resource-constrained environments. This study is being conducted by Mr. J A P H Perera, under the supervision of Professor. K.P.Hewagamage is a partial fulfillment of the requirements of a Master of Science in Computer Science Degree at the University of Colombo.

You will be participating in a 15 minutes online quiz. We will be organizing a session to provide instructions about the procedure.

There are no known risks if you decided to participate in this research study. There are no costs to you for participating in the study. Your contribution to this study will help us to evaluate the costs of current online proctoring mechanisms and help us to develop a low-cost model for the said purpose. The information collected may not benefit you directly. But the information evaluated in this study will provide more general benefits in preserving academic integrity.

You will provide your contact details only for communication purposes. Participants' data will be kept confidential. Your participation in this study is voluntary. By completing this form, you are voluntarily agreeing to participate.

If you have any questions about the study, please contact

Pubudu Perera via +94715785300 or emai: japh.perera@gmail.com

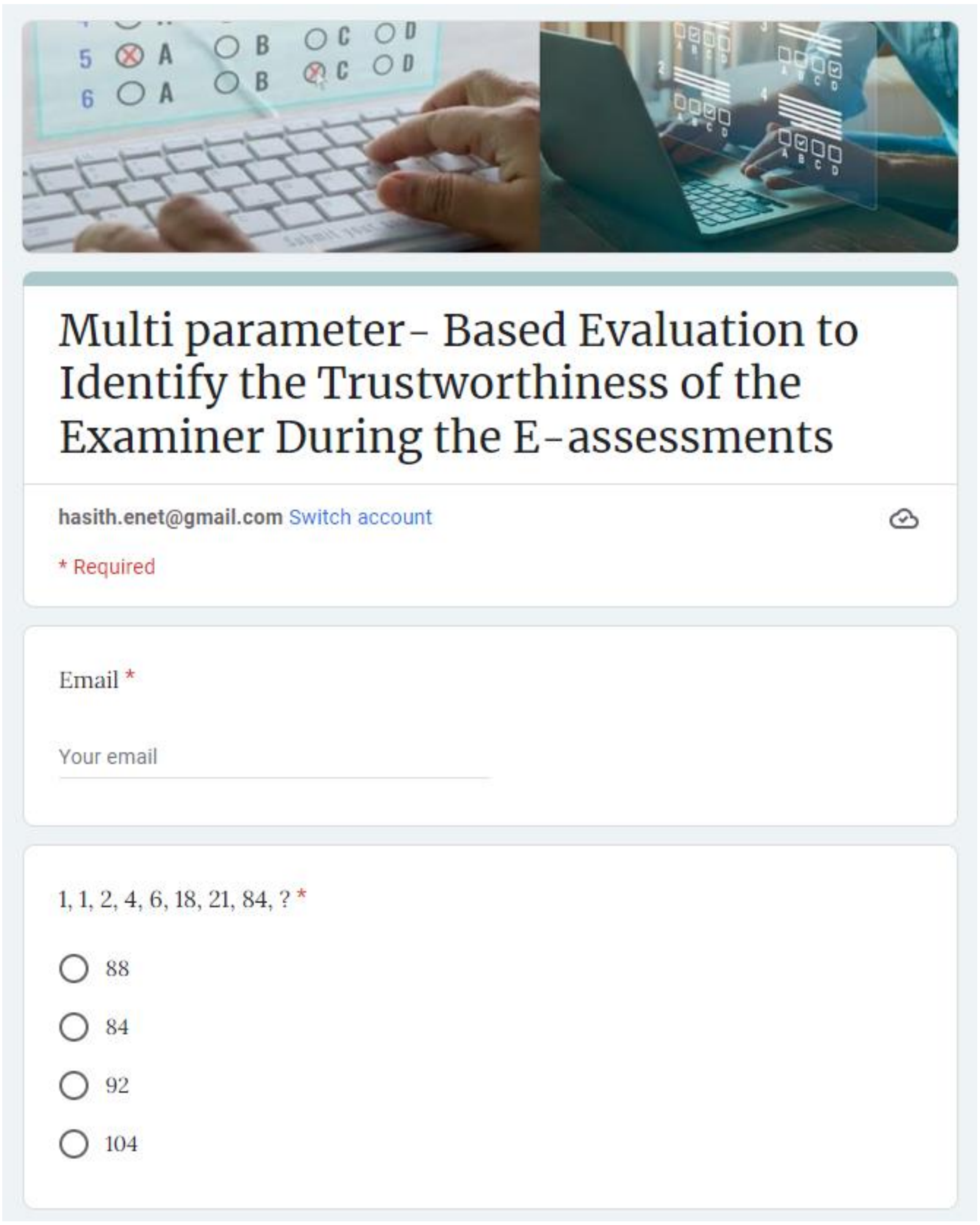
Name :

Email :

Contact Details :

Sign :

IQ Test



Multi parameter- Based Evaluation to Identify the Trustworthiness of the Examiner During the E-assessments

hasith.enet@gmail.com Switch account

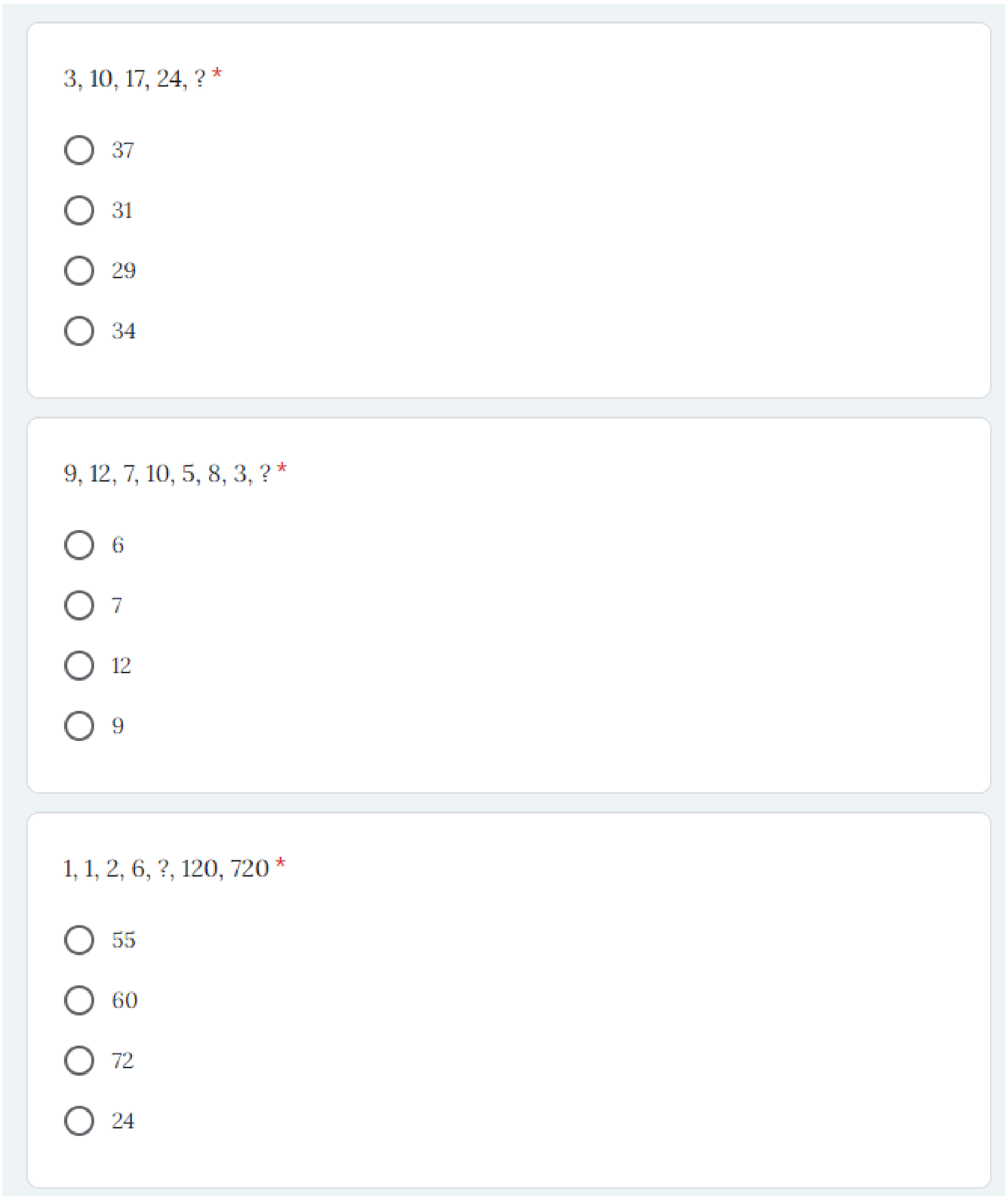
* Required

Email *

Your email

1, 1, 2, 4, 6, 18, 21, 84, ? *

- ○ 88
- ○ 84
- ○ 92
- ○ 104



3, 10, 17, 24, ? *

○ 37

○ 31

○ 29

○ 34

9, 12, 7, 10, 5, 8, 3, ? *

○ 6

○ 7

○ 12

○ 9

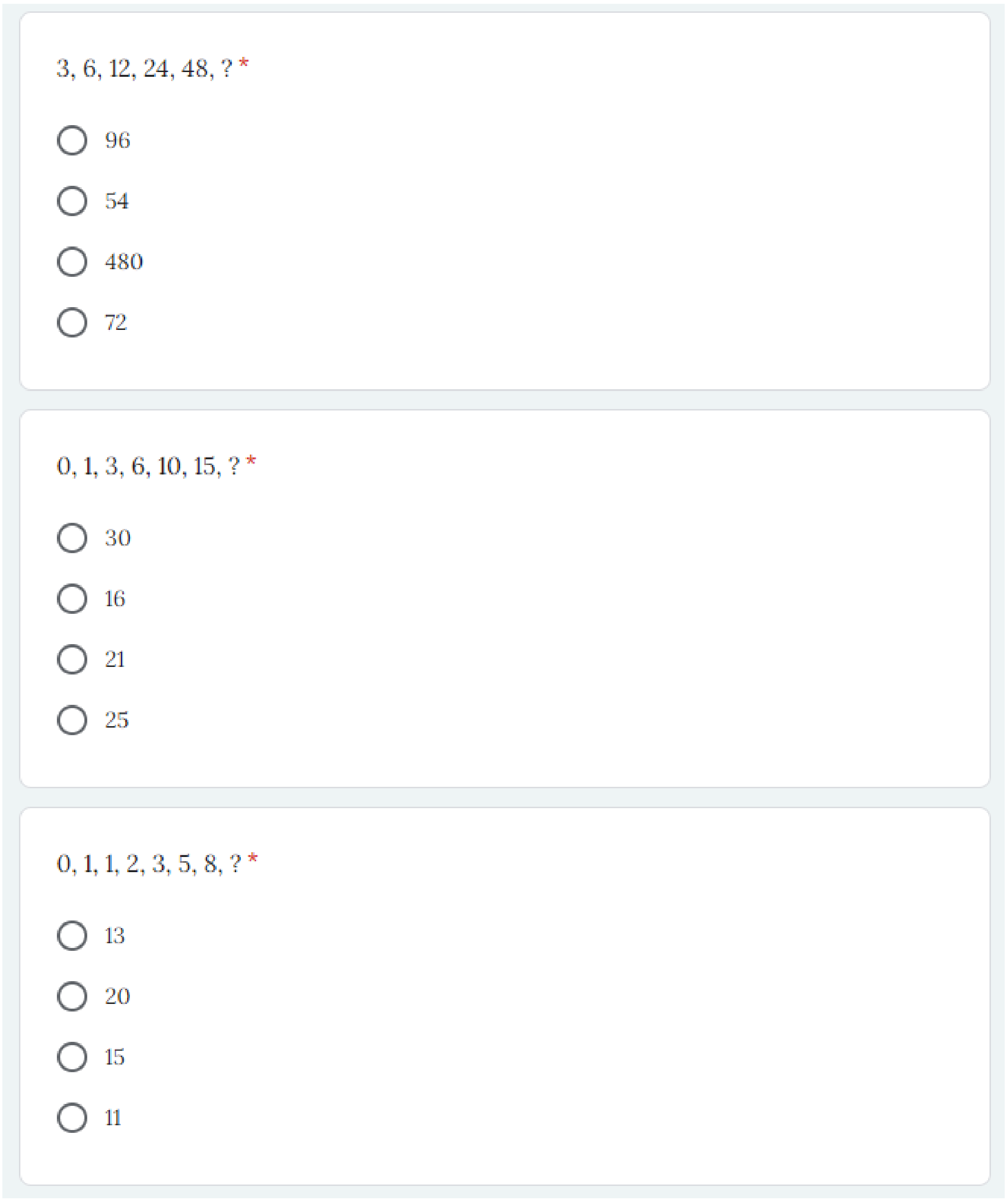
1, 1, 2, 6, ?, 120, 720 *

○ 55

○ 60

○ 72

○ 24



3, 6, 12, 24, 48, ? *

○ 96

○ 54

○ 480

○ 72

0, 1, 3, 6, 10, 15, ? *

○ 30

○ 16

○ 21

○ 25

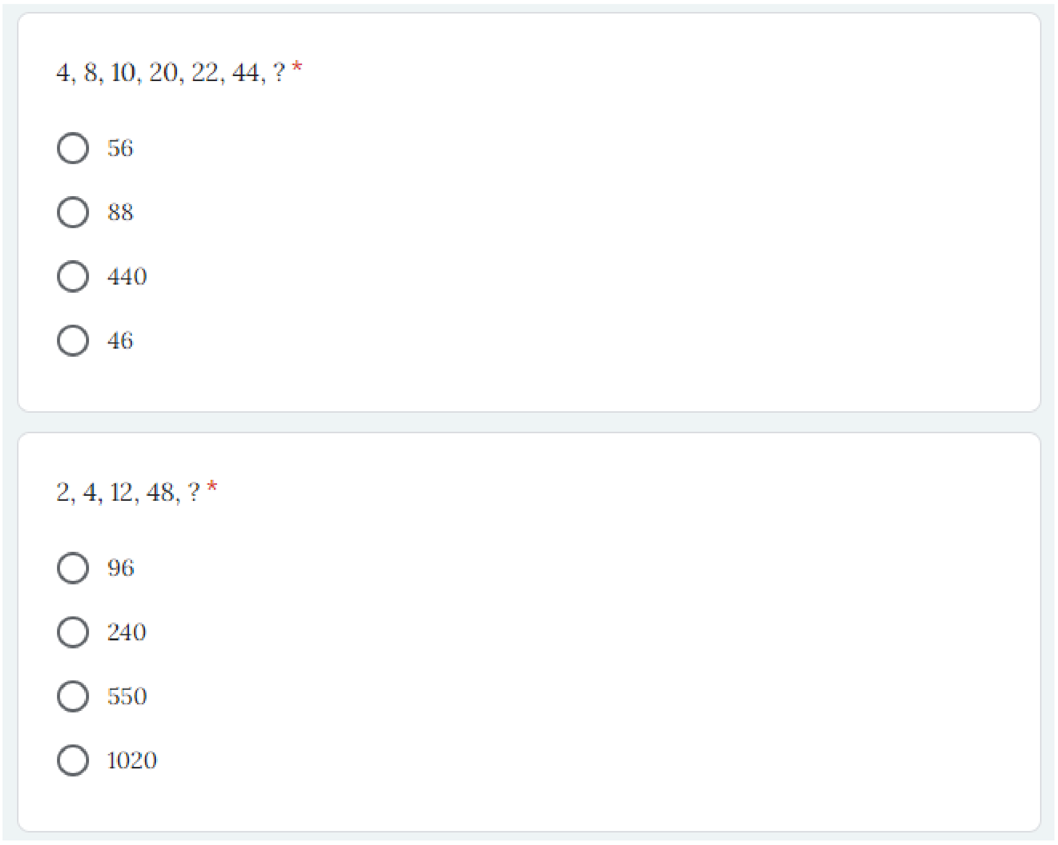
0, 1, 1, 2, 3, 5, 8, ? *

○ 13

○ 20

○ 15

○ 11



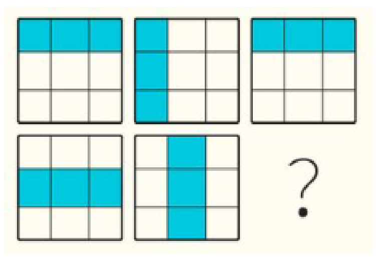
4, 8, 10, 20, 22, 44, ? *

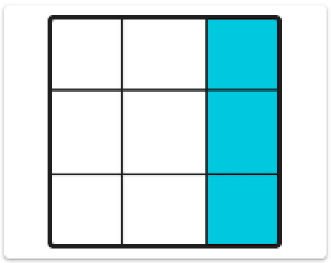
- ○ 56
- ○ 88
- ○ 440
- ○ 46

2, 4, 12, 48, ? *

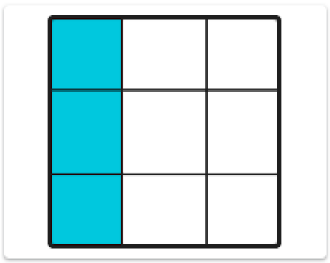
- ○ 96
- ○ 240
- ○ 550
- ○ 1020

*

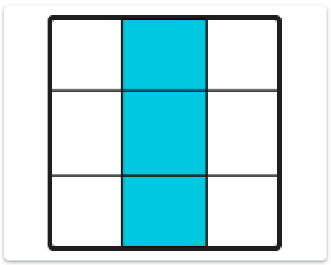




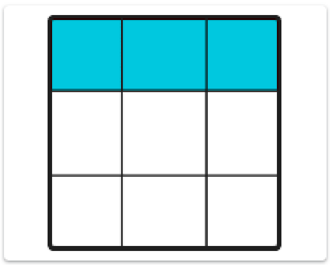
◯ Option 1



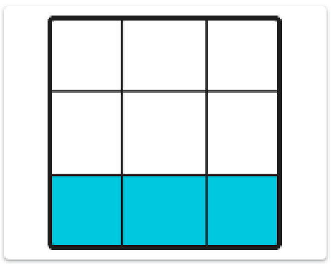
◯ Option 2



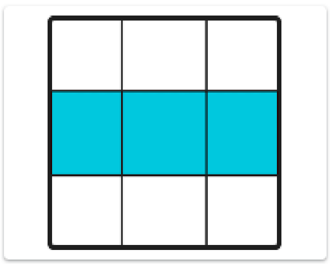
◯ Option 3



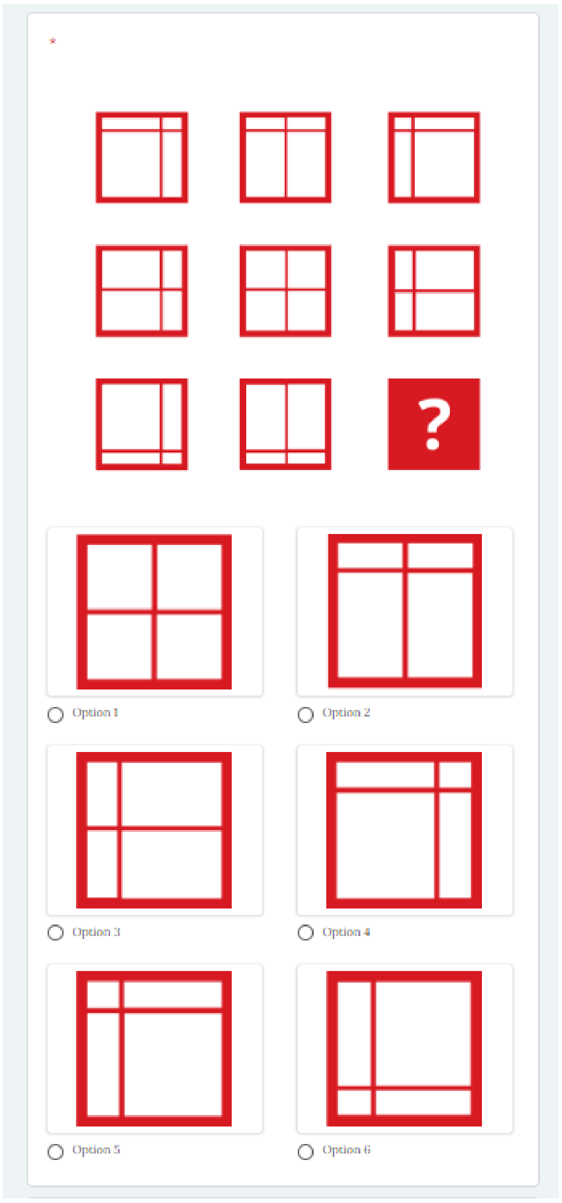
◯ Option 4



◯ Option 5

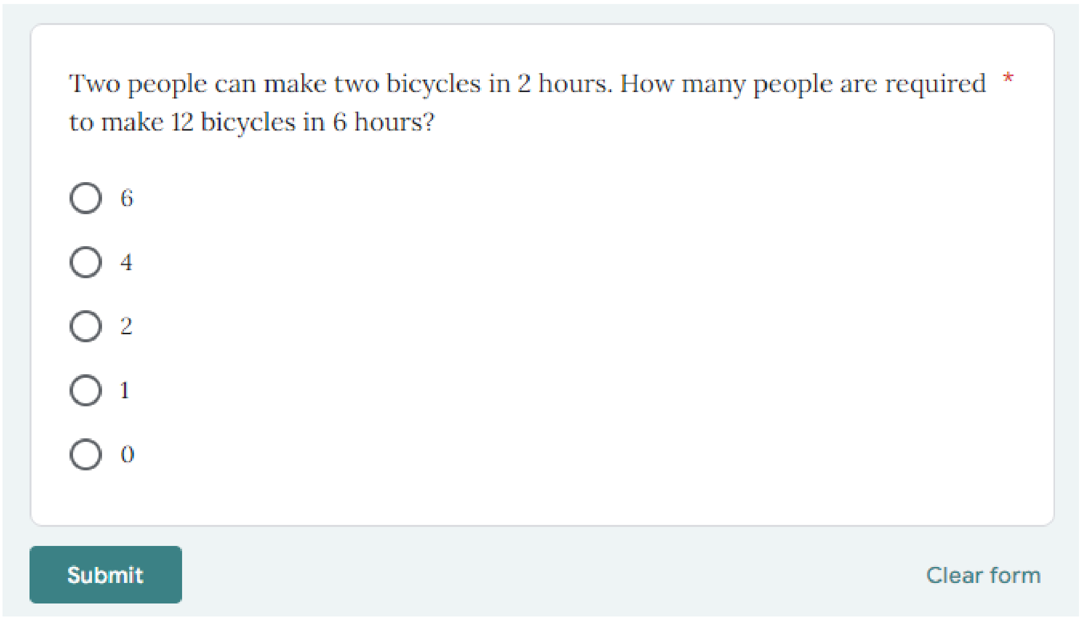


◯ Option 6



*

○ Option 1
○ Option 2
○ Option 3
○ Option 4
○ Option 5
○ Option 6

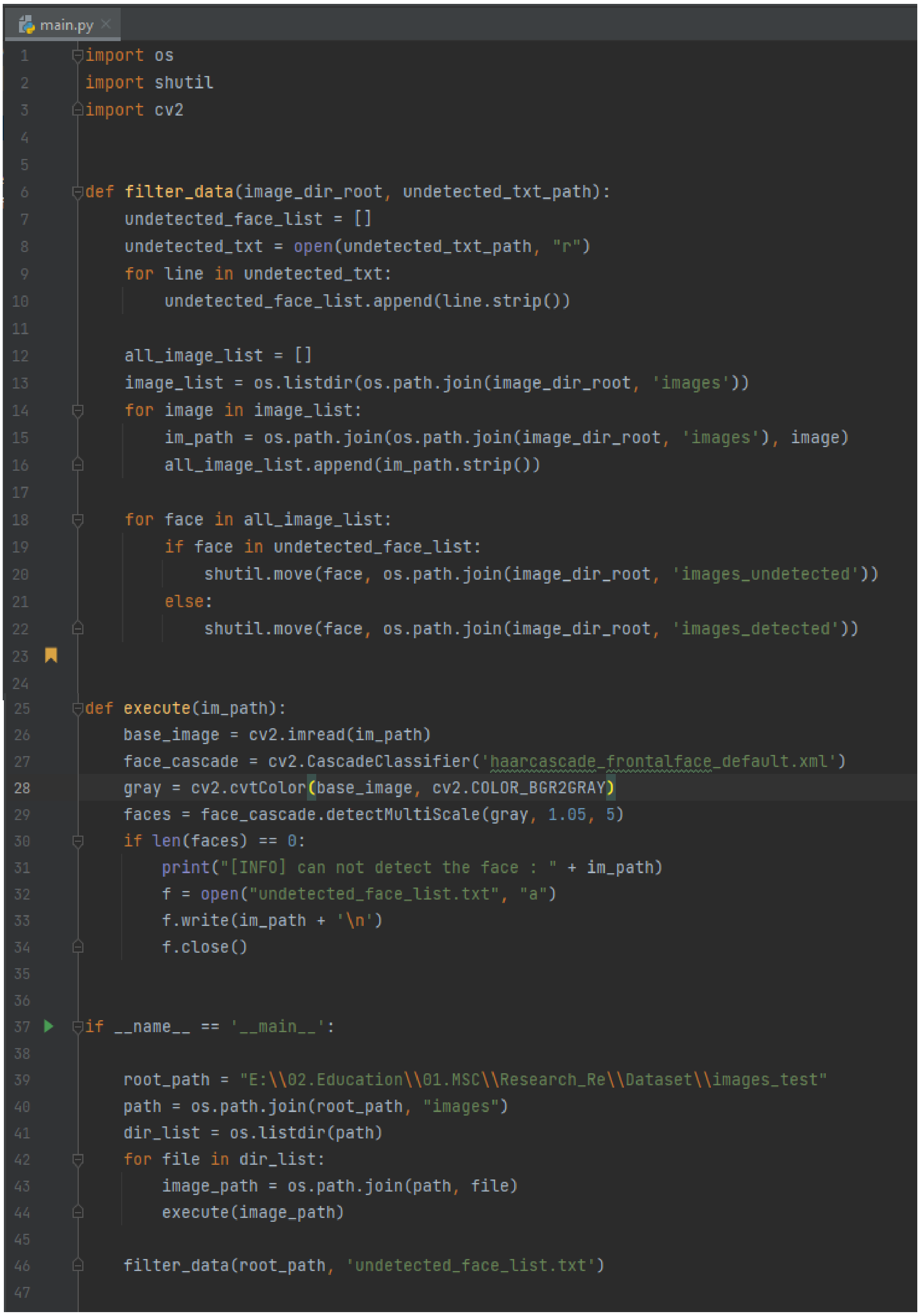


Two people can make two bicycles in 2 hours. How many people are required to make 12 bicycles in 6 hours? *

○ 6
○ 4
○ 2
○ 1
○ 0

APPENDIX B: Source Codes

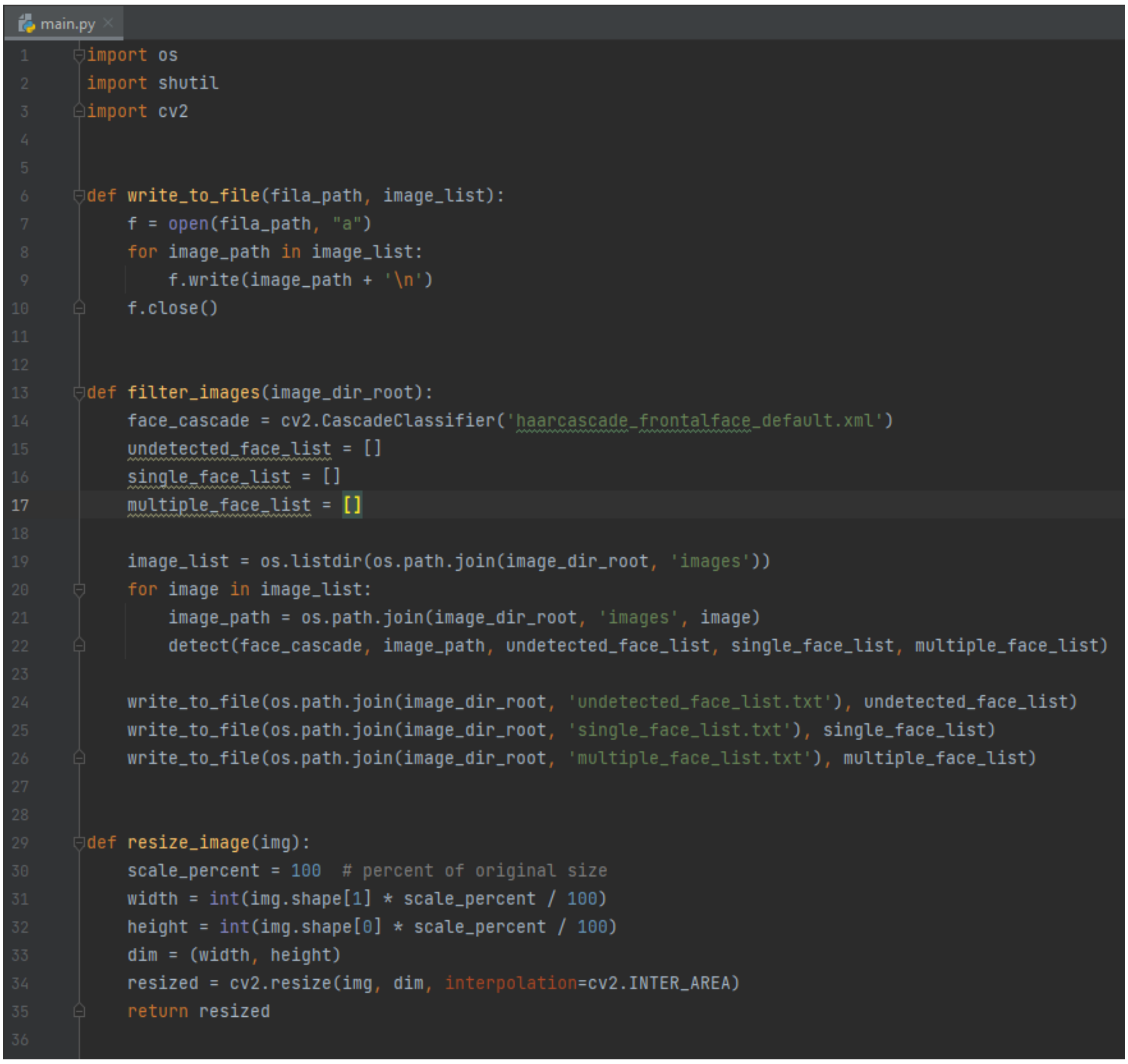
OpenCV Based Face Detection



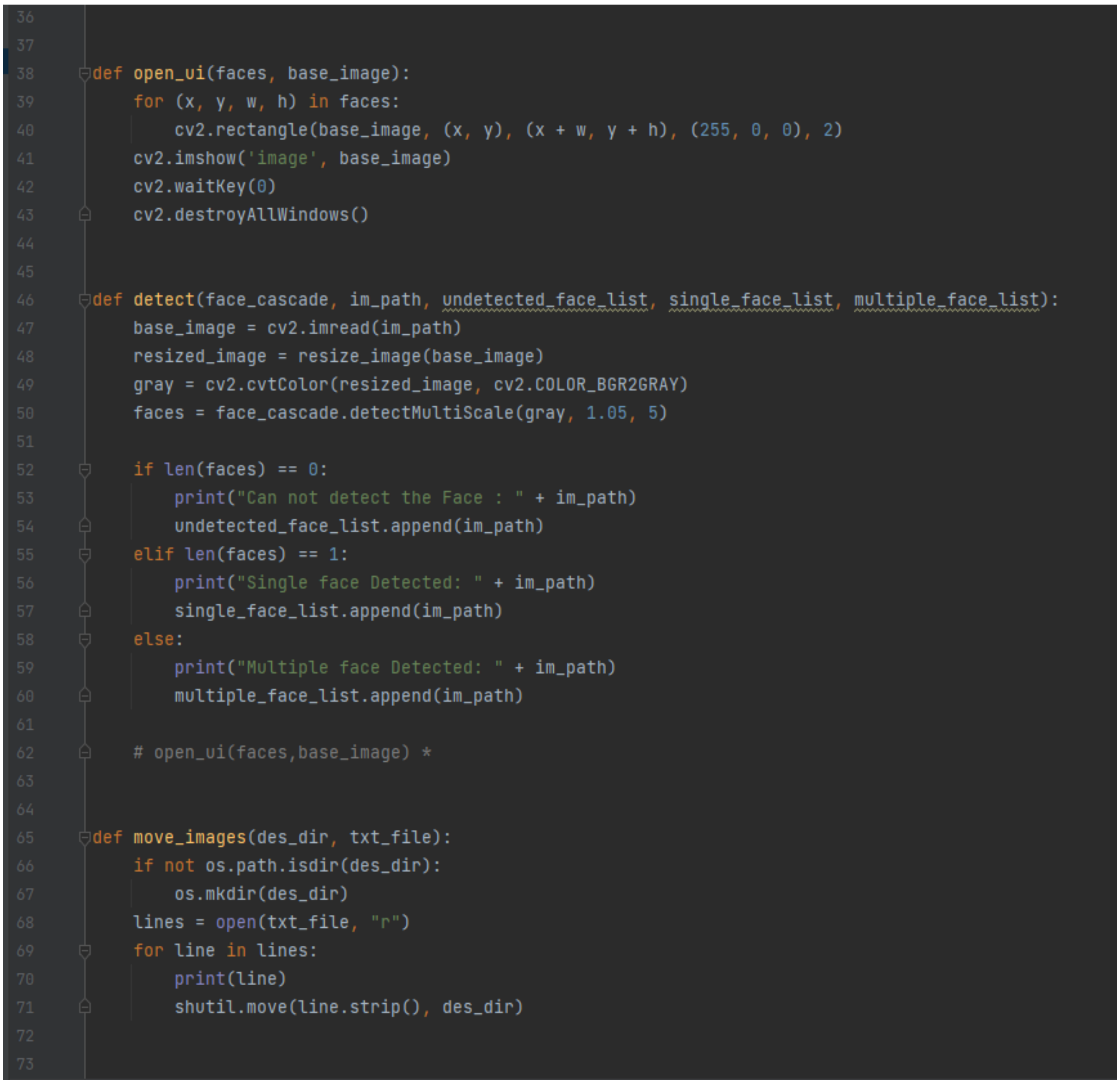
main.py

```python
import os
import shutil
import cv2


def filter_data(image_dir_root, undetected_txt_path):
    undetected_face_list = []
    undetected_txt = open(undetected_txt_path, "r")
    for line in undetected_txt:
        undetected_face_list.append(line.strip())

    all_image_list = []
    image_list = os.listdir(os.path.join(image_dir_root, 'images'))
    for image in image_list:
        im_path = os.path.join(os.path.join(image_dir_root, 'images'), image)
        all_image_list.append(im_path.strip())

    for face in all_image_list:
        if face in undetected_face_list:
            shutil.move(face, os.path.join(image_dir_root, 'images_undetected'))
        else:
            shutil.move(face, os.path.join(image_dir_root, 'images_detected'))


def execute(im_path):
    base_image = cv2.imread(im_path)
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(base_image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.05, 5)
    if len(faces) == 0:
        print("[INFO] can not detect the face : " + im_path)
        f = open("undetected_face_list.txt", "a")
        f.write(im_path + '\n')
        f.close()


if __name__ == '__main__':

    root_path = "E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\images_test"
    path = os.path.join(root_path, "images")
    dir_list = os.listdir(path)
    for file in dir_list:
        image_path = os.path.join(path, file)
        execute(image_path)

    filter_data(root_path, 'undetected_face_list.txt')

```

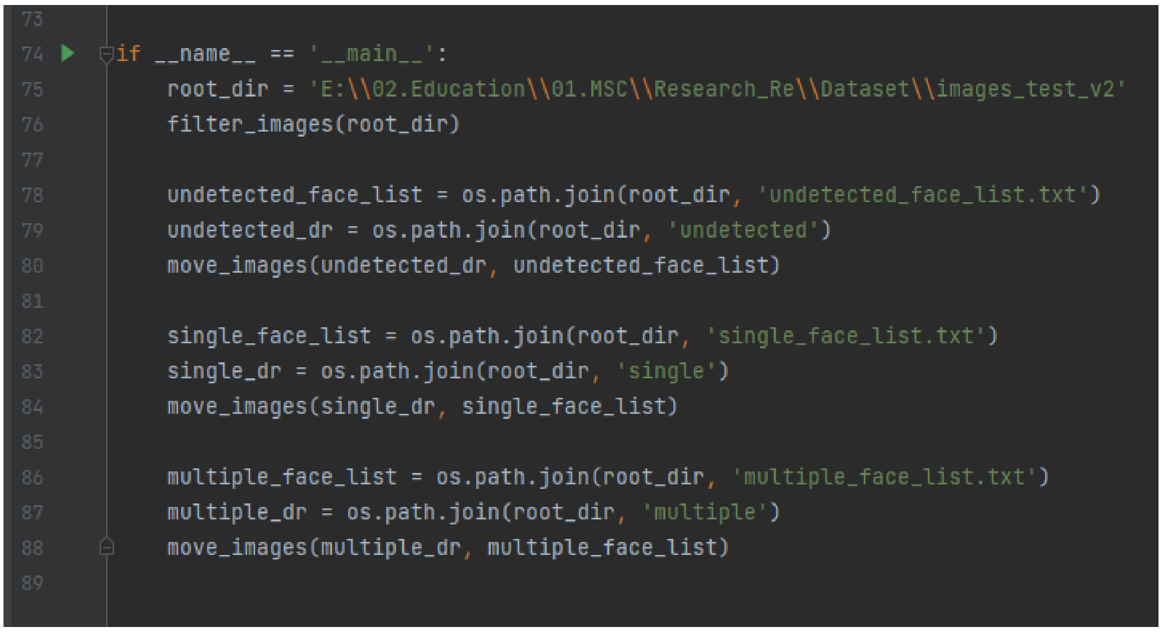
OpenCV-Based Face Multiple Face Detection



```python
import os
import shutil
import cv2


def write_to_file(fila_path, image_list):
    f = open(fila_path, "a")
    for image_path in image_list:
        f.write(image_path + '\n')
    f.close()


def filter_images(image_dir_root):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    undetected_face_list = []
    single_face_list = []
    multiple_face_list = []

    image_list = os.listdir(os.path.join(image_dir_root, 'images'))
    for image in image_list:
        image_path = os.path.join(image_dir_root, 'images', image)
        detect(face_cascade, image_path, undetected_face_list, single_face_list, multiple_face_list)

    write_to_file(os.path.join(image_dir_root, 'undetected_face_list.txt'), undetected_face_list)
    write_to_file(os.path.join(image_dir_root, 'single_face_list.txt'), single_face_list)
    write_to_file(os.path.join(image_dir_root, 'multiple_face_list.txt'), multiple_face_list)


def resize_image(img):
    scale_percent = 100  # percent of original size
    width = int(img.shape[1] * scale_percent / 100)
    height = int(img.shape[0] * scale_percent / 100)
    dim = (width, height)
    resized = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
    return resized

```

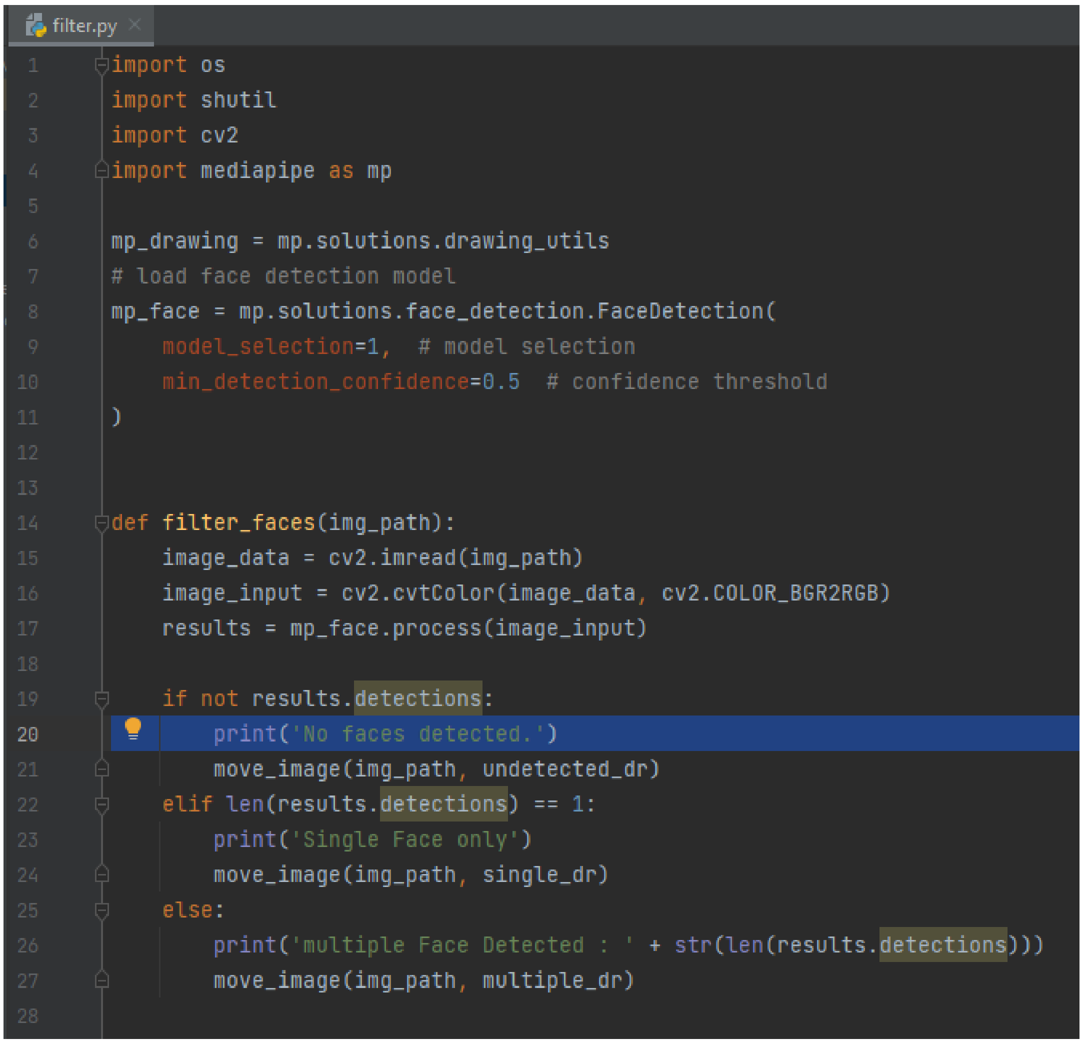


```python


def open_ui(faces, base_image):
    for (x, y, w, h) in faces:
        cv2.rectangle(base_image, (x, y), (x + w, y + h), (255, 0, 0), 2)
    cv2.imshow('image', base_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def detect(face_cascade, im_path, undetected_face_list, single_face_list, multiple_face_list):
    base_image = cv2.imread(im_path)
    resized_image = resize_image(base_image)
    gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.05, 5)

    if len(faces) == 0:
        print("Can not detect the Face : " + im_path)
        undetected_face_list.append(im_path)
    elif len(faces) == 1:
        print("Single face Detected: " + im_path)
        single_face_list.append(im_path)
    else:
        print("Multiple face Detected: " + im_path)
        multiple_face_list.append(im_path)

    # open_ui(faces,base_image) *


def move_images(des_dir, txt_file):
    if not os.path.isdir(des_dir):
        os.mkdir(des_dir)
    lines = open(txt_file, "r")
    for line in lines:
        print(line)
        shutil.move(line.strip(), des_dir)


```



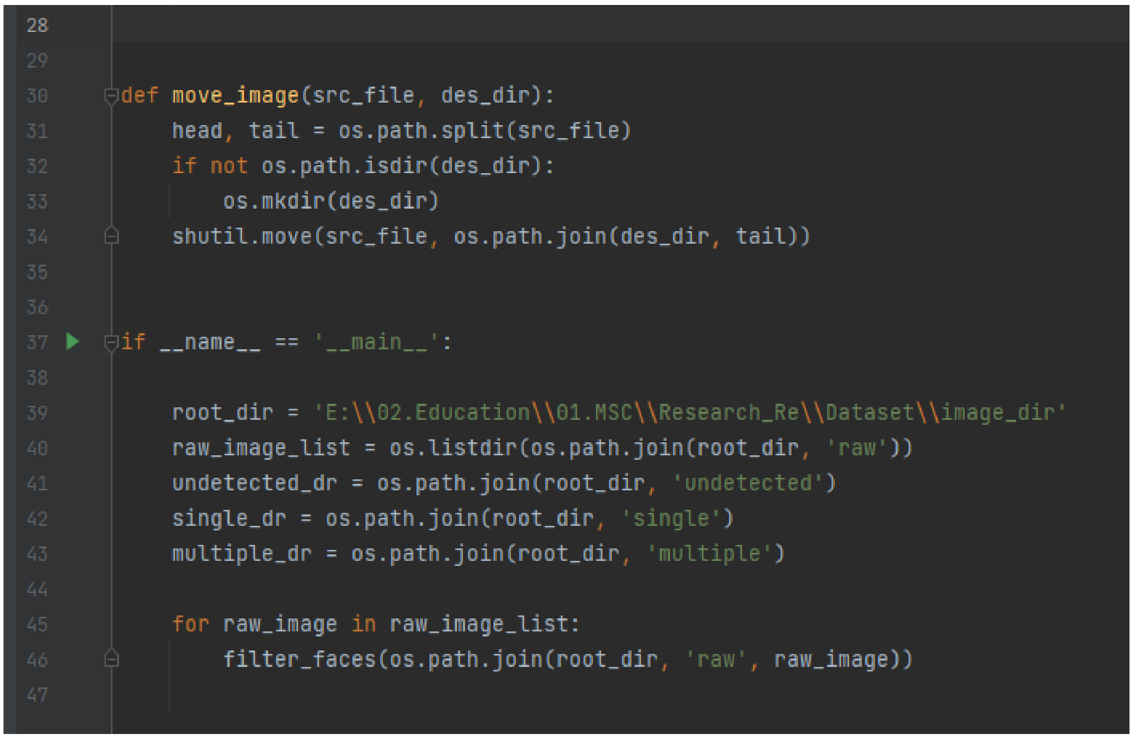
```python

if __name__ == '__main__':
    root_dir = 'E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\images_test_v2'
    filter_images(root_dir)

    undetected_face_list = os.path.join(root_dir, 'undetected_face_list.txt')
    undetected_dr = os.path.join(root_dir, 'undetected')
    move_images(undetected_dr, undetected_face_list)

    single_face_list = os.path.join(root_dir, 'single_face_list.txt')
    single_dr = os.path.join(root_dir, 'single')
    move_images(single_dr, single_face_list)

    multiple_face_list = os.path.join(root_dir, 'multiple_face_list.txt')
    multiple_dr = os.path.join(root_dir, 'multiple')
    move_images(multiple_dr, multiple_face_list)

```

MediaPipe Based Face Multiple Face Detection



filter.py

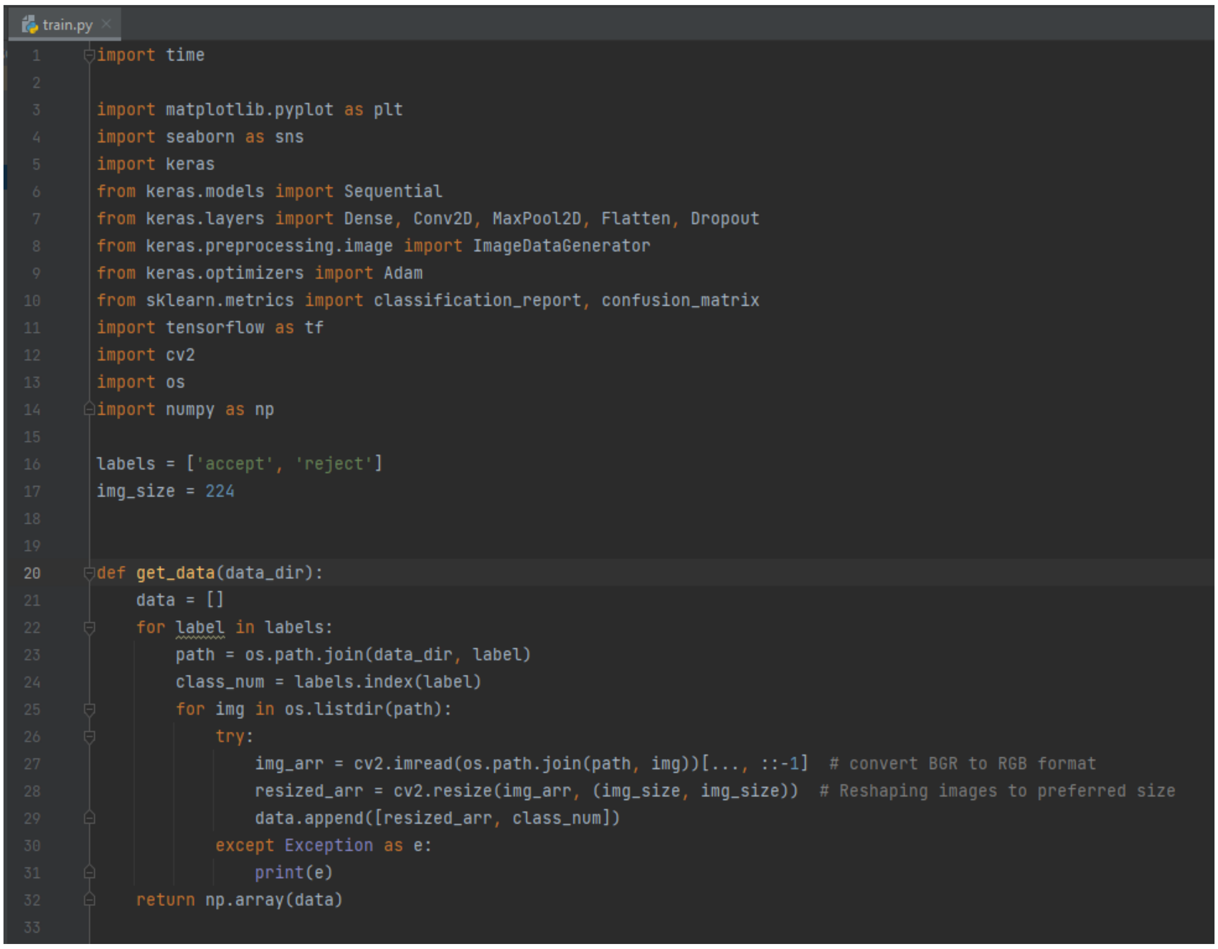
```python
import os
import shutil
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
# load face detection model
mp_face = mp.solutions.face_detection.FaceDetection(
    model_selection=1,  # model selection
    min_detection_confidence=0.5  # confidence threshold
)


def filter_faces(img_path):
    image_data = cv2.imread(img_path)
    image_input = cv2.cvtColor(image_data, cv2.COLOR_BGR2RGB)
    results = mp_face.process(image_input)

    if not results.detections:
        print('No faces detected.')
        move_image(img_path, undetected_dr)
    elif len(results.detections) == 1:
        print('Single Face only')
        move_image(img_path, single_dr)
    else:
        print('multiple Face Detected : ' + str(len(results.detections)))
        move_image(img_path, multiple_dr)

```



```python


def move_image(src_file, des_dir):
    head, tail = os.path.split(src_file)
    if not os.path.isdir(des_dir):
        os.mkdir(des_dir)
    shutil.move(src_file, os.path.join(des_dir, tail))


if __name__ == '__main__':

    root_dir = 'E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\image_dir'
    raw_image_list = os.listdir(os.path.join(root_dir, 'raw'))
    undetected_dr = os.path.join(root_dir, 'undetected')
    single_dr = os.path.join(root_dir, 'single')
    multiple_dr = os.path.join(root_dir, 'multiple')

    for raw_image in raw_image_list:
        filter_faces(os.path.join(root_dir, 'raw', raw_image))

```

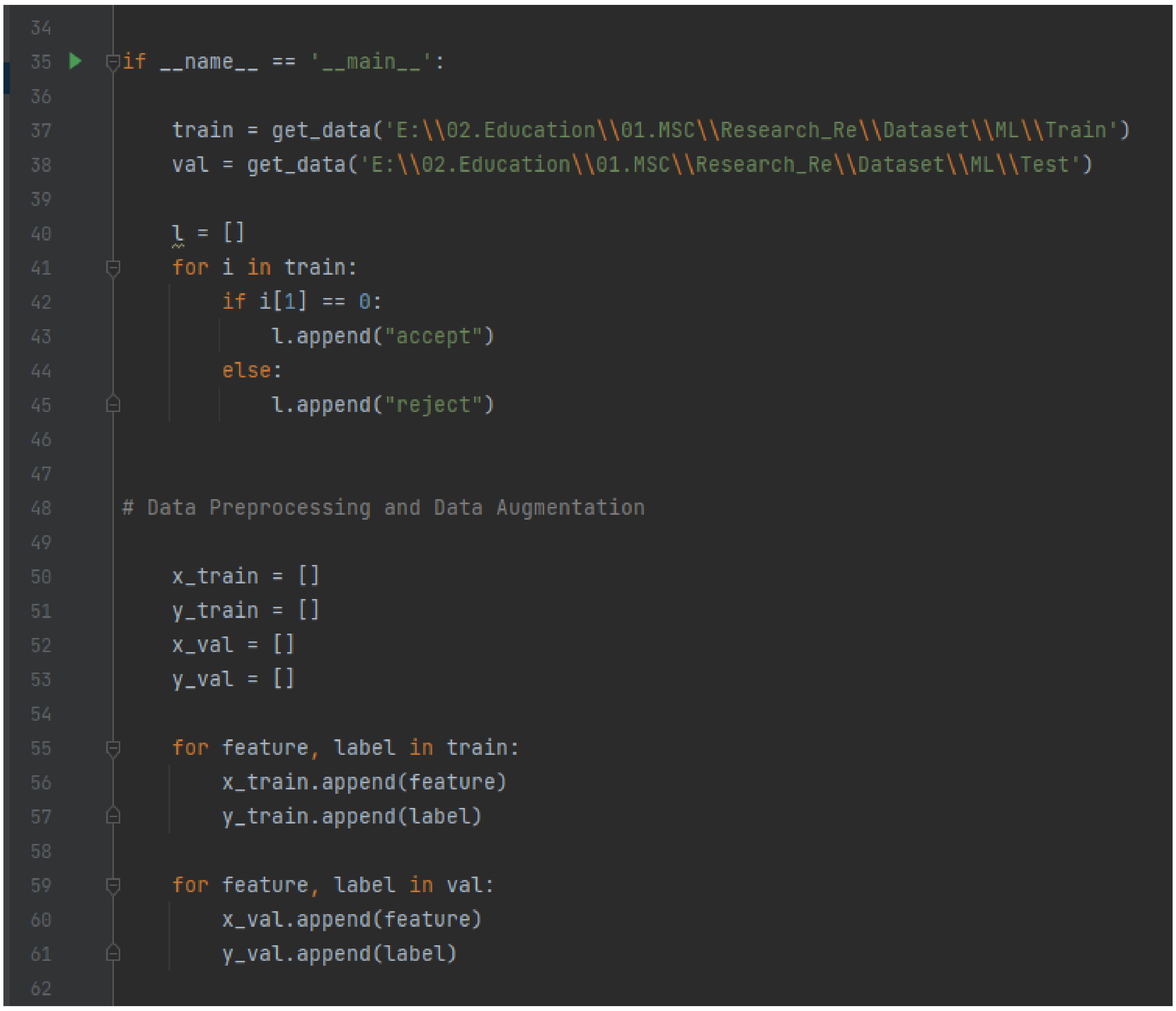
ML Image Classifier

Model training

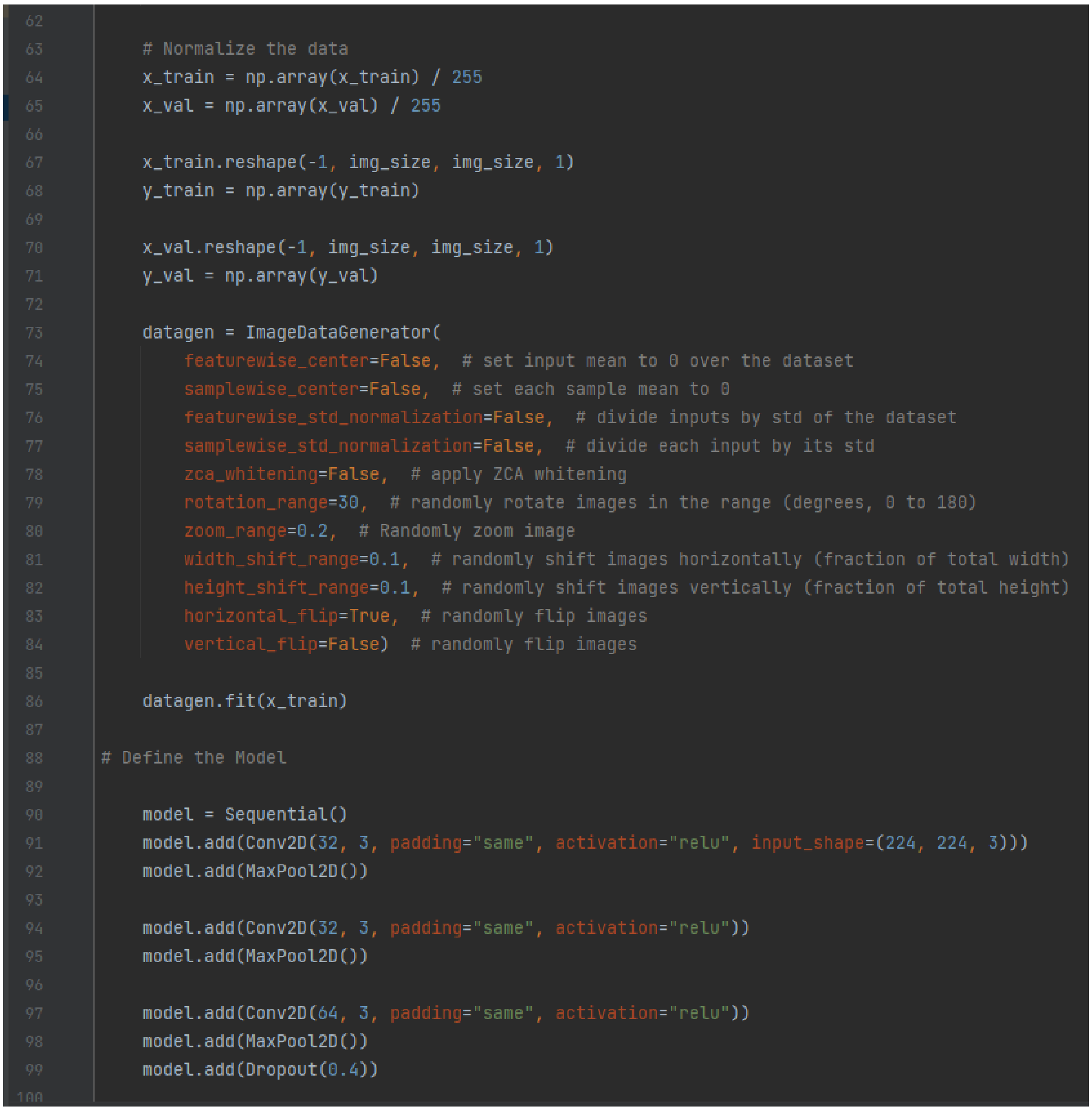


train.py

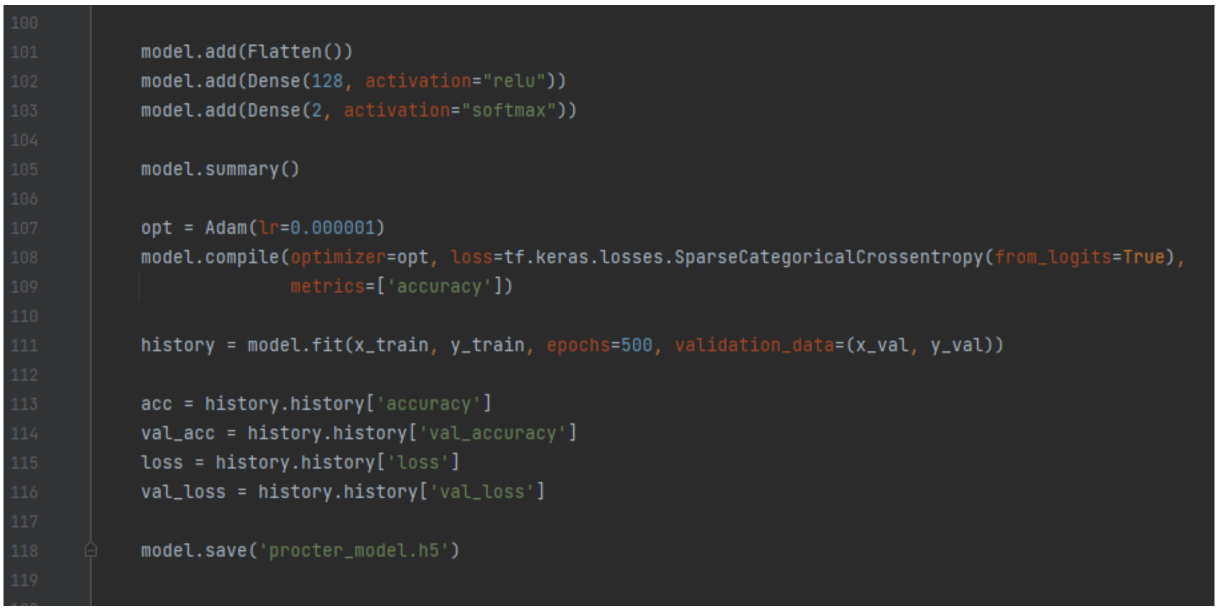
```python
import time

import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
import cv2
import os
import numpy as np

labels = ['accept', 'reject']
img_size = 224


def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img))[..., ::-1]  # convert BGR to RGB format
                resized_arr = cv2.resize(img_arr, (img_size, img_size))  # Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

```



```
if __name__ == '__main__':

    train = get_data('E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\ML\\Train')
    val = get_data('E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\ML\\Test')

    l = []
    for i in train:
        if i[1] == 0:
            l.append("accept")
        else:
            l.append("reject")


# Data Preprocessing and Data Augmentation

    x_train = []
    y_train = []
    x_val = []
    y_val = []

    for feature, label in train:
        x_train.append(feature)
        y_train.append(label)

    for feature, label in val:
        x_val.append(feature)
        y_val.append(label)

```

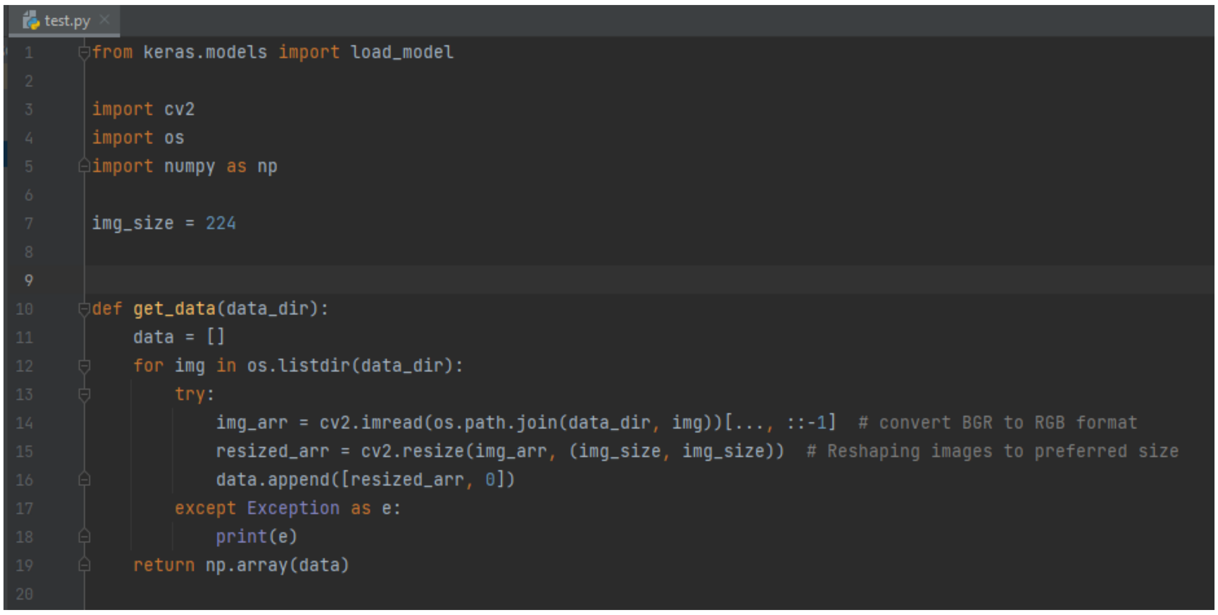


```python

    # Normalize the data
    x_train = np.array(x_train) / 255
    x_val = np.array(x_val) / 255

    x_train.reshape(-1, img_size, img_size, 1)
    y_train = np.array(y_train)

    x_val.reshape(-1, img_size, img_size, 1)
    y_val = np.array(y_val)

    datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=30,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range=0.2,  # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images

    datagen.fit(x_train)

# Define the Model

    model = Sequential()
    model.add(Conv2D(32, 3, padding="same", activation="relu", input_shape=(224, 224, 3)))
    model.add(MaxPool2D())

    model.add(Conv2D(32, 3, padding="same", activation="relu"))
    model.add(MaxPool2D())

    model.add(Conv2D(64, 3, padding="same", activation="relu"))
    model.add(MaxPool2D())
    model.add(Dropout(0.4))

```



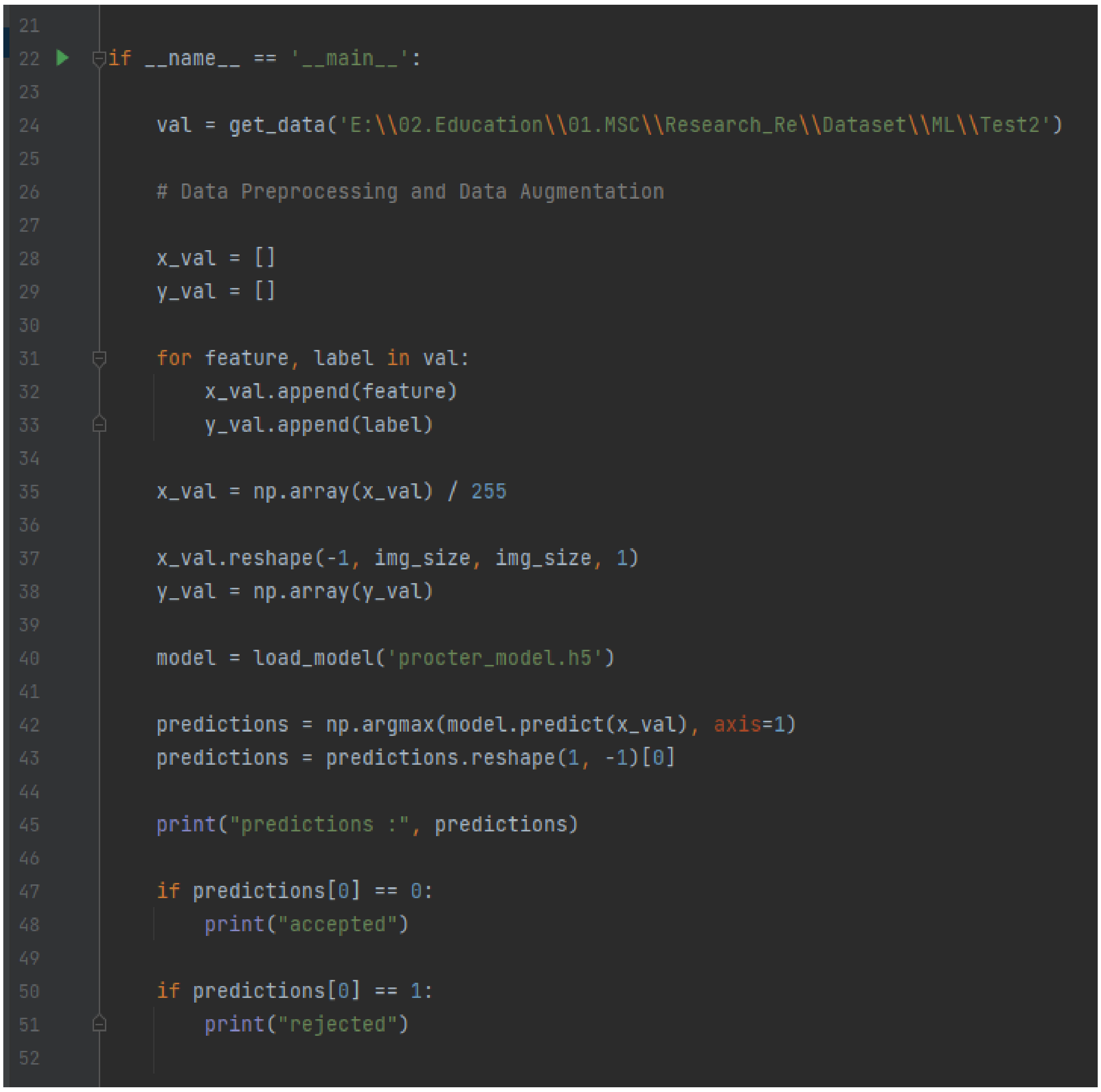
```python

    model.add(Flatten())
    model.add(Dense(128, activation="relu"))
    model.add(Dense(2, activation="softmax"))

    model.summary()

    opt = Adam(lr=0.000001)
    model.compile(optimizer=opt, loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    history = model.fit(x_train, y_train, epochs=500, validation_data=(x_val, y_val))

    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    model.save('procter_model.h5')

```

Model Testing



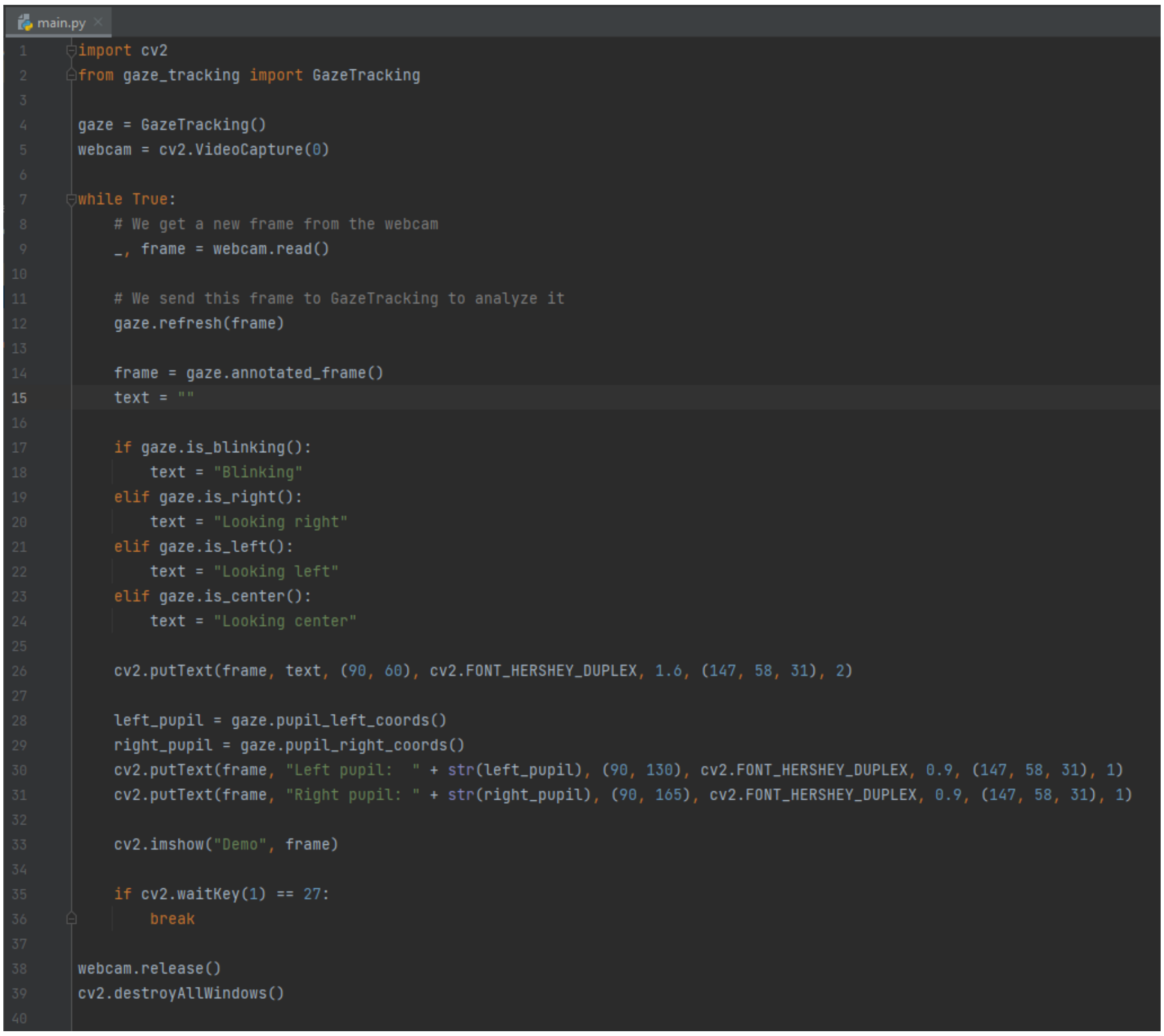
test.py

```python
from keras.models import load_model

import cv2
import os
import numpy as np

img_size = 224


def get_data(data_dir):
    data = []
    for img in os.listdir(data_dir):
        try:
            img_arr = cv2.imread(os.path.join(data_dir, img))[..., ::-1]  # convert BGR to RGB format
            resized_arr = cv2.resize(img_arr, (img_size, img_size))  # Reshaping images to preferred size
            data.append([resized_arr, 0])
        except Exception as e:
            print(e)
    return np.array(data)

```



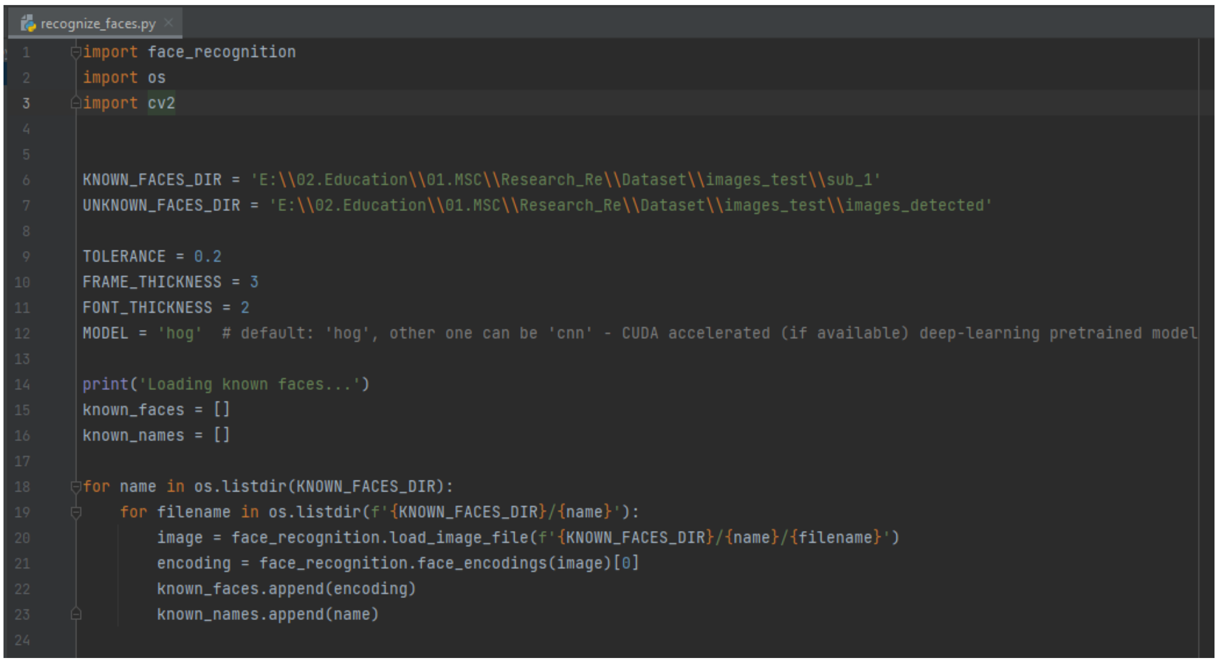
```
if __name__ == '__main__':

    val = get_data('E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\ML\\Test2')

    # Data Preprocessing and Data Augmentation

    x_val = []
    y_val = []

    for feature, label in val:
        x_val.append(feature)
        y_val.append(label)

    x_val = np.array(x_val) / 255

    x_val.reshape(-1, img_size, img_size, 1)
    y_val = np.array(y_val)

    model = load_model('procter_model.h5')

    predictions = np.argmax(model.predict(x_val), axis=1)
    predictions = predictions.reshape(1, -1)[0]

    print("predictions :", predictions)

    if predictions[0] == 0:
        print("accepted")

    if predictions[0] == 1:
        print("rejected")

```

Eye Gaze Detection

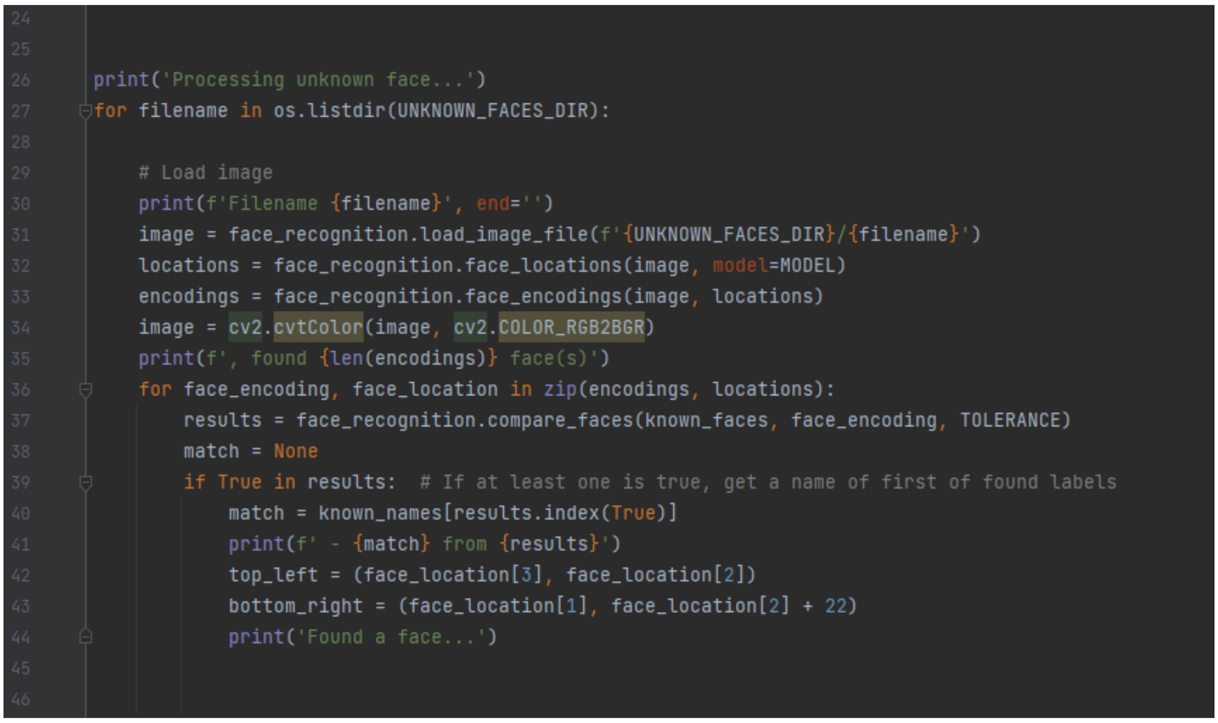

main.py

```python
import cv2
from gaze_tracking import GazeTracking

gaze = GazeTracking()
webcam = cv2.VideoCapture(0)

while True:
    # We get a new frame from the webcam
    _, frame = webcam.read()

    # We send this frame to GazeTracking to analyze it
    gaze.refresh(frame)

    frame = gaze.annotated_frame()
    text = ""

    if gaze.is_blinking():
        text = "Blinking"
    elif gaze.is_right():
        text = "Looking right"
    elif gaze.is_left():
        text = "Looking left"
    elif gaze.is_center():
        text = "Looking center"

    cv2.putText(frame, text, (90, 60), cv2.FONT_HERSHEY_DUPLEX, 1.6, (147, 58, 31), 2)

    left_pupil = gaze.pupil_left_coords()
    right_pupil = gaze.pupil_right_coords()
    cv2.putText(frame, "Left pupil:  " + str(left_pupil), (90, 130), cv2.FONT_HERSHEY_DUPLEX, 0.9, (147, 58, 31), 1)
    cv2.putText(frame, "Right pupil: " + str(right_pupil), (90, 165), cv2.FONT_HERSHEY_DUPLEX, 0.9, (147, 58, 31), 1)

    cv2.imshow("Demo", frame)

    if cv2.waitKey(1) == 27:
        break

webcam.release()
cv2.destroyAllWindows()
```

Identity Detection

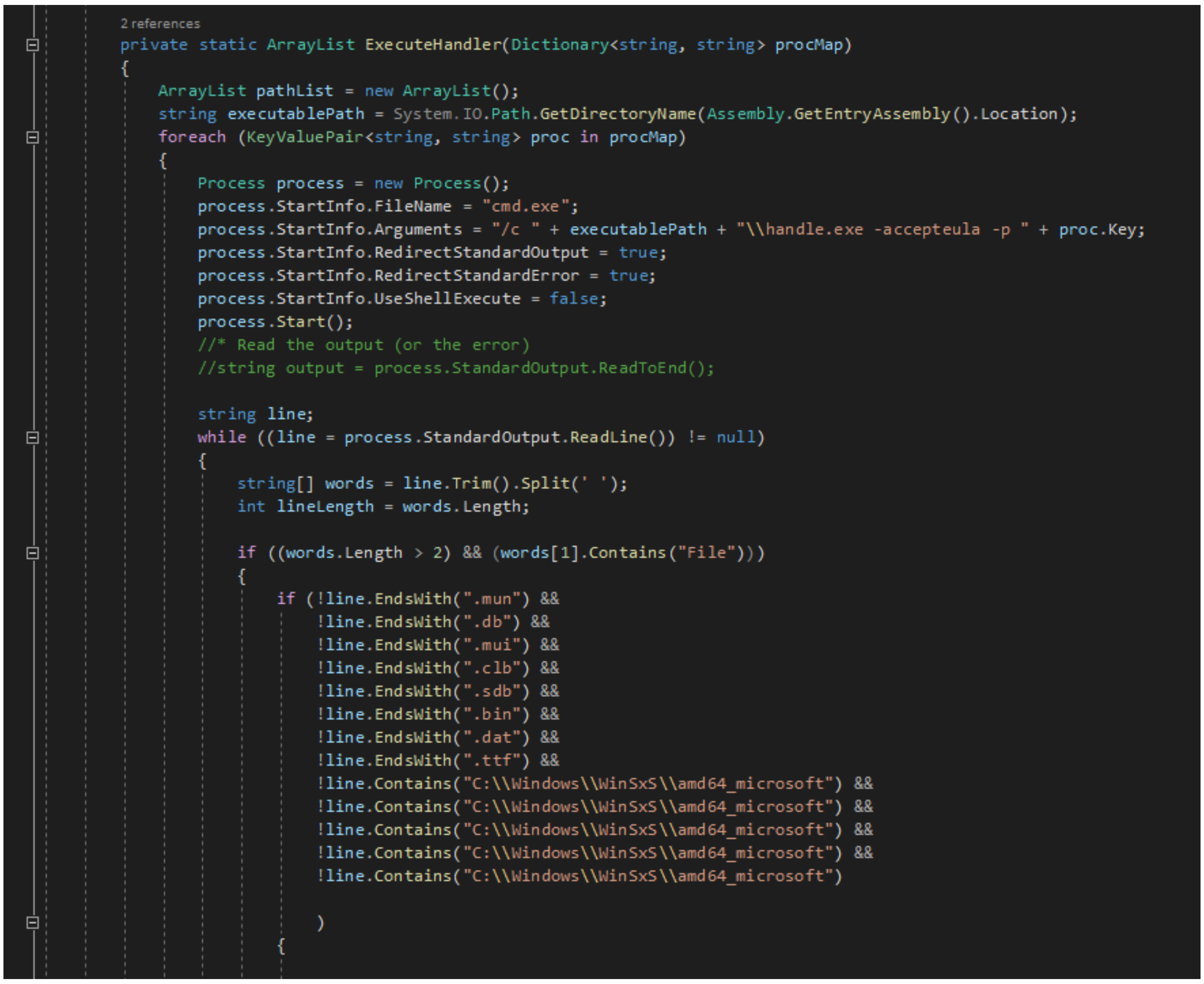


```python
import face_recognition
import os
import cv2


KNOWN_FACES_DIR = 'E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\images_test\\sub_1'
UNKNOWN_FACES_DIR = 'E:\\02.Education\\01.MSC\\Research_Re\\Dataset\\images_test\\images_detected'

TOLERANCE = 0.2
FRAME_THICKNESS = 3
FONT_THICKNESS = 2
MODEL = 'hog'  # default: 'hog', other one can be 'cnn' - CUDA accelerated (if available) deep-learning pretrained model

print('Loading known faces...')
known_faces = []
known_names = []

for name in os.listdir(KNOWN_FACES_DIR):
    for filename in os.listdir(f'{KNOWN_FACES_DIR}/{name}'):
        image = face_recognition.load_image_file(f'{KNOWN_FACES_DIR}/{name}/{filename}')
        encoding = face_recognition.face_encodings(image)[0]
        known_faces.append(encoding)
        known_names.append(name)

```

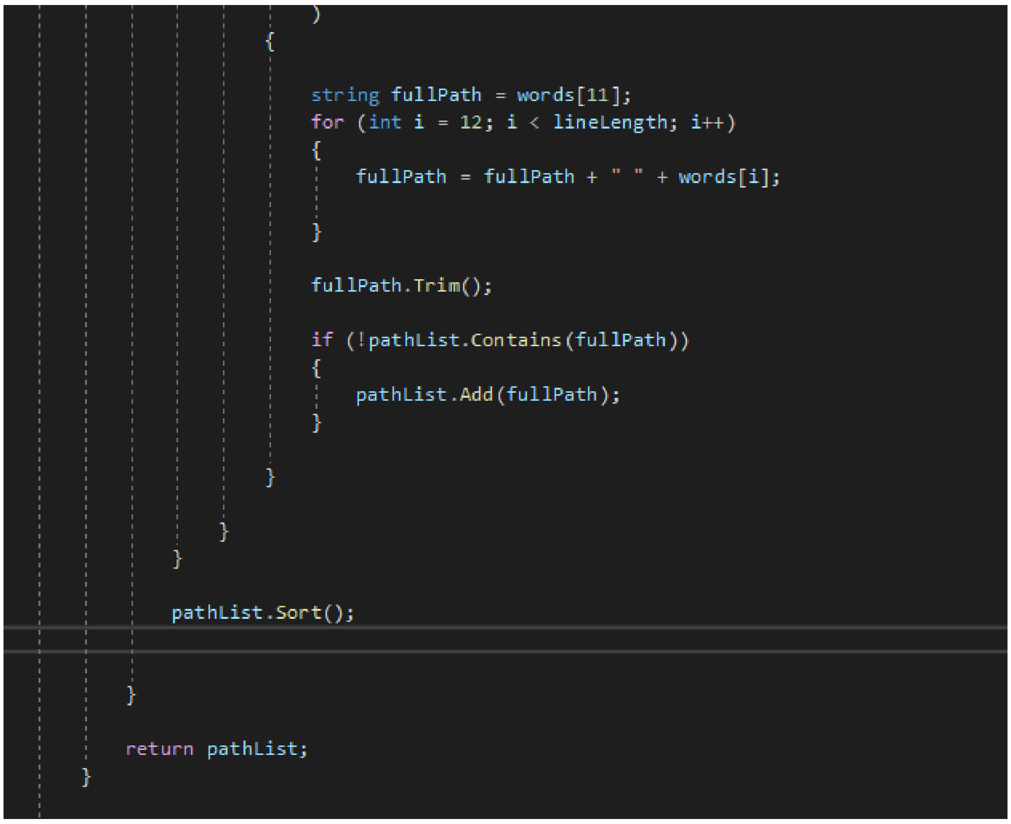


```python


print('Processing unknown face...')
for filename in os.listdir(UNKNOWN_FACES_DIR):

    # Load image
    print(f'Filename {filename}', end='')
    image = face_recognition.load_image_file(f'{UNKNOWN_FACES_DIR}/{filename}')
    locations = face_recognition.face_locations(image, model=MODEL)
    encodings = face_recognition.face_encodings(image, locations)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    print(f', found {len(encodings)} face(s)')
    for face_encoding, face_location in zip(encodings, locations):
        results = face_recognition.compare_faces(known_faces, face_encoding, TOLERANCE)
        match = None
        if True in results:  # If at least one is true, get a name of first of found labels
            match = known_names[results.index(True)]
            print(f' - {match} from {results}')
            top_left = (face_location[3], face_location[2])
            bottom_right = (face_location[1], face_location[2] + 22)
            print('Found a face...')


```

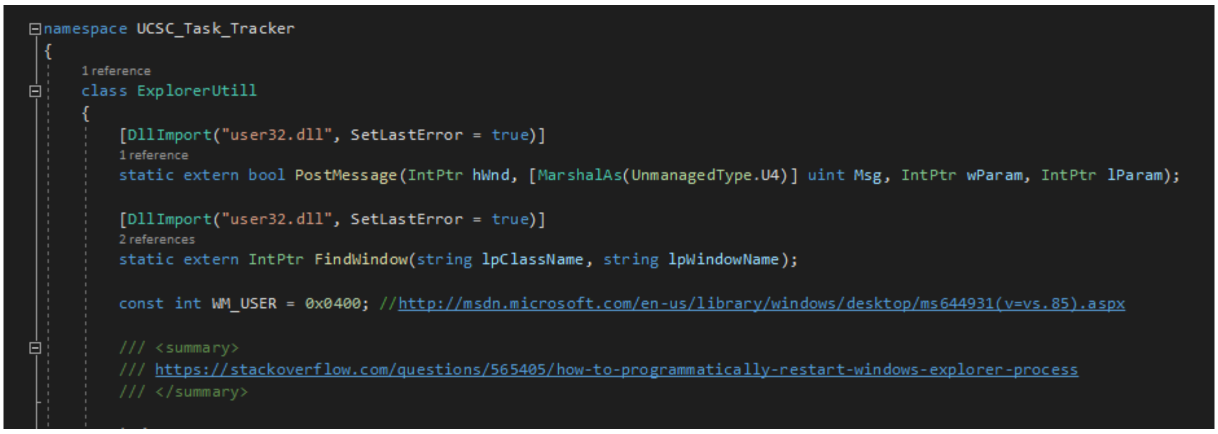
Activity Detection



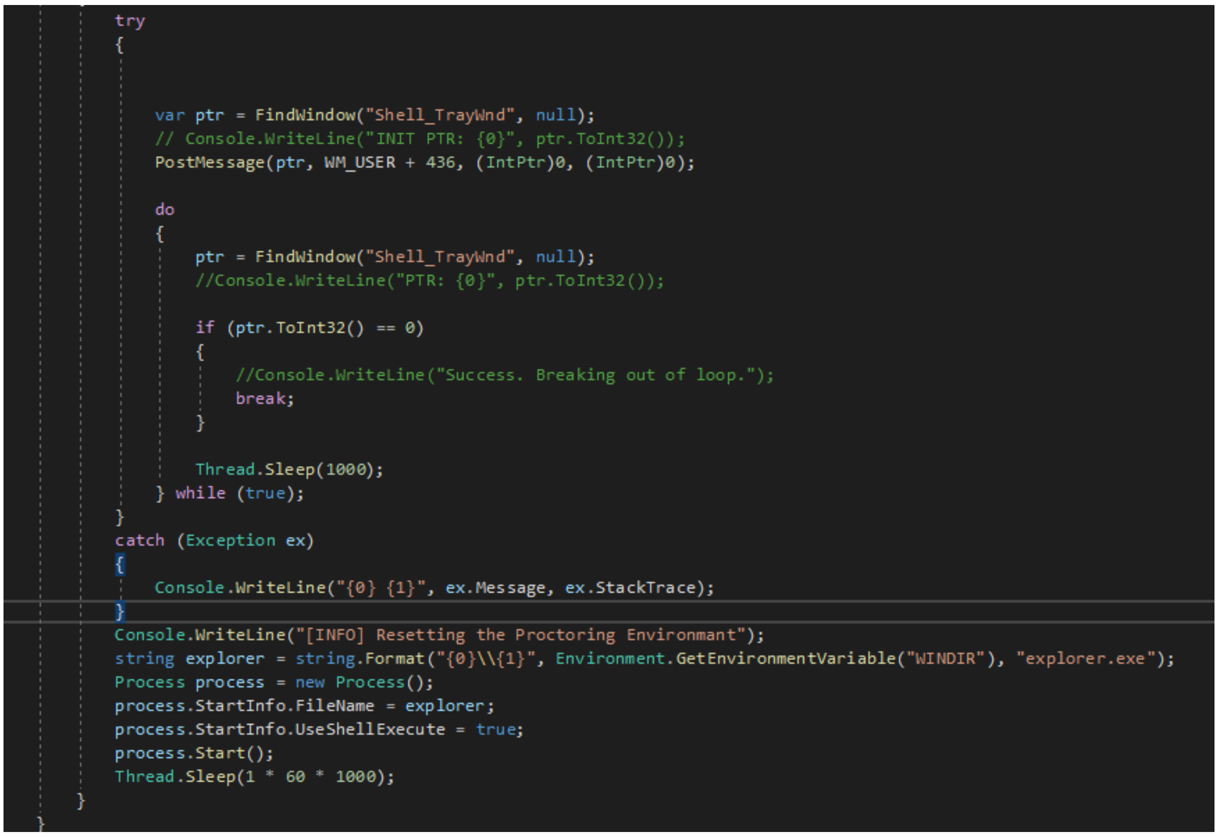
```
2 references
private static ArrayList ExecuteHandler(Dictionary<string, string> procMap)
{
    ArrayList pathList = new ArrayList();
    string executablePath = System.IO.Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);
    foreach (KeyValuePair<string, string> proc in procMap)
    {
        Process process = new Process();
        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.Arguments = "/c " + executablePath + "\\handle.exe -accepteula -p " + proc.Key;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.RedirectStandardError = true;
        process.StartInfo.UseShellExecute = false;
        process.Start();
        //* Read the output (or the error)
        //string output = process.StandardOutput.ReadToEnd();

        string line;
        while ((line = process.StandardOutput.ReadLine()) != null)
        {
            string[] words = line.Trim().Split(' ');
            int lineLength = words.Length;

            if ((words.Length > 2) && (words[1].Contains("File")))
            {
                if (!line.EndsWith(".mun") &&
                    !line.EndsWith(".db") &&
                    !line.EndsWith(".mui") &&
                    !line.EndsWith(".clb") &&
                    !line.EndsWith(".sdb") &&
                    !line.EndsWith(".bin") &&
                    !line.EndsWith(".dat") &&
                    !line.EndsWith(".ttf") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft")

                    )
                {
```



```
                    )
                    {

                        string fullPath = words[11];
                        for (int i = 12; i < lineLength; i++)
                        {
                            fullPath = fullPath + " " + words[i];

                        }

                        fullPath.Trim();

                        if (!pathList.Contains(fullPath))
                        {
                            pathList.Add(fullPath);
                        }

                    }

                }
            }

            pathList.Sort();


        }

        return pathList;
    }
```

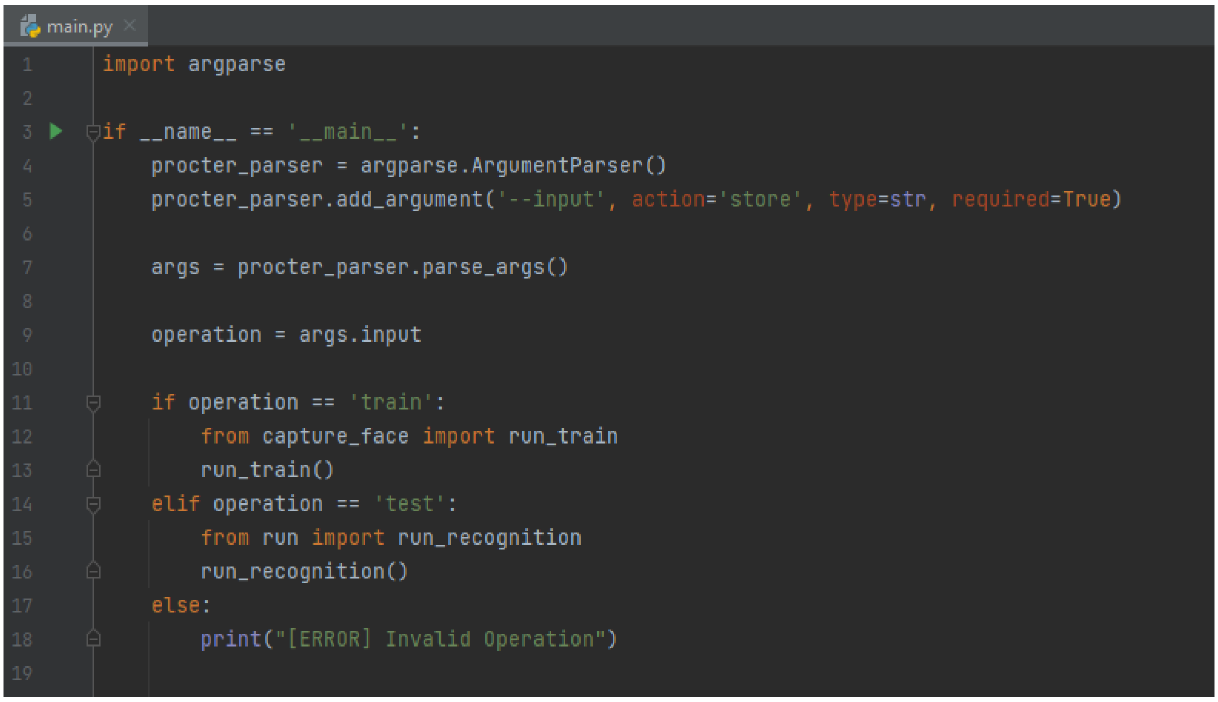


```
namespace UCSC_Task_Tracker
{
    1 reference
    class ExplorerUtill
    {
        [DllImport("user32.dll", SetLastError = true)]
        1 reference
        static extern bool PostMessage(IntPtr hWnd, [MarshalAs(UnmanagedType.U4)] uint Msg, IntPtr wParam, IntPtr lParam);

        [DllImport("user32.dll", SetLastError = true)]
        2 references
        static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

        const int WM_USER = 0x0400; //http://msdn.microsoft.com/en-us/library/windows/desktop/ms644931(v=vs.85).aspx

        /// <summary>
        /// https://stackoverflow.com/questions/565405/how-to-programmatically-restart-windows-explorer-process
        /// </summary>
```

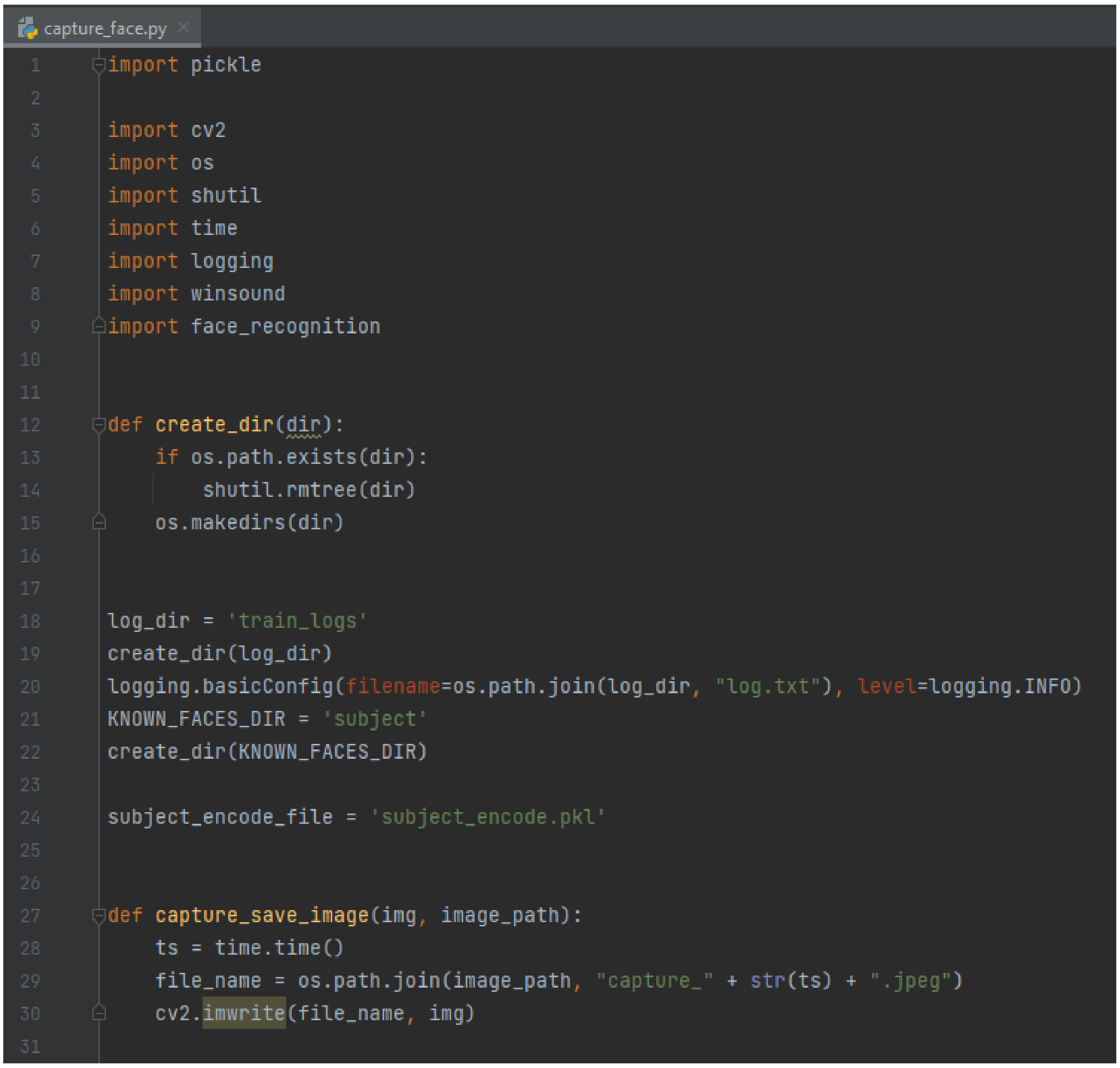


```
        try
        {


            var ptr = FindWindow("Shell_TrayWnd", null);
            // Console.WriteLine("INIT PTR: {0}", ptr.ToInt32());
            PostMessage(ptr, WM_USER + 436, (IntPtr)0, (IntPtr)0);

            do
            {
                ptr = FindWindow("Shell_TrayWnd", null);
                //Console.WriteLine("PTR: {0}", ptr.ToInt32());

                if (ptr.ToInt32() == 0)
                {
                    //Console.WriteLine("Success. Breaking out of loop.");
                    break;
                }

                Thread.Sleep(1000);
            } while (true);
        }
        catch (Exception ex)
        {
            Console.WriteLine("{0} {1}", ex.Message, ex.StackTrace);
        }
        Console.WriteLine("[INFO] Resetting the Proctoring Environmant");
        string explorer = string.Format("{0}\\{1}", Environment.GetEnvironmentVariable("WINDIR"), "explorer.exe");
        Process process = new Process();
        process.StartInfo.FileName = explorer;
        process.StartInfo.UseShellExecute = true;
        process.Start();
        Thread.Sleep(1 * 60 * 1000);
    }
}
```

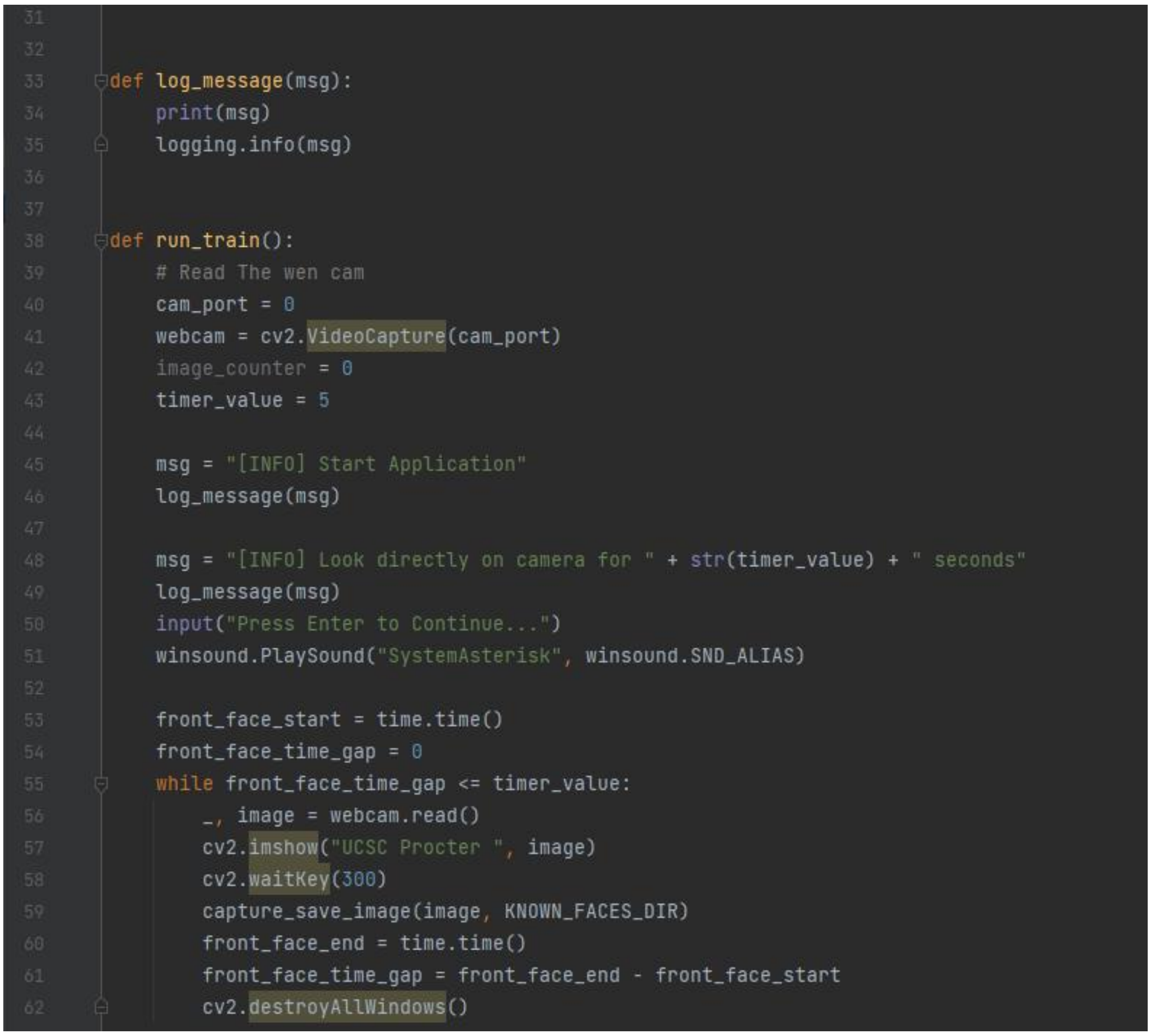
Python-based image analysis module


main.py

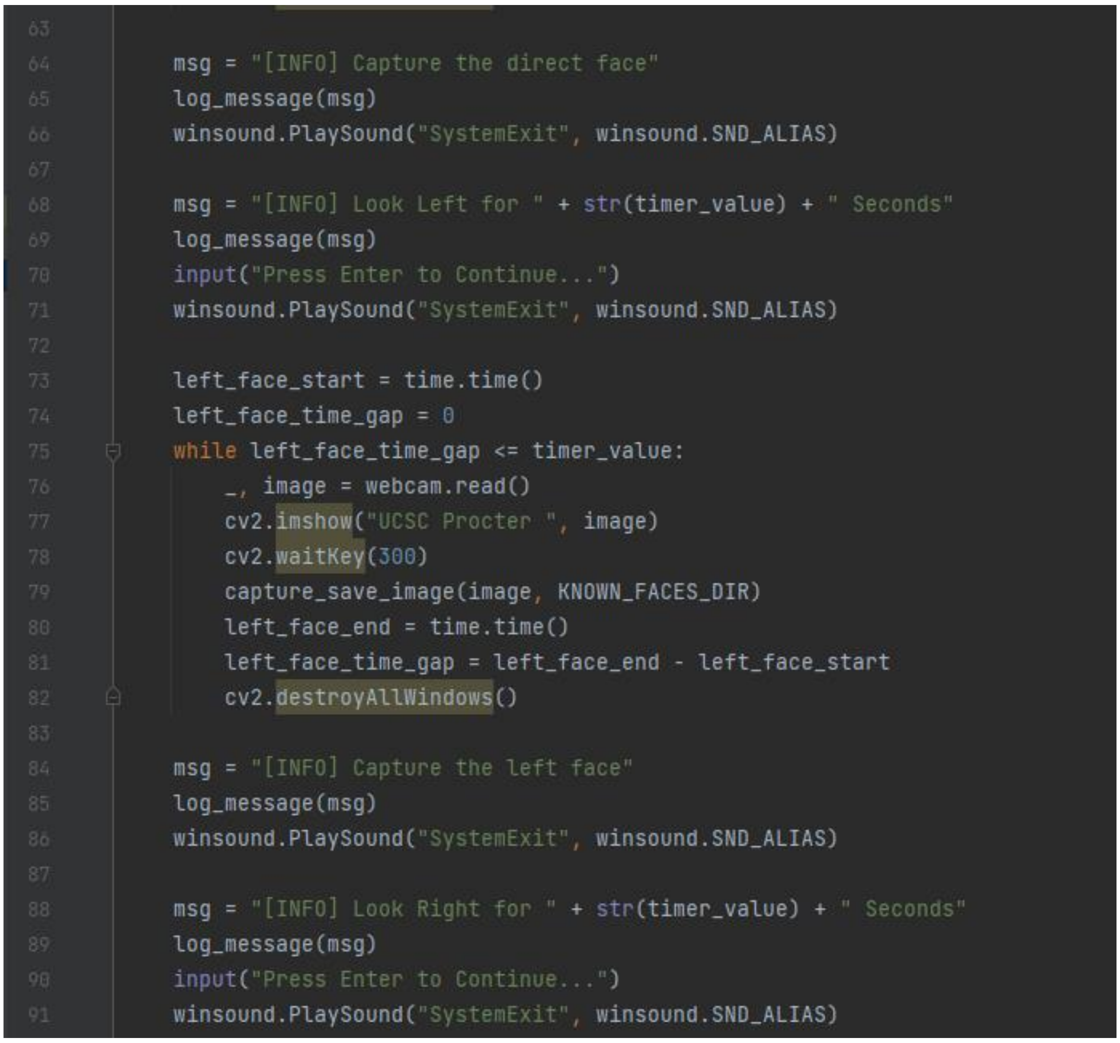
```python
import argparse

if __name__ == '__main__':
    procter_parser = argparse.ArgumentParser()
    procter_parser.add_argument('--input', action='store', type=str, required=True)

    args = procter_parser.parse_args()

    operation = args.input

    if operation == 'train':
        from capture_face import run_train
        run_train()
    elif operation == 'test':
        from run import run_recognition
        run_recognition()
    else:
        print("[ERROR] Invalid Operation")

```



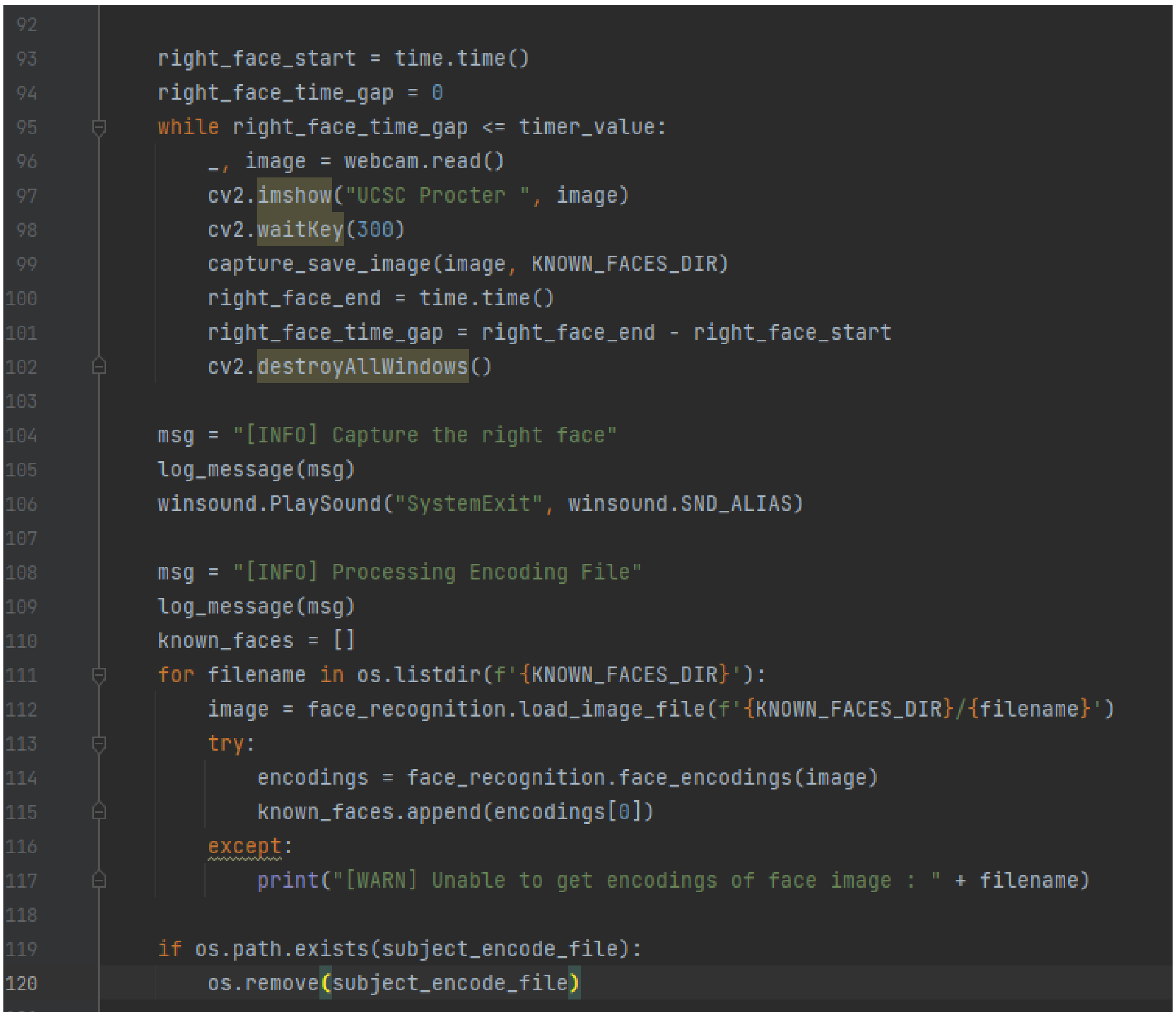
```python
import pickle

import cv2
import os
import shutil
import time
import logging
import winsound
import face_recognition


def create_dir(dir):
    if os.path.exists(dir):
        shutil.rmtree(dir)
    os.makedirs(dir)


log_dir = 'train_logs'
create_dir(log_dir)
logging.basicConfig(filename=os.path.join(log_dir, "log.txt"), level=logging.INFO)
KNOWN_FACES_DIR = 'subject'
create_dir(KNOWN_FACES_DIR)

subject_encode_file = 'subject_encode.pkl'


def capture_save_image(img, image_path):
    ts = time.time()
    file_name = os.path.join(image_path, "capture_" + str(ts) + ".jpeg")
    cv2.imwrite(file_name, img)

```



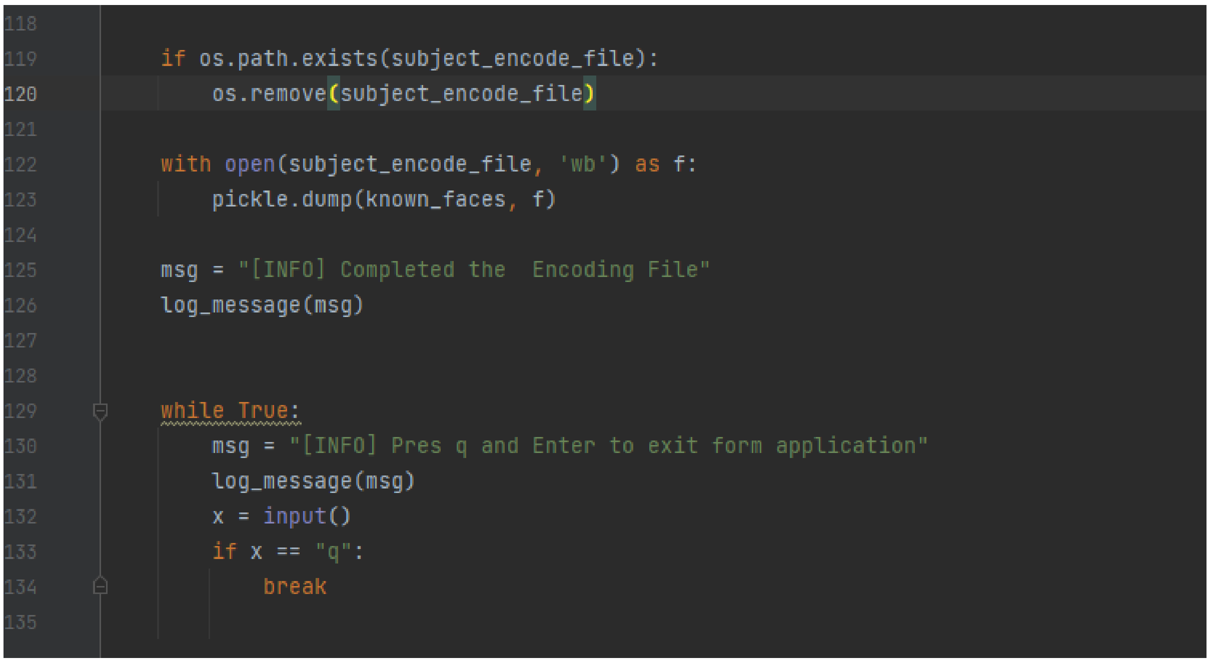
```python


def log_message(msg):
    print(msg)
    logging.info(msg)


def run_train():
    # Read The wen cam
    cam_port = 0
    webcam = cv2.VideoCapture(cam_port)
    image_counter = 0
    timer_value = 5

    msg = "[INFO] Start Application"
    log_message(msg)

    msg = "[INFO] Look directly on camera for " + str(timer_value) + " seconds"
    log_message(msg)
    input("Press Enter to Continue...")
    winsound.PlaySound("SystemAsterisk", winsound.SND_ALIAS)

    front_face_start = time.time()
    front_face_time_gap = 0
    while front_face_time_gap <= timer_value:
        _, image = webcam.read()
        cv2.imshow("UCSC Procter ", image)
        cv2.waitKey(300)
        capture_save_image(image, KNOWN_FACES_DIR)
        front_face_end = time.time()
        front_face_time_gap = front_face_end - front_face_start
        cv2.destroyAllWindows()
```



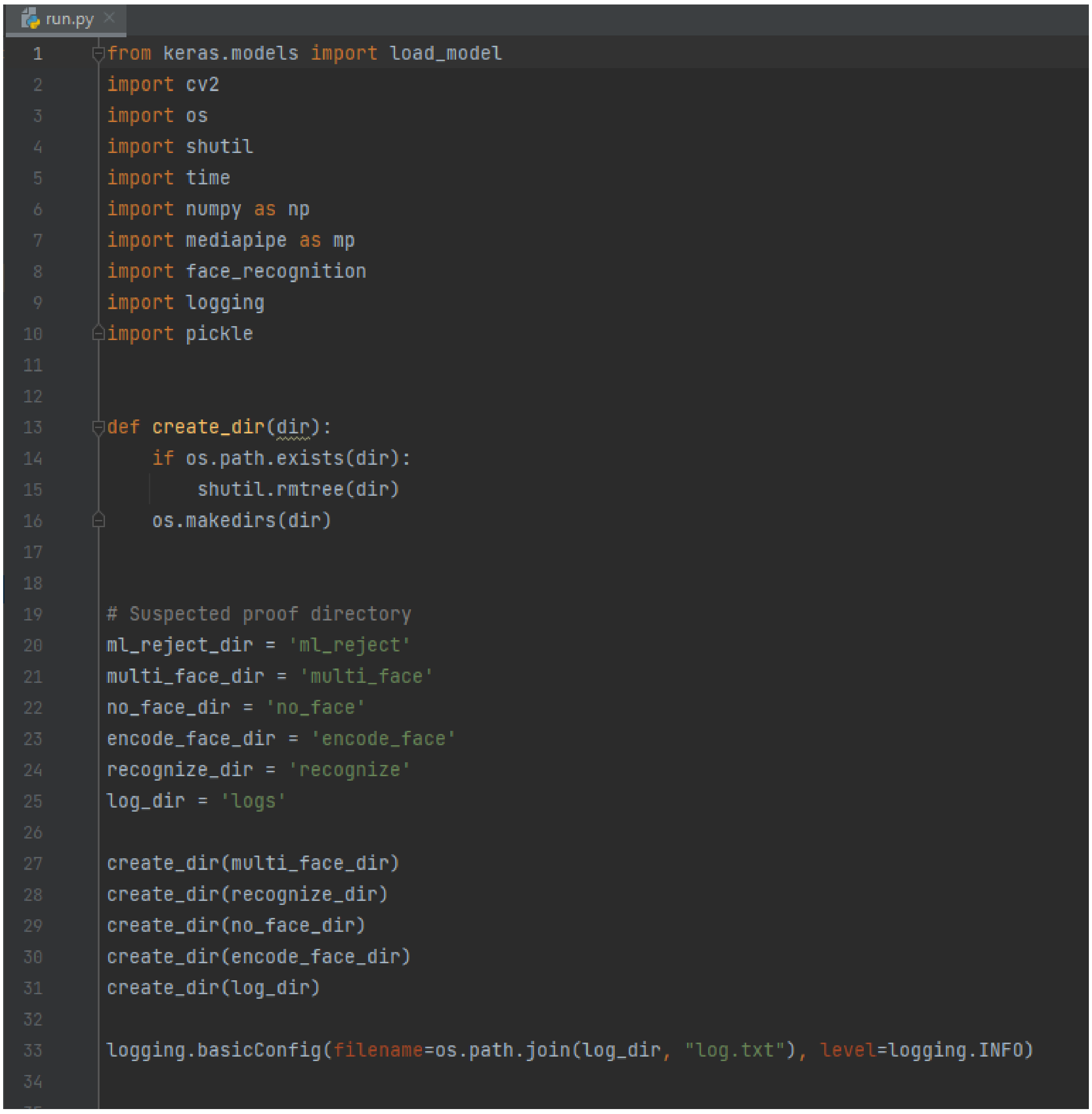
```python

    msg = "[INFO] Capture the direct face"
    log_message(msg)
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

    msg = "[INFO] Look Left for " + str(timer_value) + " Seconds"
    log_message(msg)
    input("Press Enter to Continue...")
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

    left_face_start = time.time()
    left_face_time_gap = 0
    while left_face_time_gap <= timer_value:
        _, image = webcam.read()
        cv2.imshow("UCSC Procter ", image)
        cv2.waitKey(300)
        capture_save_image(image, KNOWN_FACES_DIR)
        left_face_end = time.time()
        left_face_time_gap = left_face_end - left_face_start
        cv2.destroyAllWindows()

    msg = "[INFO] Capture the left face"
    log_message(msg)
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

    msg = "[INFO] Look Right for " + str(timer_value) + " Seconds"
    log_message(msg)
    input("Press Enter to Continue...")
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
```



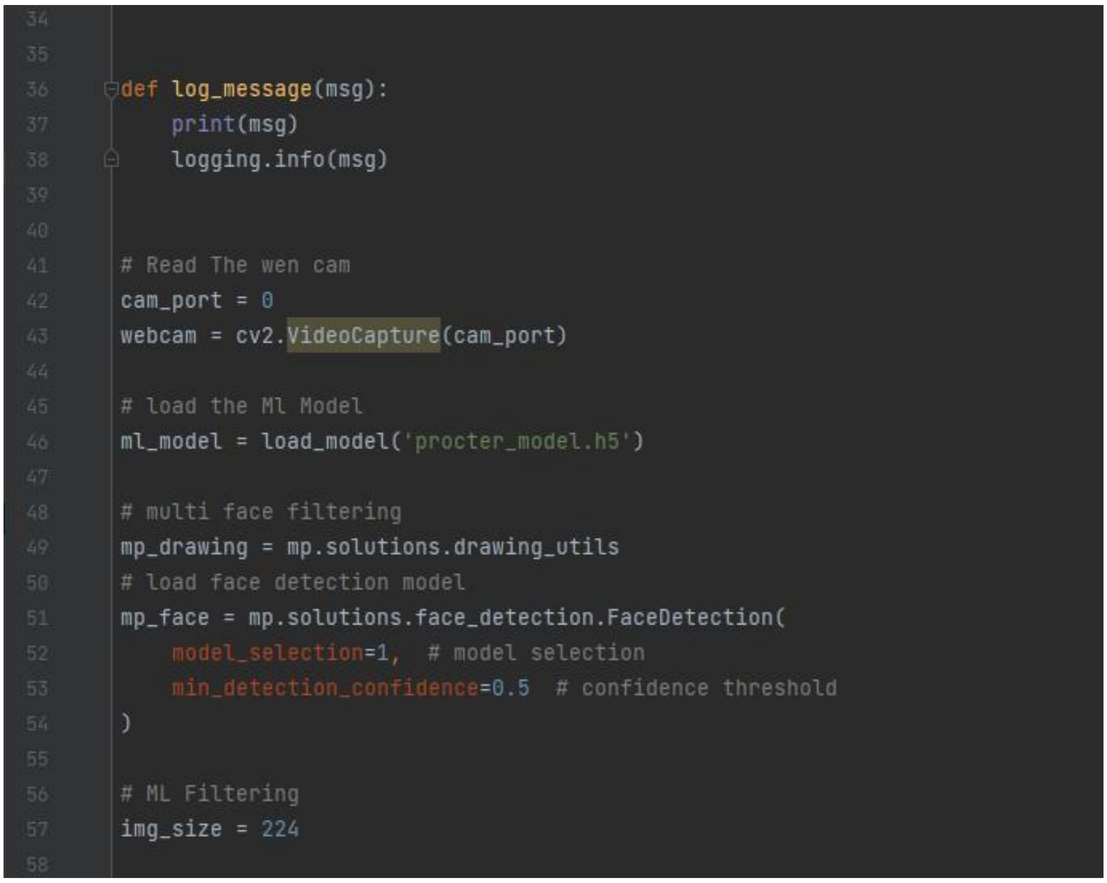
```python

    right_face_start = time.time()
    right_face_time_gap = 0
    while right_face_time_gap <= timer_value:
        _, image = webcam.read()
        cv2.imshow("UCSC Procter ", image)
        cv2.waitKey(300)
        capture_save_image(image, KNOWN_FACES_DIR)
        right_face_end = time.time()
        right_face_time_gap = right_face_end - right_face_start
        cv2.destroyAllWindows()

    msg = "[INFO] Capture the right face"
    log_message(msg)
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

    msg = "[INFO] Processing Encoding File"
    log_message(msg)
    known_faces = []
    for filename in os.listdir(f'{KNOWN_FACES_DIR}'):
        image = face_recognition.load_image_file(f'{KNOWN_FACES_DIR}/{filename}')
        try:
            encodings = face_recognition.face_encodings(image)
            known_faces.append(encodings[0])
        except:
            print("[WARN] Unable to get encodings of face image : " + filename)

    if os.path.exists(subject_encode_file):
        os.remove(subject_encode_file)
```



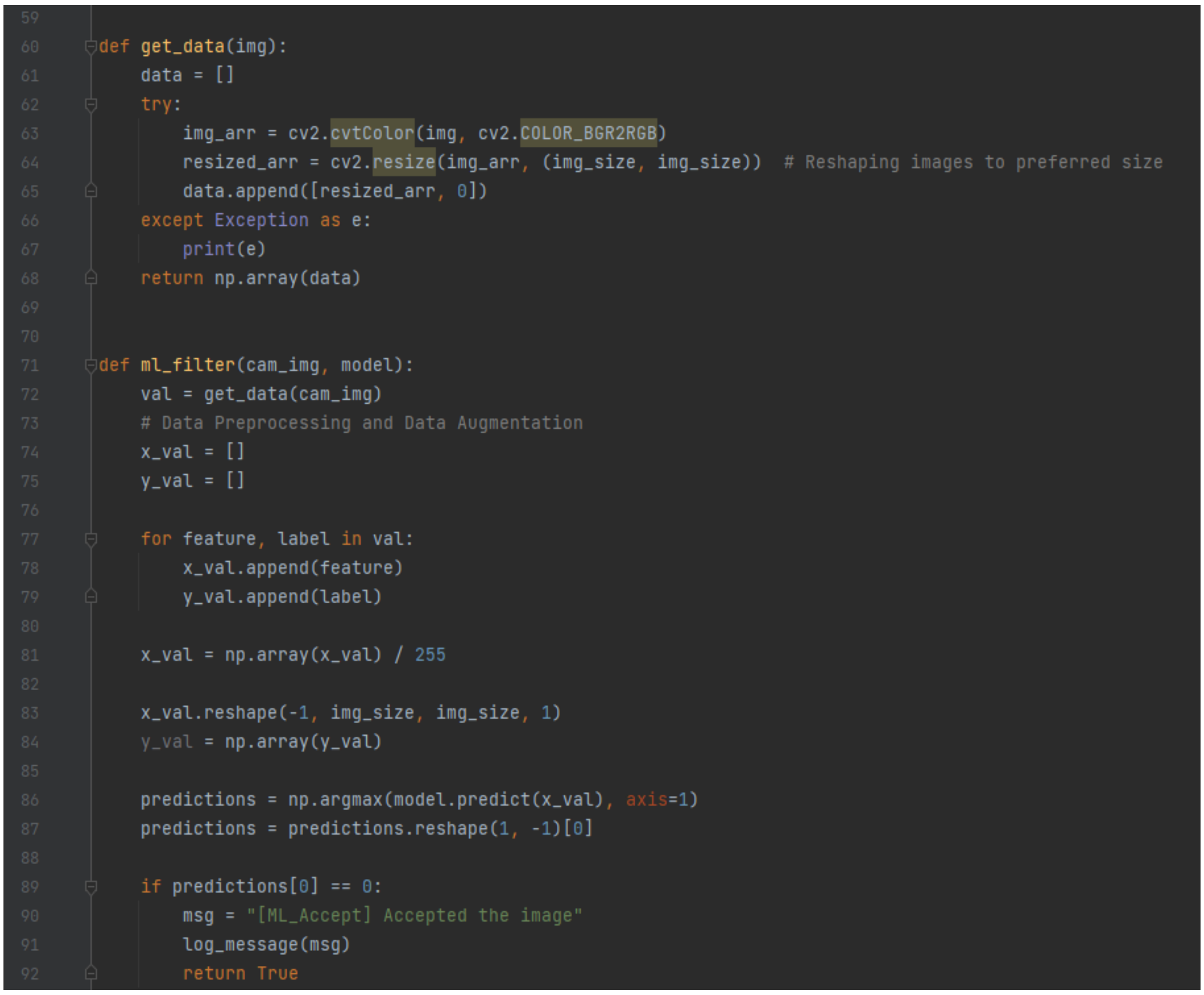
```python

    if os.path.exists(subject_encode_file):
        os.remove(subject_encode_file)

    with open(subject_encode_file, 'wb') as f:
        pickle.dump(known_faces, f)

    msg = "[INFO] Completed the  Encoding File"
    log_message(msg)


    while True:
        msg = "[INFO] Pres q and Enter to exit form application"
        log_message(msg)
        x = input()
        if x == "q":
            break

```



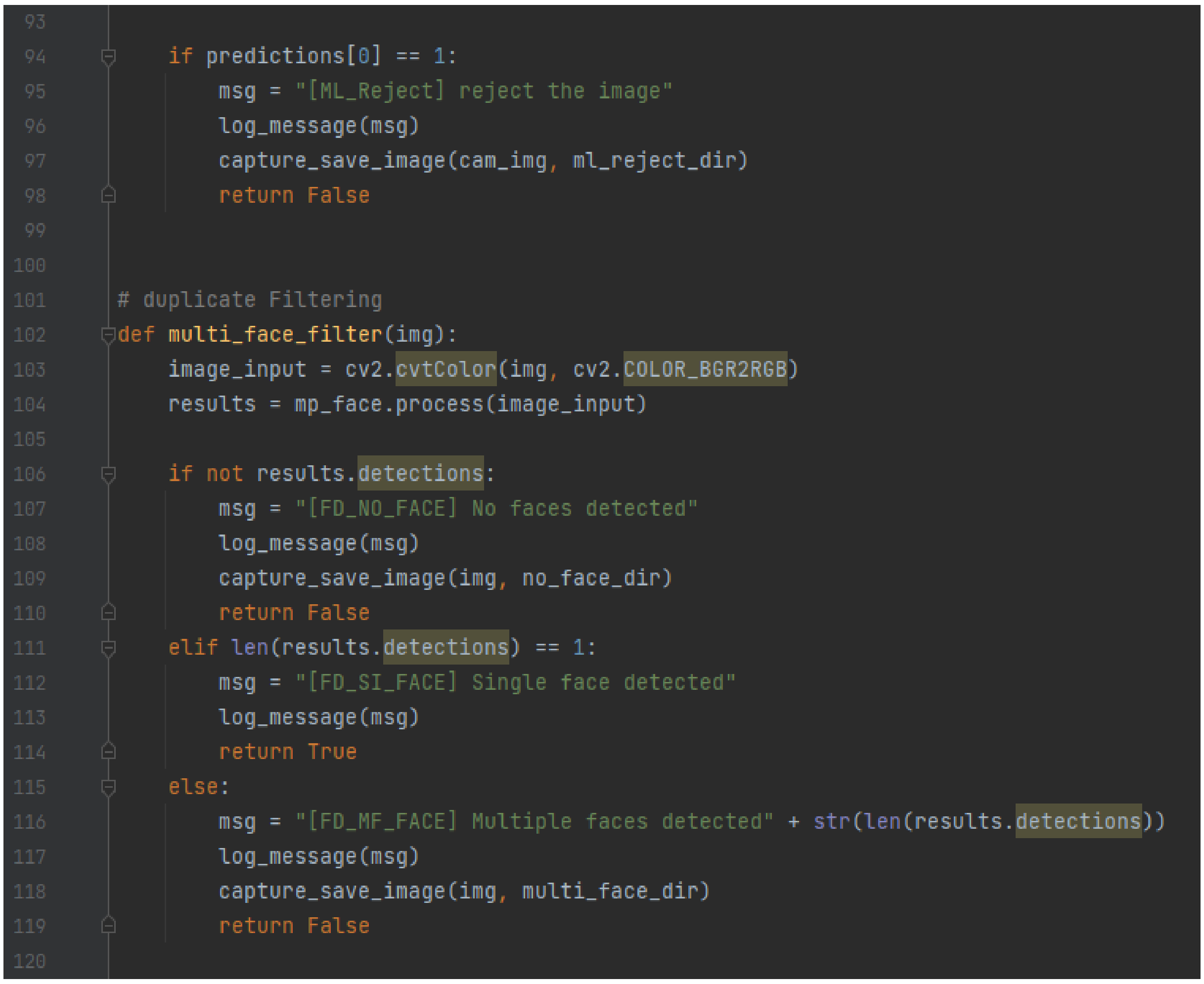
```python
from keras.models import load_model
import cv2
import os
import shutil
import time
import numpy as np
import mediapipe as mp
import face_recognition
import logging
import pickle


def create_dir(dir):
    if os.path.exists(dir):
        shutil.rmtree(dir)
    os.makedirs(dir)


# Suspected proof directory
ml_reject_dir = 'ml_reject'
multi_face_dir = 'multi_face'
no_face_dir = 'no_face'
encode_face_dir = 'encode_face'
recognize_dir = 'recognize'
log_dir = 'logs'

create_dir(multi_face_dir)
create_dir(recognize_dir)
create_dir(no_face_dir)
create_dir(encode_face_dir)
create_dir(log_dir)

logging.basicConfig(filename=os.path.join(log_dir, "log.txt"), level=logging.INFO)

```



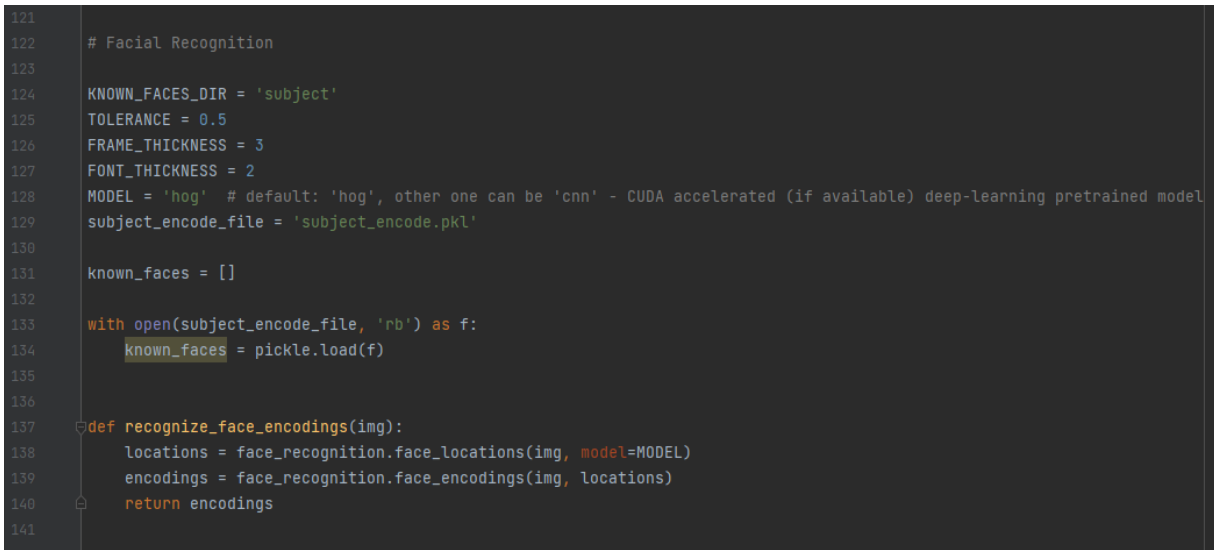
```


def log_message(msg):
    print(msg)
    logging.info(msg)


# Read The wen cam
cam_port = 0
webcam = cv2.VideoCapture(cam_port)

# load the Ml Model
ml_model = load_model('procter_model.h5')

# multi face filtering
mp_drawing = mp.solutions.drawing_utils
# load face detection model
mp_face = mp.solutions.face_detection.FaceDetection(
    model_selection=1,  # model selection
    min_detection_confidence=0.5  # confidence threshold
)

# ML Filtering
img_size = 224

```



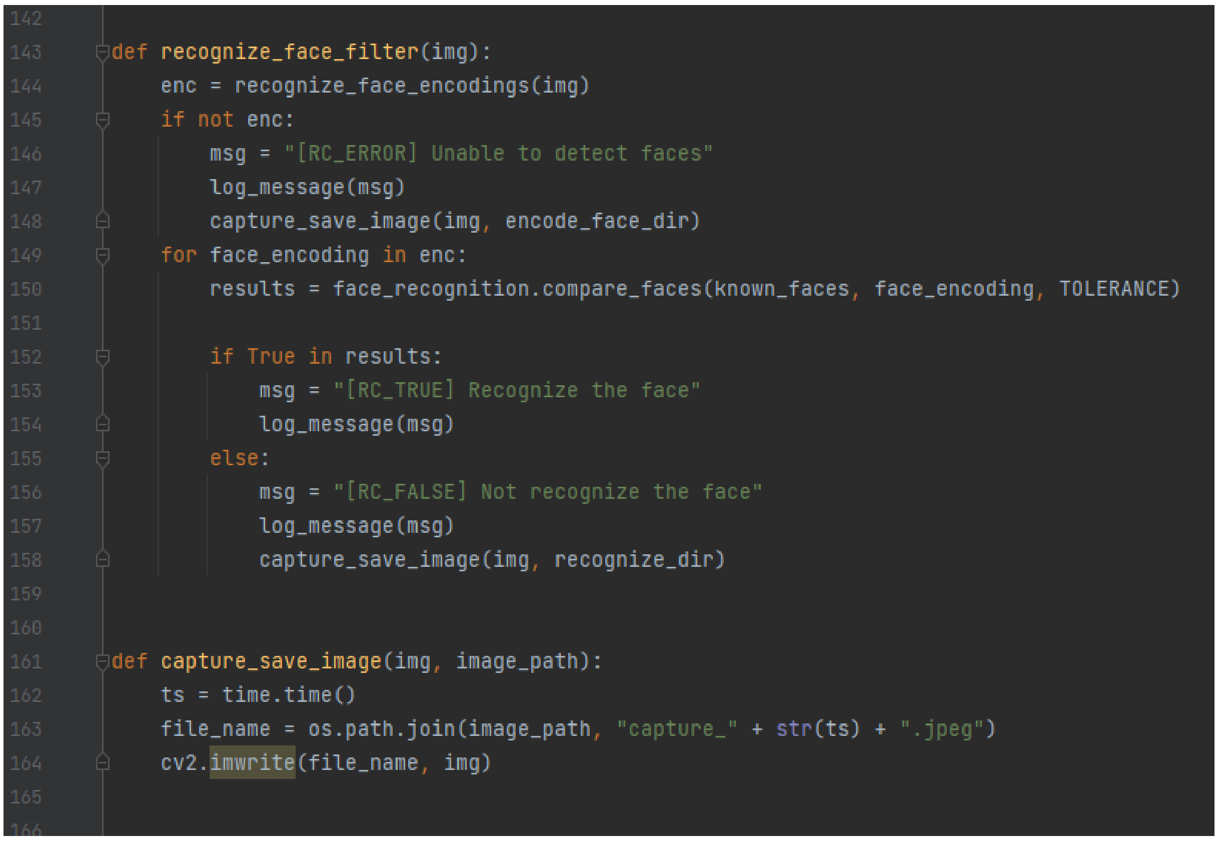
```python

def get_data(img):
    data = []
    try:
        img_arr = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        resized_arr = cv2.resize(img_arr, (img_size, img_size))  # Reshaping images to preferred size
        data.append([resized_arr, 0])
    except Exception as e:
        print(e)
    return np.array(data)


def ml_filter(cam_img, model):
    val = get_data(cam_img)
    # Data Preprocessing and Data Augmentation
    x_val = []
    y_val = []

    for feature, label in val:
        x_val.append(feature)
        y_val.append(label)

    x_val = np.array(x_val) / 255

    x_val.reshape(-1, img_size, img_size, 1)
    y_val = np.array(y_val)

    predictions = np.argmax(model.predict(x_val), axis=1)
    predictions = predictions.reshape(1, -1)[0]

    if predictions[0] == 0:
        msg = "[ML_Accept] Accepted the image"
        log_message(msg)
        return True
```



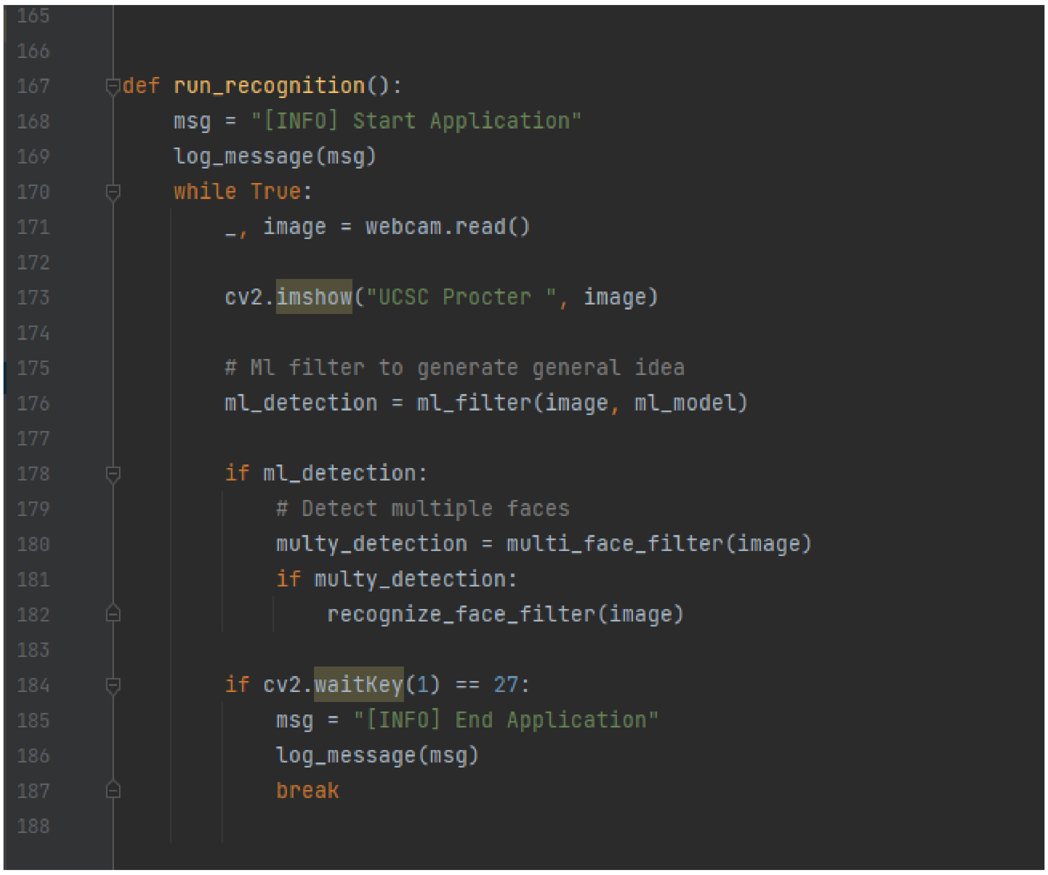
```python

    if predictions[0] == 1:
        msg = "[ML_Reject] reject the image"
        log_message(msg)
        capture_save_image(cam_img, ml_reject_dir)
        return False


# duplicate Filtering
def multi_face_filter(img):
    image_input = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = mp_face.process(image_input)

    if not results.detections:
        msg = "[FD_NO_FACE] No faces detected"
        log_message(msg)
        capture_save_image(img, no_face_dir)
        return False
    elif len(results.detections) == 1:
        msg = "[FD_SI_FACE] Single face detected"
        log_message(msg)
        return True
    else:
        msg = "[FD_MF_FACE] Multiple faces detected" + str(len(results.detections))
        log_message(msg)
        capture_save_image(img, multi_face_dir)
        return False

```



```python

# Facial Recognition

KNOWN_FACES_DIR = 'subject'
TOLERANCE = 0.5
FRAME_THICKNESS = 3
FONT_THICKNESS = 2
MODEL = 'hog'  # default: 'hog', other one can be 'cnn' - CUDA accelerated (if available) deep-learning pretrained model
subject_encode_file = 'subject_encode.pkl'

known_faces = []

with open(subject_encode_file, 'rb') as f:
    known_faces = pickle.load(f)


def recognize_face_encodings(img):
    locations = face_recognition.face_locations(img, model=MODEL)
    encodings = face_recognition.face_encodings(img, locations)
    return encodings

```

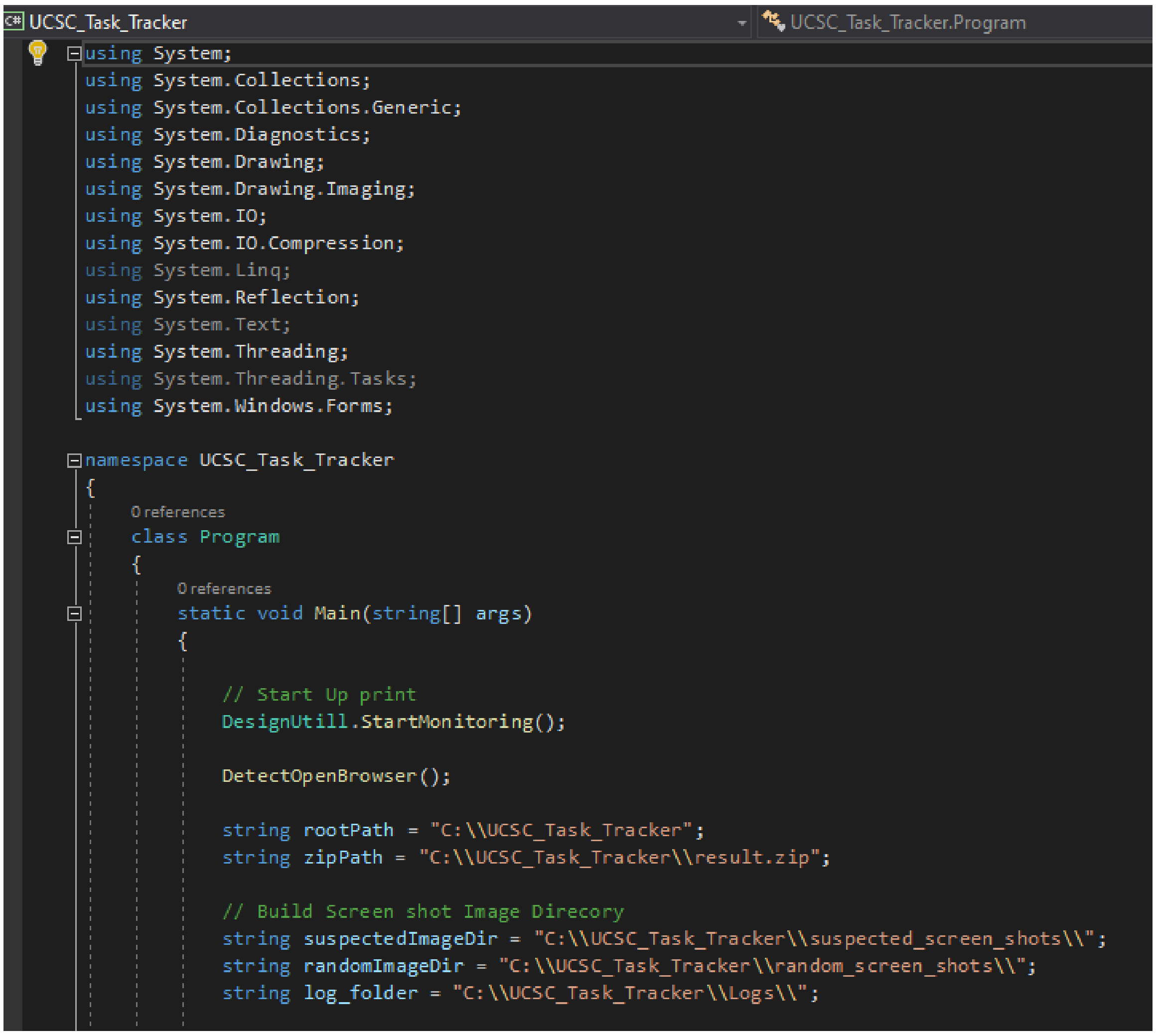


```python

def recognize_face_filter(img):
    enc = recognize_face_encodings(img)
    if not enc:
        msg = "[RC_ERROR] Unable to detect faces"
        log_message(msg)
        capture_save_image(img, encode_face_dir)
    for face_encoding in enc:
        results = face_recognition.compare_faces(known_faces, face_encoding, TOLERANCE)

        if True in results:
            msg = "[RC_TRUE] Recognize the face"
            log_message(msg)
        else:
            msg = "[RC_FALSE] Not recognize the face"
            log_message(msg)
            capture_save_image(img, recognize_dir)


def capture_save_image(img, image_path):
    ts = time.time()
    file_name = os.path.join(image_path, "capture_" + str(ts) + ".jpeg")
    cv2.imwrite(file_name, img)

```



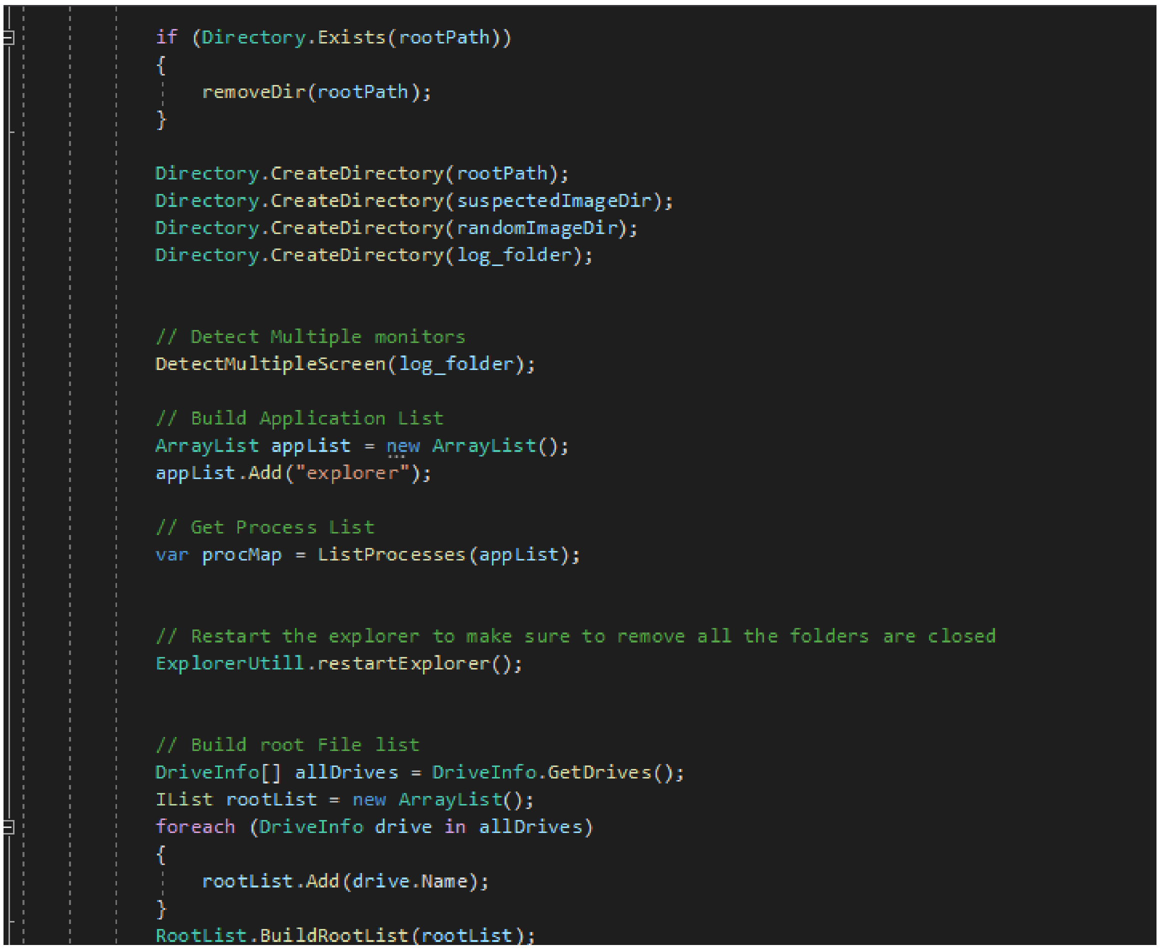
```python


def run_recognition():
    msg = "[INFO] Start Application"
    log_message(msg)
    while True:
        _, image = webcam.read()

        cv2.imshow("UCSC Procter ", image)

        # Ml filter to generate general idea
        ml_detection = ml_filter(image, ml_model)

        if ml_detection:
            # Detect multiple faces
            multy_detection = multi_face_filter(image)
            if multy_detection:
                recognize_face_filter(image)

        if cv2.waitKey(1) == 27:
            msg = "[INFO] End Application"
            log_message(msg)
            break

```

.net based OS activity tracking module

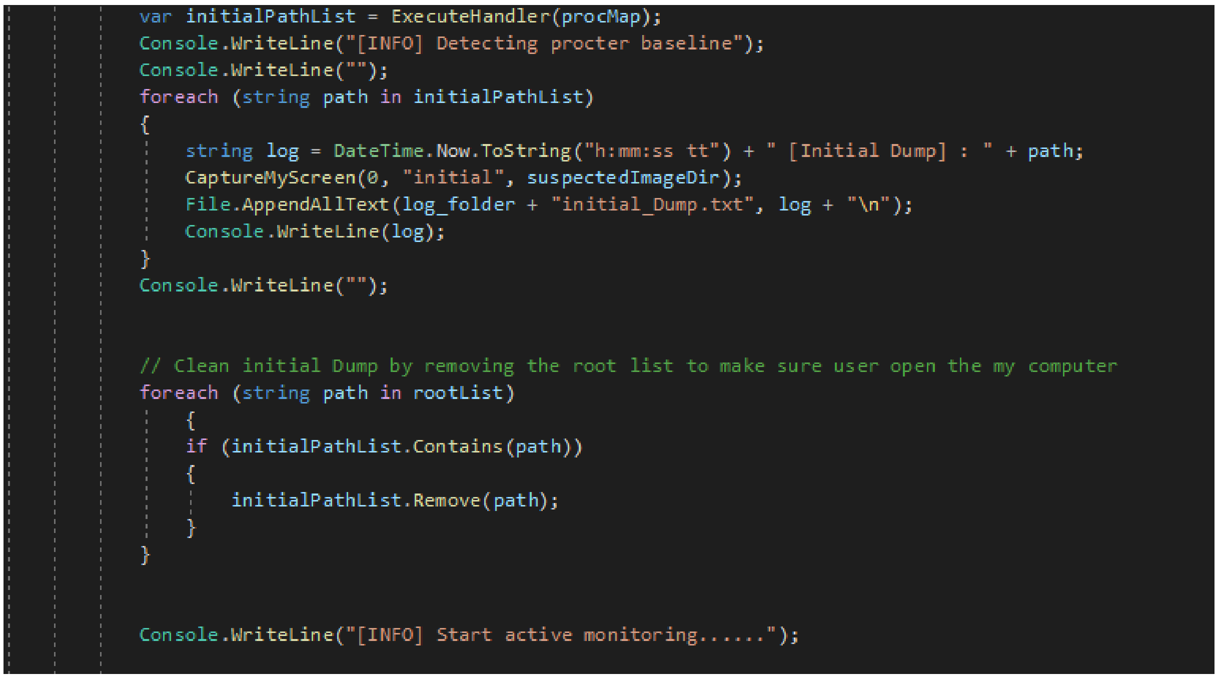


UCSC_Task_Tracker | UCSC_Task_Tracker.Program

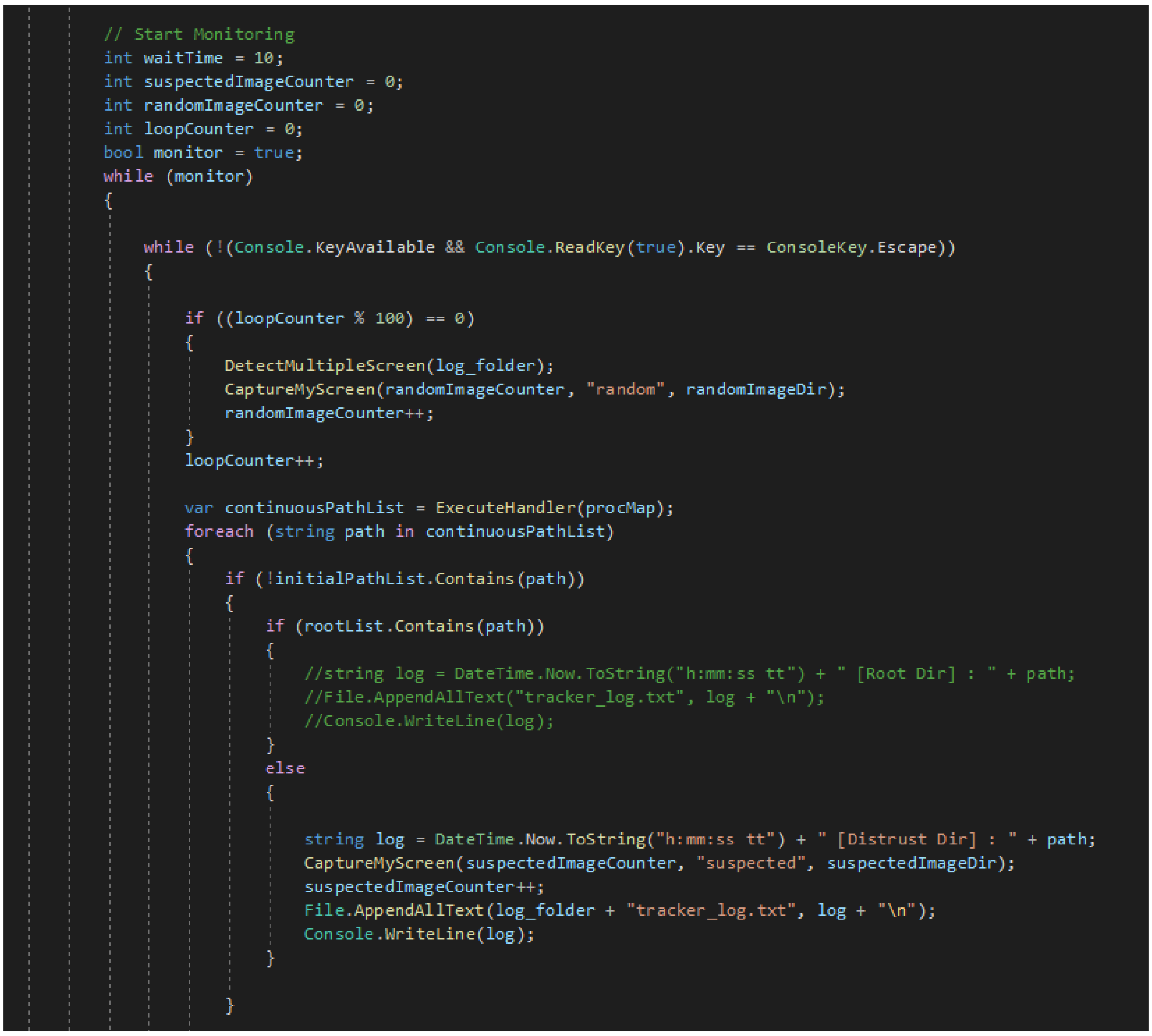
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.IO.Compression;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace UCSC_Task_Tracker
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {

            // Start Up print
            DesignUtill.StartMonitoring();

            DetectOpenBrowser();

            string rootPath = "C:\\UCSC_Task_Tracker";
            string zipPath = "C:\\UCSC_Task_Tracker\\result.zip";

            // Build Screen shot Image Direcory
            string suspectedImageDir = "C:\\UCSC_Task_Tracker\\suspected_screen_shots\\";
            string randomImageDir = "C:\\UCSC_Task_Tracker\\random_screen_shots\\";
            string log_folder = "C:\\UCSC_Task_Tracker\\Logs\\";
```



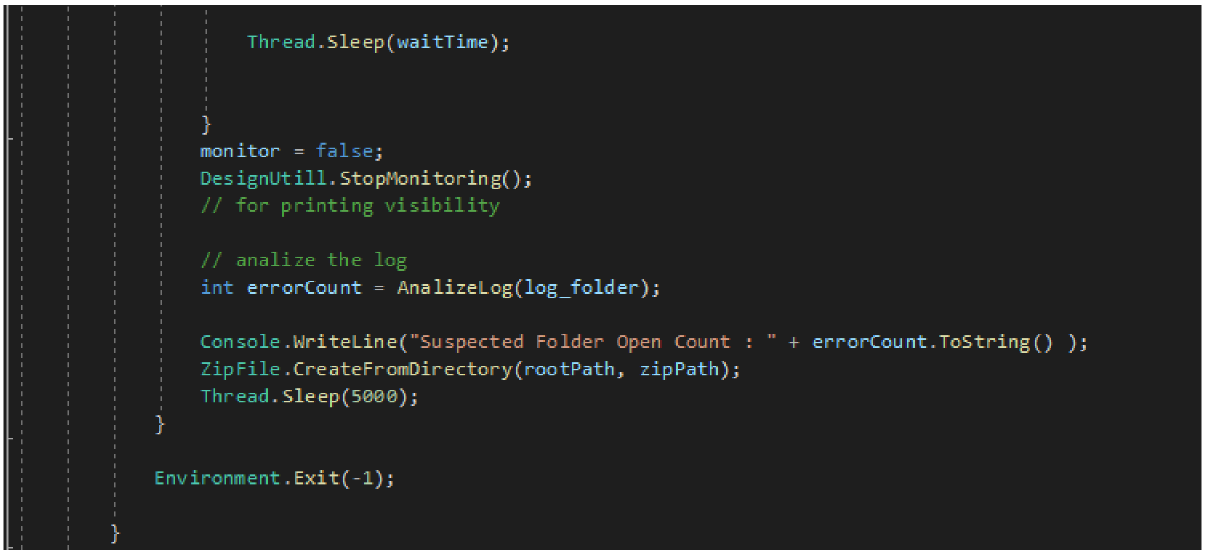
```
if (Directory.Exists(rootPath))
{
    removeDir(rootPath);
}

Directory.CreateDirectory(rootPath);
Directory.CreateDirectory(suspectedImageDir);
Directory.CreateDirectory(randomImageDir);
Directory.CreateDirectory(log_folder);


// Detect Multiple monitors
DetectMultipleScreen(log_folder);

// Build Application List
ArrayList appList = new ArrayList();
appList.Add("explorer");

// Get Process List
var procMap = ListProcesses(appList);


// Restart the explorer to make sure to remove all the folders are closed
ExplorerUtill.restartExplorer();


// Build root File list
DriveInfo[] allDrives = DriveInfo.GetDrives();
IList rootList = new ArrayList();
foreach (DriveInfo drive in allDrives)
{
    rootList.Add(drive.Name);
}
RootList.BuildRootList(rootList);
```



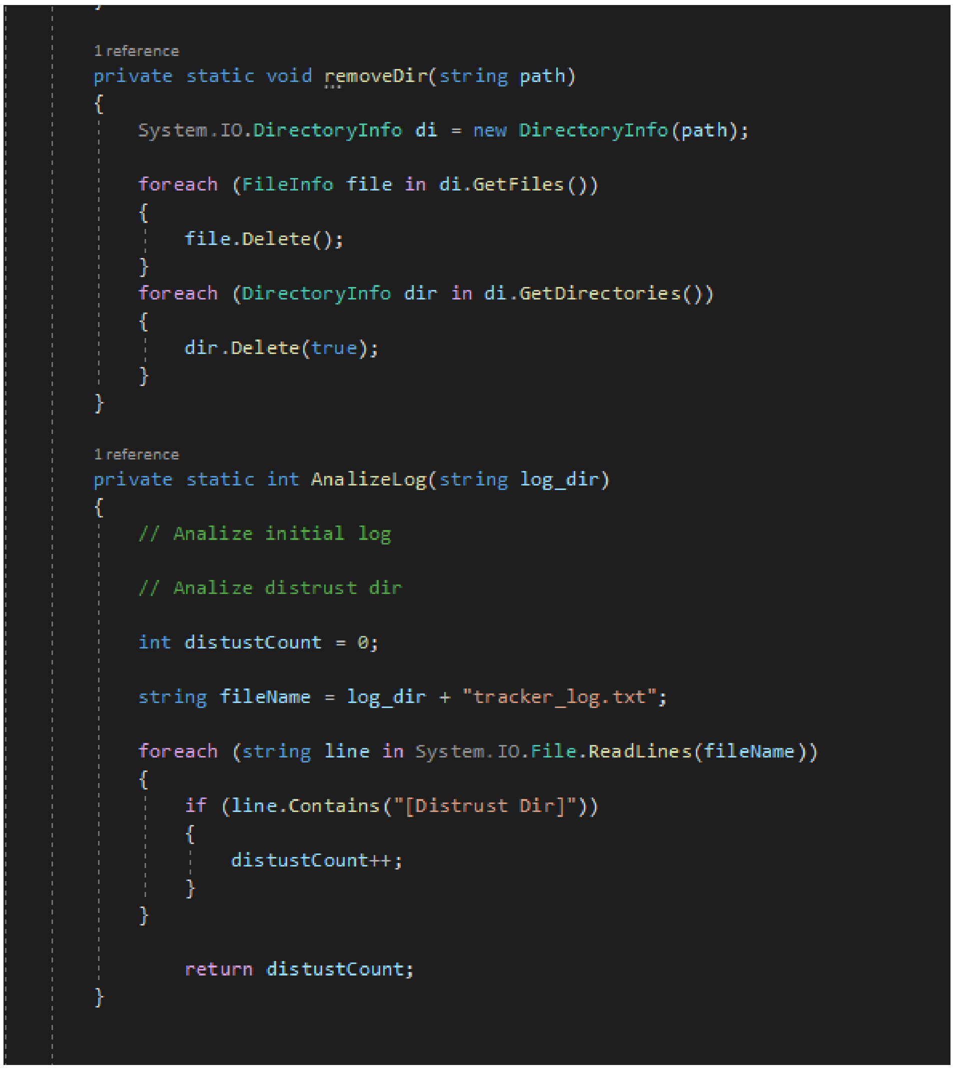
```
var initialPathList = ExecuteHandler(procMap);
Console.WriteLine("[INFO] Detecting procter baseline");
Console.WriteLine("");
foreach (string path in initialPathList)
{
    string log = DateTime.Now.ToString("h:mm:ss tt") + " [Initial Dump] : " + path;
    CaptureMyScreen(0, "initial", suspectedImageDir);
    File.AppendAllText(log_folder + "initial_Dump.txt", log + "\n");
    Console.WriteLine(log);
}
Console.WriteLine("");


// Clean initial Dump by removing the root list to make sure user open the my computer
foreach (string path in rootList)
    {
    if (initialPathList.Contains(path))
    {
        initialPathList.Remove(path);
    }
}


Console.WriteLine("[INFO] Start active monitoring......");
```



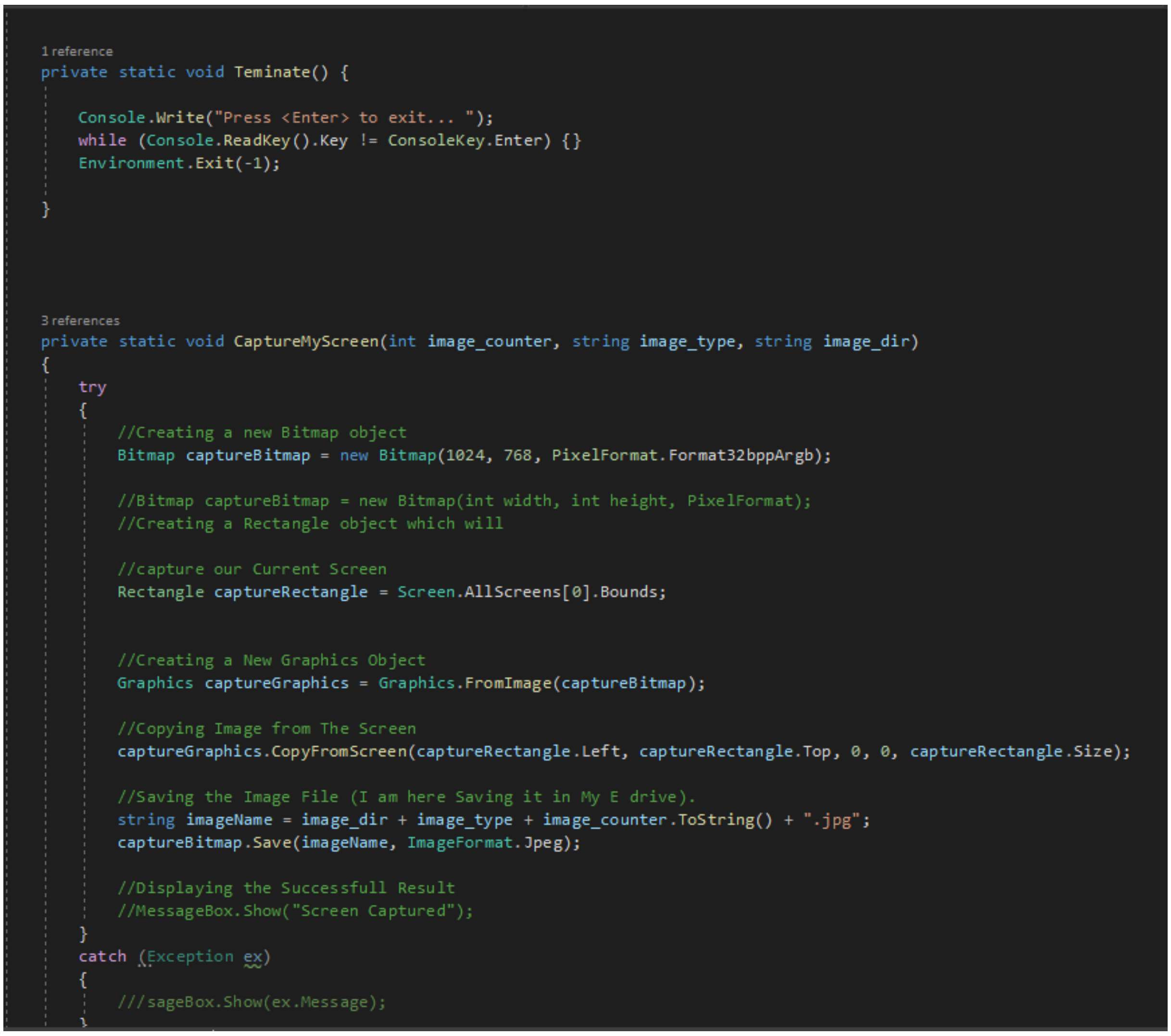
```
    // Start Monitoring
    int waitTime = 10;
    int suspectedImageCounter = 0;
    int randomImageCounter = 0;
    int loopCounter = 0;
    bool monitor = true;
    while (monitor)
    {

        while (!(Console.KeyAvailable && Console.ReadKey(true).Key == ConsoleKey.Escape))
        {

            if ((loopCounter % 100) == 0)
            {
                DetectMultipleScreen(log_folder);
                CaptureMyScreen(randomImageCounter, "random", randomImageDir);
                randomImageCounter++;
            }
            loopCounter++;

            var continuousPathList = ExecuteHandler(procMap);
            foreach (string path in continuousPathList)
            {
                if (!initialPathList.Contains(path))
                {
                    if (rootList.Contains(path))
                    {
                        //string log = DateTime.Now.ToString("h:mm:ss tt") + " [Root Dir] : " + path;
                        //File.AppendAllText("tracker_log.txt", log + "\n");
                        //Console.WriteLine(log);
                    }
                    else
                    {

                        string log = DateTime.Now.ToString("h:mm:ss tt") + " [Distrust Dir] : " + path;
                        CaptureMyScreen(suspectedImageCounter, "suspected", suspectedImageDir);
                        suspectedImageCounter++;
                        File.AppendAllText(log_folder + "tracker_log.txt", log + "\n");
                        Console.WriteLine(log);
                    }

                }
```



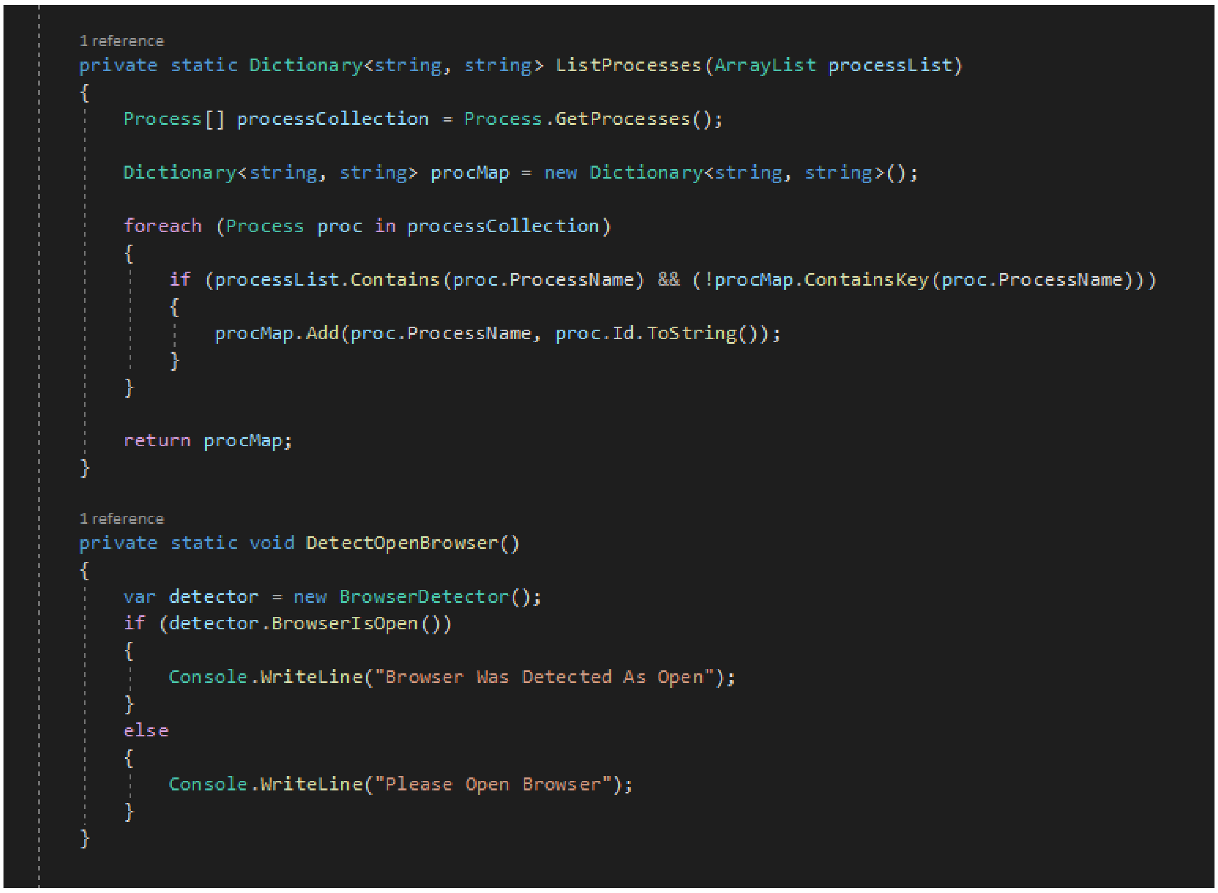
```
            Thread.Sleep(waitTime);


        }
        monitor = false;
        DesignUtill.StopMonitoring();
        // for printing visibility

        // analize the log
        int errorCount = AnalizeLog(log_folder);

        Console.WriteLine("Suspected Folder Open Count : " + errorCount.ToString() );
        ZipFile.CreateFromDirectory(rootPath, zipPath);
        Thread.Sleep(5000);
    }

    Environment.Exit(-1);

}
```



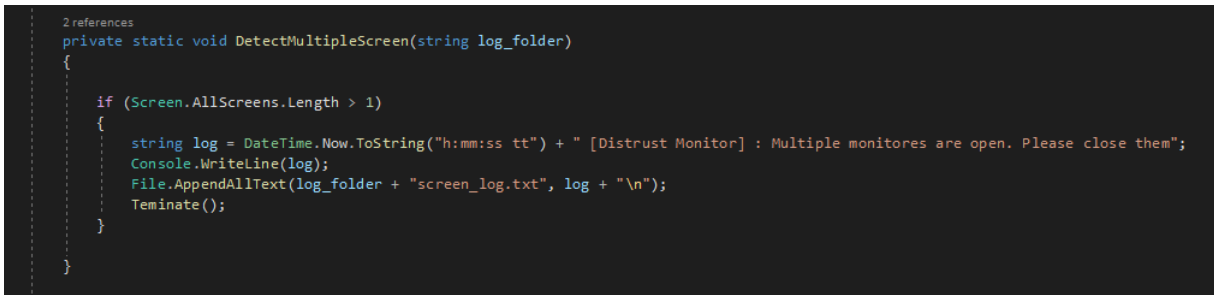
```
        1 reference
        private static void removeDir(string path)
        {
            System.IO.DirectoryInfo di = new DirectoryInfo(path);

            foreach (FileInfo file in di.GetFiles())
            {
                file.Delete();
            }
            foreach (DirectoryInfo dir in di.GetDirectories())
            {
                dir.Delete(true);
            }
        }

        1 reference
        private static int AnalizeLog(string log_dir)
        {
            // Analize initial log

            // Analize distrust dir

            int distustCount = 0;

            string fileName = log_dir + "tracker_log.txt";

            foreach (string line in System.IO.File.ReadLines(fileName))
            {
                if (line.Contains("[Distrust Dir]"))
                {
                    distustCount++;
                }
            }

                return distustCount;
        }
```



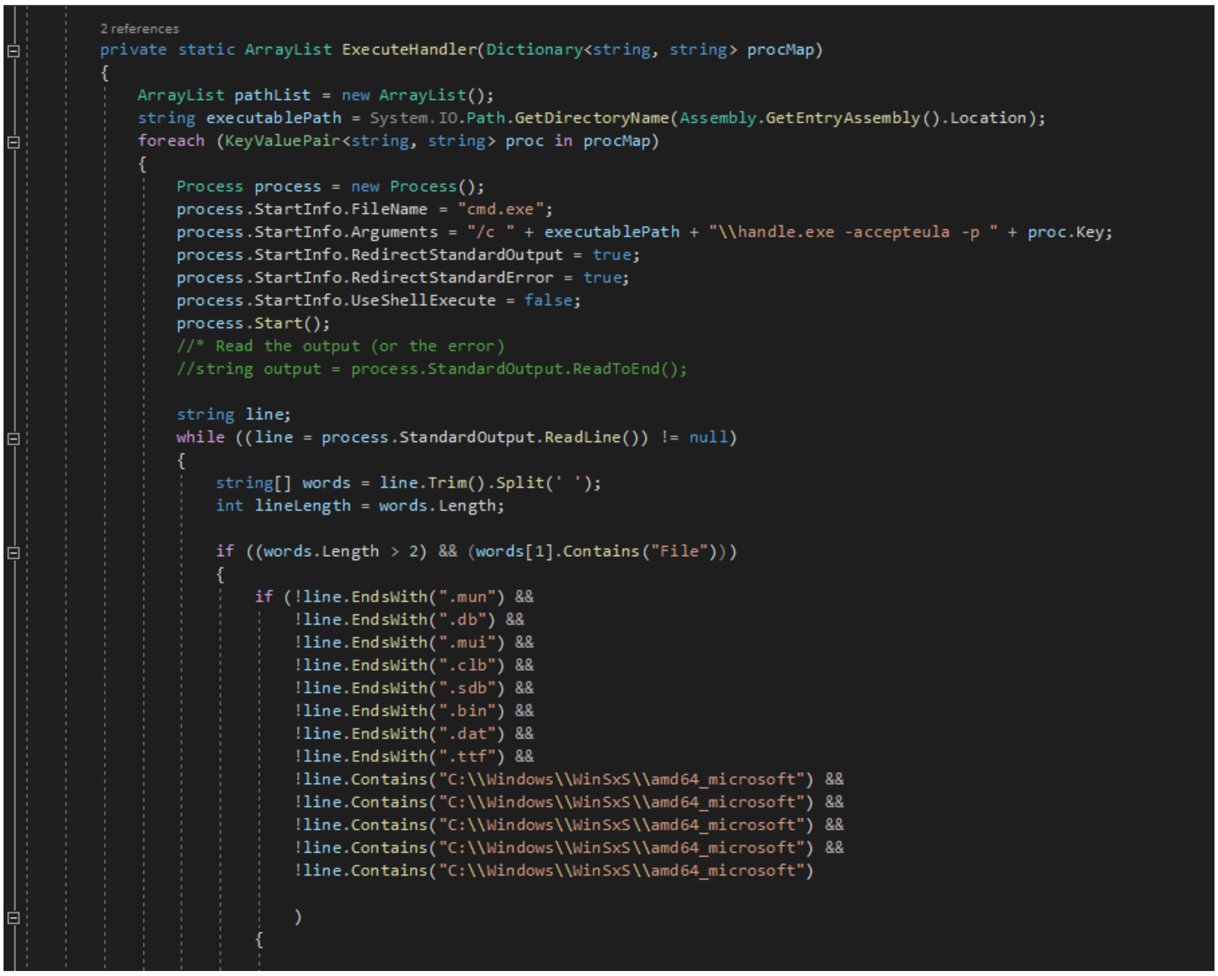
```
1 reference
private static void Teminate() {

    Console.Write("Press <Enter> to exit... ");
    while (Console.ReadKey().Key != ConsoleKey.Enter) {}
    Environment.Exit(-1);

}



3 references
private static void CaptureMyScreen(int image_counter, string image_type, string image_dir)
{
    try
    {
        //Creating a new Bitmap object
        Bitmap captureBitmap = new Bitmap(1024, 768, PixelFormat.Format32bppArgb);

        //Bitmap captureBitmap = new Bitmap(int width, int height, PixelFormat);
        //Creating a Rectangle object which will

        //capture our Current Screen
        Rectangle captureRectangle = Screen.AllScreens[0].Bounds;


        //Creating a New Graphics Object
        Graphics captureGraphics = Graphics.FromImage(captureBitmap);

        //Copying Image from The Screen
        captureGraphics.CopyFromScreen(captureRectangle.Left, captureRectangle.Top, 0, 0, captureRectangle.Size);

        //Saving the Image File (I am here Saving it in My E drive).
        string imageName = image_dir + image_type + image_counter.ToString() + ".jpg";
        captureBitmap.Save(imageName, ImageFormat.Jpeg);

        //Displaying the Successfull Result
        //MessageBox.Show("Screen Captured");
    }
    catch (Exception ex)
    {
        ///sageBox.Show(ex.Message);
```



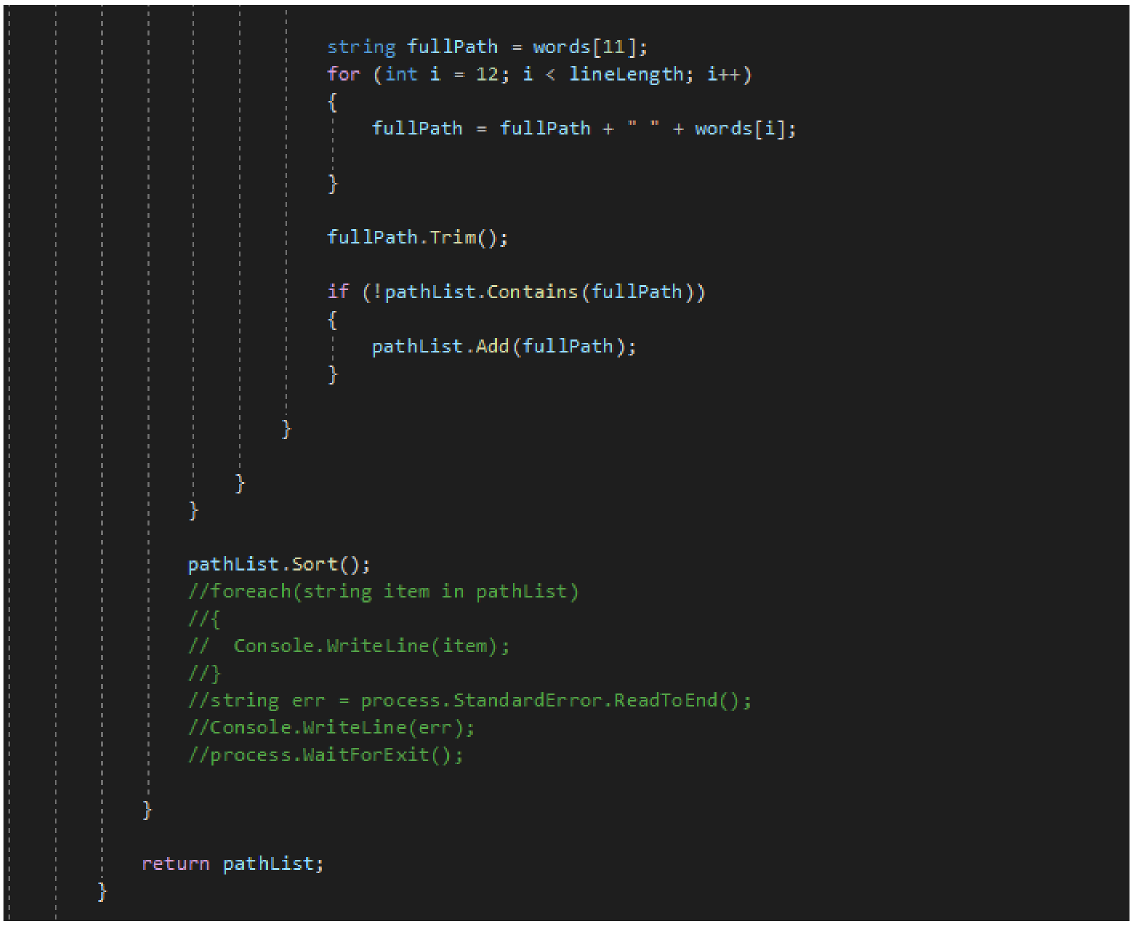
```
1 reference
private static Dictionary<string, string> ListProcesses(ArrayList processList)
{
    Process[] processCollection = Process.GetProcesses();

    Dictionary<string, string> procMap = new Dictionary<string, string>();

    foreach (Process proc in processCollection)
    {
        if (processList.Contains(proc.ProcessName) && (!procMap.ContainsKey(proc.ProcessName)))
        {
            procMap.Add(proc.ProcessName, proc.Id.ToString());
        }
    }

    return procMap;
}

1 reference
private static void DetectOpenBrowser()
{
    var detector = new BrowserDetector();
    if (detector.BrowserIsOpen())
    {
        Console.WriteLine("Browser Was Detected As Open");
    }
    else
    {
        Console.WriteLine("Please Open Browser");
    }
}
```



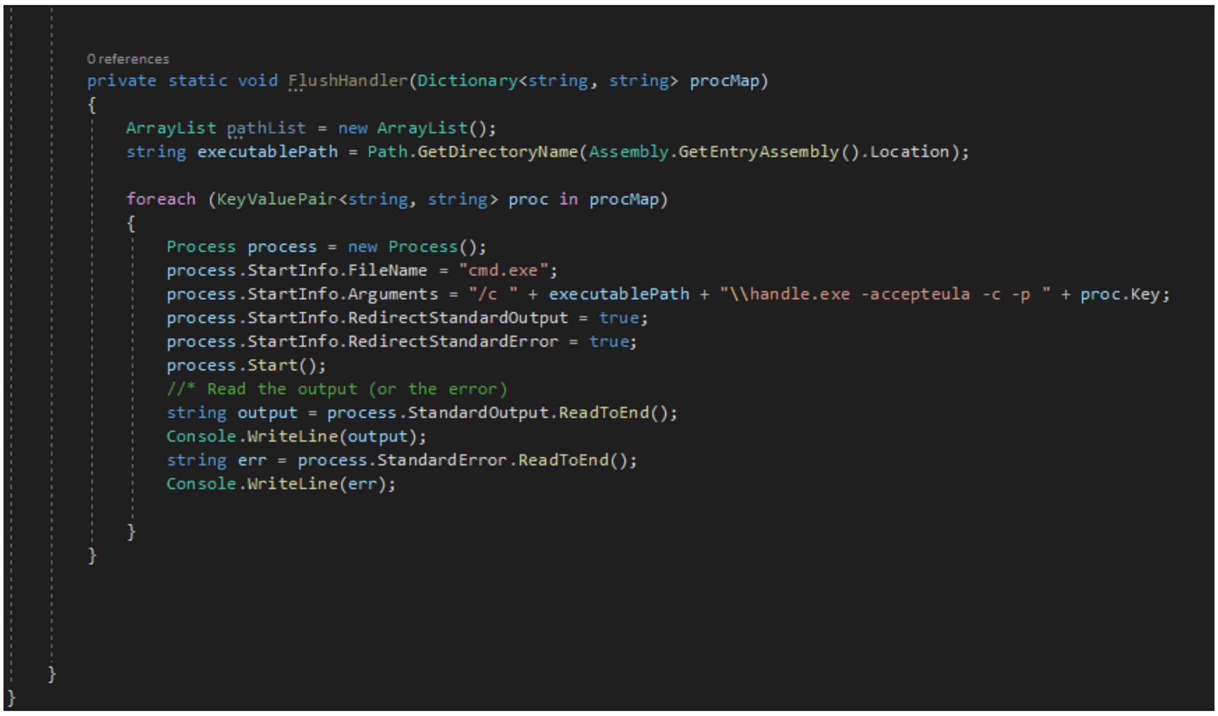
```
2 references
private static void DetectMultipleScreen(string log_folder)
{

    if (Screen.AllScreens.Length > 1)
    {
        string log = DateTime.Now.ToString("h:mm:ss tt") + " [Distrust Monitor] : Multiple monitores are open. Please close them";
        Console.WriteLine(log);
        File.AppendAllText(log_folder + "screen_log.txt", log + "\n");
        Teminate();
    }

}
```



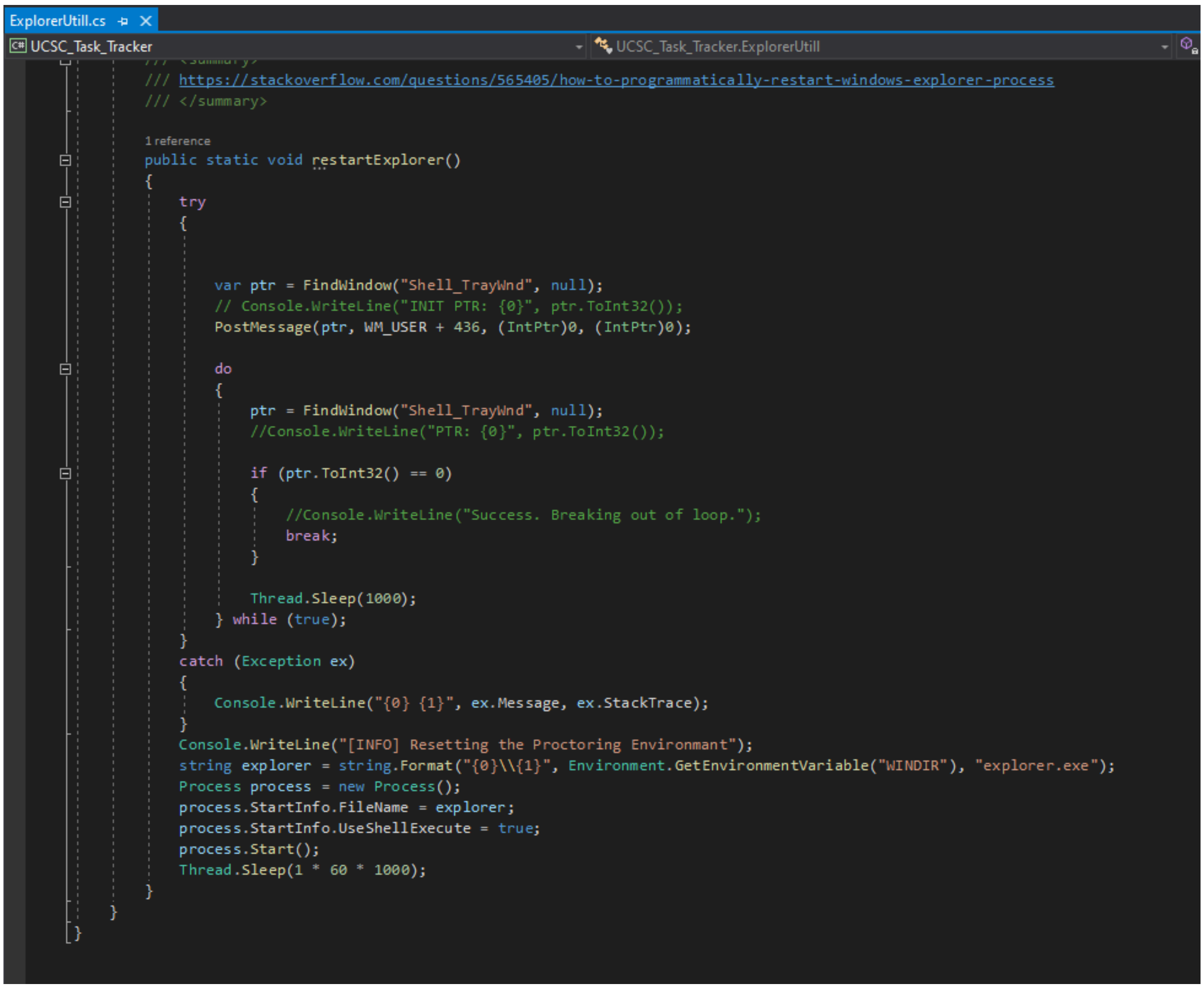
```
2 references
private static ArrayList ExecuteHandler(Dictionary<string, string> procMap)
{
    ArrayList pathList = new ArrayList();
    string executablePath = System.IO.Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);
    foreach (KeyValuePair<string, string> proc in procMap)
    {
        Process process = new Process();
        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.Arguments = "/c " + executablePath + "\\handle.exe -accepteula -p " + proc.Key;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.RedirectStandardError = true;
        process.StartInfo.UseShellExecute = false;
        process.Start();
        //* Read the output (or the error)
        //string output = process.StandardOutput.ReadToEnd();

        string line;
        while ((line = process.StandardOutput.ReadLine()) != null)
        {
            string[] words = line.Trim().Split(' ');
            int lineLength = words.Length;

            if ((words.Length > 2) && (words[1].Contains("File")))
            {
                if (!line.EndsWith(".mun") &&
                    !line.EndsWith(".db") &&
                    !line.EndsWith(".mui") &&
                    !line.EndsWith(".clb") &&
                    !line.EndsWith(".sdb") &&
                    !line.EndsWith(".bin") &&
                    !line.EndsWith(".dat") &&
                    !line.EndsWith(".ttf") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft") &&
                    !line.Contains("C:\\Windows\\WinSxS\\amd64_microsoft")

                    )
                {
```



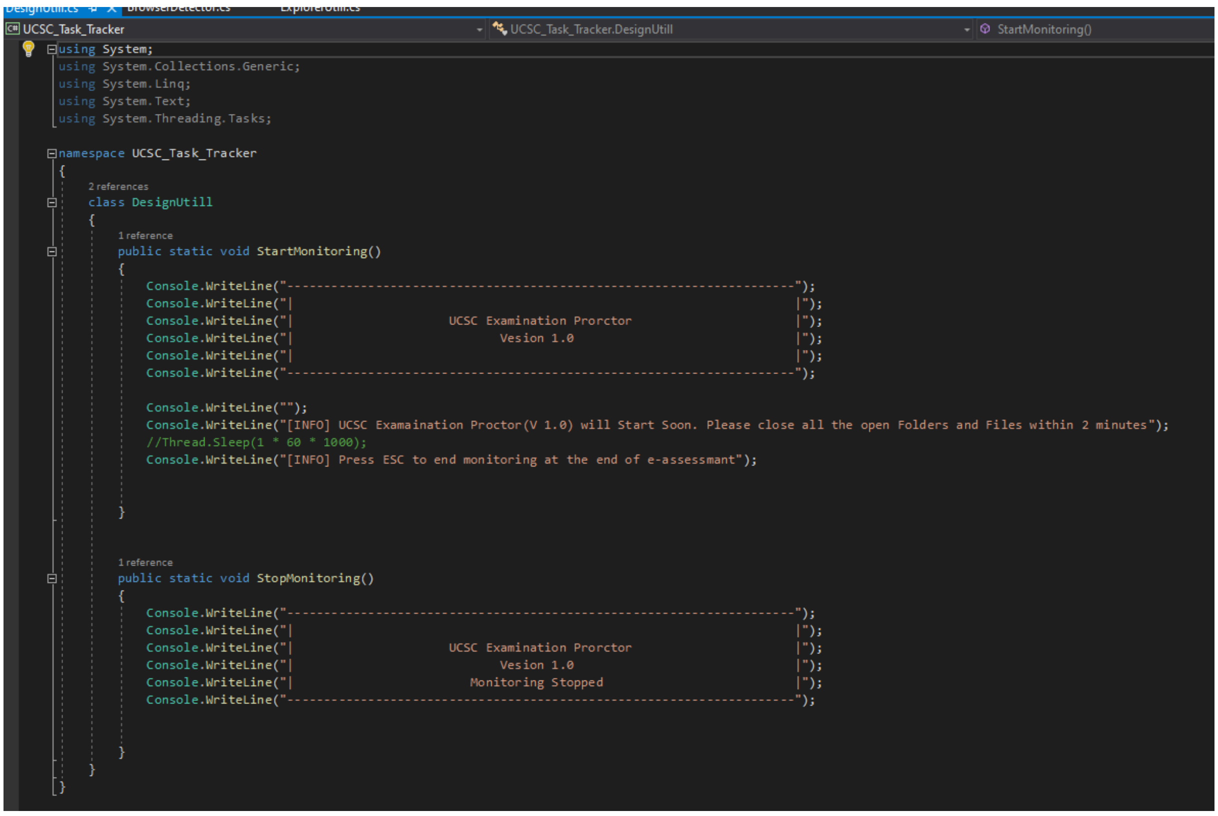
```
                            string fullPath = words[11];
                            for (int i = 12; i < lineLength; i++)
                            {
                                fullPath = fullPath + " " + words[i];

                            }

                            fullPath.Trim();

                            if (!pathList.Contains(fullPath))
                            {
                                pathList.Add(fullPath);
                            }

                        }

                    }
                }

                pathList.Sort();
                //foreach(string item in pathList)
                //{
                //  Console.WriteLine(item);
                //}
                //string err = process.StandardError.ReadToEnd();
                //Console.WriteLine(err);
                //process.WaitForExit();

            }

            return pathList;
        }
```



```
        0 references
        private static void FlushHandler(Dictionary<string, string> procMap)
        {
            ArrayList pathList = new ArrayList();
            string executablePath = Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);

            foreach (KeyValuePair<string, string> proc in procMap)
            {
                Process process = new Process();
                process.StartInfo.FileName = "cmd.exe";
                process.StartInfo.Arguments = "/c " + executablePath + "\\handle.exe -accepteula -c -p " + proc.Key;
                process.StartInfo.RedirectStandardOutput = true;
                process.StartInfo.RedirectStandardError = true;
                process.Start();
                //* Read the output (or the error)
                string output = process.StandardOutput.ReadToEnd();
                Console.WriteLine(output);
                string err = process.StandardError.ReadToEnd();
                Console.WriteLine(err);

            }
        }



    }
}
```



```
        /// https://stackoverflow.com/questions/565405/how-to-programmatically-restart-windows-explorer-process
        /// </summary>

        1 reference
        public static void restartExplorer()
        {
            try
            {


                var ptr = FindWindow("Shell_TrayWnd", null);
                // Console.WriteLine("INIT PTR: {0}", ptr.ToInt32());
                PostMessage(ptr, WM_USER + 436, (IntPtr)0, (IntPtr)0);

                do
                {
                    ptr = FindWindow("Shell_TrayWnd", null);
                    //Console.WriteLine("PTR: {0}", ptr.ToInt32());

                    if (ptr.ToInt32() == 0)
                    {
                        //Console.WriteLine("Success. Breaking out of loop.");
                        break;
                    }

                    Thread.Sleep(1000);
                } while (true);
            }
            catch (Exception ex)
            {
                Console.WriteLine("{0} {1}", ex.Message, ex.StackTrace);
            }
            Console.WriteLine("[INFO] Resetting the Proctoring Environment");
            string explorer = string.Format("{0}\\{1}", Environment.GetEnvironmentVariable("WINDIR"), "explorer.exe");
            Process process = new Process();
            process.StartInfo.FileName = explorer;
            process.StartInfo.UseShellExecute = true;
            process.Start();
            Thread.Sleep(1 * 60 * 1000);
        }
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace UCSC_Task_Tracker
{
    2 references
    class DesignUtill
    {
        1 reference
        public static void StartMonitoring()
        {
            Console.WriteLine("----------------------------------------------------------------------");
            Console.WriteLine("|                                                                    |");
            Console.WriteLine("|                      UCSC Examination Prorctor                     |");
            Console.WriteLine("|                              Vesion 1.0                            |");
            Console.WriteLine("|                                                                    |");
            Console.WriteLine("----------------------------------------------------------------------");

            Console.WriteLine("");
            Console.WriteLine("[INFO] UCSC Examaination Proctor(V 1.0) will Start Soon. Please close all the open Folders and Files within 2 minutes");
            //Thread.Sleep(1 * 60 * 1000);
            Console.WriteLine("[INFO] Press ESC to end monitoring at the end of e-assessmant");


        }


        1 reference
        public static void StopMonitoring()
        {
            Console.WriteLine("----------------------------------------------------------------------");
            Console.WriteLine("|                                                                    |");
            Console.WriteLine("|                      UCSC Examination Prorctor                     |");
            Console.WriteLine("|                              Vesion 1.0                            |");
            Console.WriteLine("|                          Monitoring Stopped                        |");
            Console.WriteLine("----------------------------------------------------------------------");


        }
    }
}
```

XL