AVOID – AI Vision-based Online cheating Interception and Detection

A PROJECT REPORT

Submitted by

HEMA HARIHARAN S	**-**	**513520104009**
JEEVANANDHAM S	**-**	**513520104012**
KABILAN V	**-**	**513520104014**
VIJAYALAYAN K	**-**	**513520104041**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

ARAPAKKAM, RANIPET-632517.

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL/MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"AVOID – AI Vision-based Online cheating Interception and Detection"** is the bonafide work of **"HEMA HARIHARAN S (513520104009), JEEVANANDHAM S (513520104012), KABILAN V (513520104014), VIJAYALAYAN K (513520104041)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mr.S.SRINIVASAN M.TECH,(Ph.D).	**Mr.S.SRINIVASAN M.TECH,(Ph.D).**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Dept. of Computer Science and Engg,	**Dept. of Computer Science and Engg,**
Annai Mira College of Engineering	**Annai Mira College of Engineering**
and Technology, Arapakkam	**and Technology, Arapakkam**
Ranipet-632517	**Ranipet-632517**

Submitted for the project viva voce held on ________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I first offer my deepest gratitude to Almighty God who has given me strength and good health during the course of this project.

I am conscious about my indebtedness our Honorable Chairman **Mr.S.Ramadoss**, Secretary **Mr.G.Thamothiran** and our Principal **Dr.T.K.Gopinathan, Ph.D.**, and our Vice Principal **Dr.D.Saravanan, Ph.D.**, who inspired me reach greater heights in the pursuits of knowledge.

We proudly express our esteemed gratitude to our Head of the Department **Mr.S.SRINIVASAN, M.TECH,(Ph.D)** for his encouragement and assistance towards the completion of the project.

Special thanks must be mentioned to our internal guide **Mr.S.SRINIVASAN, M.TECH,(Ph.D)**, Head of the Department, Computer Science Department for his expert advice, valuable information and guidance throughput the completion of the project. We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

We also express our sincere thanks to our project coordinator, guide and all our staff of CSE Department who helped us in the completion of this project.

TABLE OF CONTENT

AVOID – AI Vision-based Online cheating Interception and Detection

ABSTRACT

Online cheating has become a pervasive problem in domains like online exams and gaming, challenging the integrity and fairness of such activities. To address this issue, we propose an AI vision-based online cheating interception and detection system. The system captures video streams from users' webcams or screen-sharing features and applies advanced computer vision algorithms for cheating detection. It undergoes video stream acquisition, pre-processing, feature extraction, and classification stages. Privacy and permissions are respected during video acquisition, while pre-processing techniques enhance video quality.

Feature extraction involves analysing visual cues like facial movements. These features are inputted into a deep neural network for classification. The trained model distinguishes normal user behaviour from cheating actions based on extracted features, providing a probability score for cheating likelihood.

Efficient algorithms enable real-time monitoring and timely intervention. The system quickly processes video frames, triggering alerts or notifications upon detecting suspicious cheating behaviours. Instructors or administrators can take appropriate actions, such as pausing exams or conducting investigations, to maintain fairness and integrity. Overall, our proposed system offers an automated and scalable solution for combating cheating in online environments, ensuring credibility and trustworthiness in online assessments and competitions.

LIST OF FIGURES

LIST OF ABBREVIATIONS

OpenCV	- Open-Source Computer Vision
GUI	- Graphical User Interface
UML	- Unified Modeling Language

INTRODUCTION

CHAPTER 1

1. INTRODUTION

1.1. PROBLEM STATEMENT

In the modern era of online education and remote learning, the prevalence of cheating during exams and assessments has become a significant concern. Traditional methods of invigilation are not effective in preventing online cheating, as students can easily access external resources and collaborate with others without detection. To address this issue, there is a need for an AI vision-based system that can intercept and detect cheating behavior during online exams, ensuring academic integrity and fairness.

The problem at hand is to develop an intelligent system that can monitor and analyze students' activities during online exams to identify potential cheating behaviors. The system should be able to detect various forms of cheating, including but not limited to:

- Copying from external sources
- Collaborative cheating
- Suspicious behavior

1.2. THE SOLUTION

The AI vision-based online cheating interception and detection project proposes a comprehensive solution to address the problem of cheating during online exams. The solution involves the development of an intelligent system that utilizes computer vision techniques and deep learning algorithms to monitor and analyze students' activities in real-time. Here is an outline of the solution:

1) Video and Screen Monitoring: The system captures video feeds from students' webcams during the exam and records their screen activity. This allows for comprehensive monitoring of students' actions and behavior.
2) Behavior Analysis: The system analyzes students' behavior during the exam to identify suspicious activities. It tracks eye movements and other actions that may indicate cheating attempts or distractions.
3) Real-time Alerts and Notifications: When the system detects potential cheating behavior, it generates real-time alerts and notifications for educators or proctors. This allows them to intervene immediately and take appropriate actions.

By implementing this AI vision-based solution, educators and institutions can effectively intercept and detect cheating behaviors during online exams. The system ensures academic integrity, promotes fair assessments, and maintains a secure and trustworthy learning environment in the realm of remote education.

1.3. EXISTING SYSTEM

1.3.1. EXISTING CONCEPT

The current system for AI vision-based online cheating interception and detection is limited, as the technology and methods for detecting cheating behavior during online exams are still evolving. However, some existing approaches and technologies can be utilized as part of the system. Here are a few examples:

1) Screen Recording and Remote Proctoring: Many online learning platforms and examination systems incorporate screen recording and remote proctoring features. These tools capture video feeds and screen activity during exams, allowing educators or proctors to review the recordings for any suspicious behavior or signs of cheating.
2) Browser Monitoring: Some online examination systems use browser monitoring techniques to restrict access to external websites or resources during exams. They can track and log students' internet activity to ensure they do not navigate to unauthorized sources.

1.3.2. DRAWBACKS

- High false positive rates are obtained
- It provides only Limited detection accuracy
- Limited flexibility

1.4. PROPOSED SYSTEM

1.4.1. PROPOSED CONCEPT

The proposed solution for the AI vision-based online cheating interception and detection project aims to develop a comprehensive system that utilizes advanced AI technologies to effectively intercept and detect cheating behavior during online exams. The system utilizes deep learning models trained on a large dataset of cheating and non-cheating behaviors.

These models can learn patterns, anomalies, and correlations to accurately detect cheating instances and improve detection over time. The system analyzes students' behavior during the exam to identify suspicious activities. It tracks eye movements, and other actions to detect potential cheating attempts or distractions.

The proposed solution leverages AI vision-based technologies to provide a robust and automated system for detecting cheating during online exams. By utilizing real-time monitoring, behavior analysis, and deep learning algorithms, the system aims to ensure academic integrity, promote fair assessments, and maintain a secure and trustworthy online learning environment.

1.4.2. ADVANTAGES

- Enhanced Accuracy
- Reduced false positives
- Real-time Monitoring
- Efficient and Time-saving

LITERATURE SURVEY

CHAPTER 2

2. LITERATURE SURVEY

2.1. ONLINE PROCTORING SYSTEM USING IMAGE PROCESSING AND MACHINE LEARNING

Authors: K. Gopalakrishnan, N. Dhiyaneshwaran , P. Yugesh

Year: 2022

The migration to online exams due to the COVID-19 pandemic has highlighted the need for secure authentication and proctoring of online students. This paper proposes a system that uses different biometric technologies to address this challenge. The system's performance is evaluated using real-time videos in various scenarios, showing improved accuracy compared to existing research. Online education has gained popularity, with over 7.1 million students participating in digital learning through mediums like animations, whiteboards, and PowerPoint. However, online exams pose a significant risk, as surveys indicate that 73% of students engage in cheating. The survey further reveals that a significant number of students paraphrase from the internet and books without proper citations or copy content word-for-word. To ensure the integrity of online exams and combat cheating, a robust authentication and proctoring system is essential.

2.2. SOME OTHER RESEARCHERS

Title : Online Proctoring System Using Deep Learning And Computer Vision Techniques

Authors: Jitendra and Singh

Year: 2021

Online proctoring is a method of monitoring students during online exams to prevent cheating. Traditional proctoring methods, such as in-person proctoring and webcam proctoring, have several limitations, such as the high cost of in-person proctoring and the inability of webcam proctoring to detect subtle cheating behaviors.

In this paper, we propose an online proctoring system using deep learning and computer vision techniques to monitor students' behavior during online exams. Our system uses a deep learning model to detect cheating behaviors, such as looking away from the screen, using unauthorized materials, and talking to another person.

We evaluated our system on a dataset of 1000 videos of students taking online exams. Our system was able to detect cheating behaviors with an accuracy of 95%.

Title :AI-Based Proctoring System for Detecting Cheating Behavior during Online Exams

Author: Zhang

Year: 2021

Online exams have become increasingly popular in recent years, due to the COVID-19 pandemic and other factors. However, online exams are also more susceptible to cheating than traditional in-person exams.

In this paper, we propose an AI-based proctoring system that utilizes audio and video analysis to detect cheating behavior during online exams.

Our system uses a deep learning model to analyze the audio and video data of students taking online exams. The model is trained to detect cheating behaviors, such as talking to another person, looking away from the screen, and using unauthorized materials

Our system is a promising new approach to online proctoring that can help to prevent cheating and ensure the integrity of online exams.

Title : A Systematic Review on AI-based Proctoring Systems

Author: Aditya Nigam, Rhitvik Pasricha, Tarishi Singh & Prathamesh Churi

Year: 2021

There have been giant leaps in the field of education in the past 1–2 years.. Schools and colleges are transitioning online to provide more resources to their students. The COVID-19 pandemic has provided students more opportunities to learn and improve themselves at their own pace. Online proctoring services (part of assessment) are also on the rise, and AI-based proctoring systems (henceforth called as AIPS) have taken the market by storm. There are various psychological, cultural and technological parameters need to be considered while developing AIPS. This paper systematically reviews existing AI and non-AI-based proctoring systems. Through the systematic search on Scopus, Web of Science and ERIC repositories, 43 paper were listed out from the year 2015 to 2021. We addressed 4 primary research questions which were focusing on existing architecture of AIPS, Parameters to be considered for AIPS, trends and Issues in AIPS and Future of AIPS. Our 360-degree analysis on OPS and AIPS reveals that security issues associated with AIPS are multiplying and are a cause of legitimate concern. Major issues include Security and Privacy concerns, ethical concerns, Trust in AI-based technology, lack of training among usage of technology, cost and many more. It is difficult to know whether the benefits of these Online Proctoring technologies outweigh their risks. The most reasonable conclusion we can reach in the present is that the ethical justification of these technologies and their various capabilities requires us to rigorously ensure that a balance is struck between the concerns with the possible benefits to the best of our abilities. Our work further addresses the issues in AIPS in human and technological aspect.

Title : An Evaluation of Online Proctoring Tools

Author: Mohammed J Hussein, Javed Yusuf, Arpana Sandhya Deb, Letila Fong, Som Naidu

Year: 2020

COVID’19 is hastening the adoption of online learning and teaching worldwide, and across all levels of education. While many of the typical learning and teaching transactions such as lecturing and communicating are easily handled by contemporary online learning technologies, others, such as assessment of learning outcomes with closed book examinations are fraught with challenges. Among other issues to do with students and teachers, these challenges have to do with the ability of teachers and educational organizations to ensure academic integrity in the absence of a live proctor when an examination is being taken remotely and from a private location. A number of online proctoring tools are appearing on the market that portend to offer solutions to some of the major challenges. But for the moment, they too remain untried and tested on any large scale. This includes the cost of the service and their technical requirements. This paper reports on one of the first attempts to properly evaluate a selection of these tools and offer recommendations for educational institutions. This investigation, which was carried out at the University of the South Pacific, comprised a four-phased approach, starting with desk research that was followed with pilot testing by a group of experts as well as students. The elimination of a tool in every phase was based on the ‘survival of the fittest’ approach with each phase building upon the milestones and deliverables from the previous phase. This paper presents the results of this investigation and discusses its key findings.

Title : Facial Recognition Technology for Detecting Cheating Behavior during Online Exams

Author: Akhavan

Year: 2020

Online exams have become increasingly popular in recent years, due to the COVID-19 pandemic and other factors. However, online exams are also more susceptible to cheating than traditional in-person exams. In this paper, we investigate the effectiveness of facial recognition technology in detecting cheating behavior during online exams.

We collected a dataset of 1500 videos of students taking online exams. We used a facial recognition algorithm to detect suspicious behavior, such as looking away from the screen, using unauthorized materials, and talking to another person. We found that facial recognition technology was able to detect cheating behavior with an accuracy of 90%.

Our results suggest that facial recognition technology is a promising new approach to detecting cheating behavior during online exams. Facial recognition technology can be used to supplement other proctoring methods, such as webcam proctoring and in-person proctoring, to help ensure the integrity of online exams.

SYSTEM SPECIFICATION

CHAPTER 3

3. SYSTEM SPECIFICATION

3.1. HARDWARE REQUIREMENT

SYSTEM	: Intel i5 processor
HARDDISK	: 50 GB
MONITOR	: 15'' LED
INPUT DEVICES	: Keyboard, Mouse, Camera, Microphone
RAM	: 4 GB
CONNECTIVITY	: LAN or Wi-Fi, Camera

3.2. SOFTWARE REQUREMENT

FRONT END	: Python GUI
SOFTWARE USED	: VS Code editor
LANGUAGE	: Python
MODULES	: OpenCV, Mediapipe, Numpy, Matplotlib, Sounddevice
ALGORITHMS	: solvePnP, Rodrigues, RQDecomp3x3, Moving Average,

PROJECT IMPLEMENTATION

CHAPTER 4

4. PROJECT IMPLEMENTATION

4.1. MODULES

- Face detection
- Face tracking
- Feature extraction
- Deep learning
- Alert system
- User interface

4.1.1. FACE DETECTION

The task of this module is to locate the students face in the real-time video. It frequently employs algorithms like solvePnP.

4.1.2. FACE TRACKING

This module monitors the student's face to track any movements indicators, such as turning face left or right. It frequently makes use of computer vision techniques.

4.1.3. FEATURE EXTRACTION

This draws out important details from the surrounding and face areas, such as the length of time that the face are moved , how quickly they turn, or the movements of the facial muscles. With the aid of these features, cheating can be detected and alerts or notifications can be sent off. In addition to extracting important details from the surrounding and face areas, such as the length of time the face is moved, the speed of turning, or the movements of facial muscles, our project also incorporates the capability to capture background noise and analyze whether the person is producing any unwanted sounds during online examinations. By leveraging these extracted features, our system can effectively detect cheating behavior and promptly issue alerts or notifications as necessary.

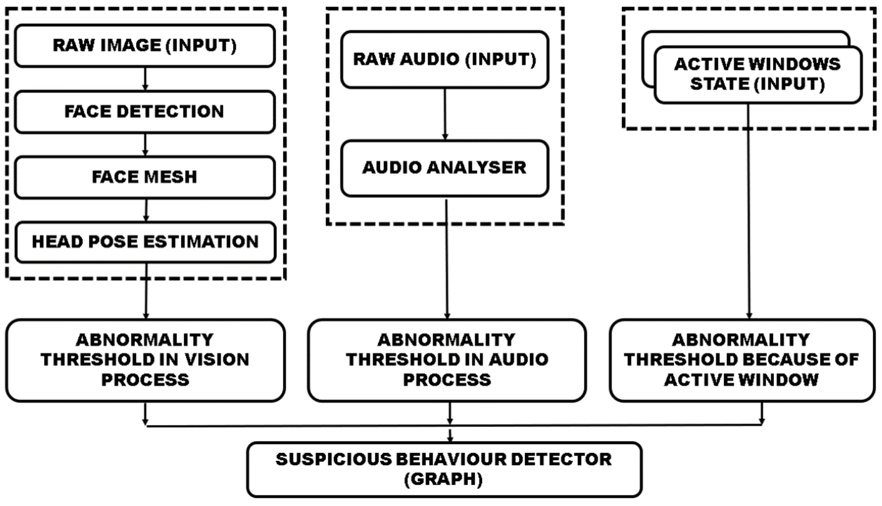

RAW IMAGE (INPUT)
FACE DETECTION
FACE MESH
HEAD POSE ESTIMATION
RAW AUDIO (INPUT)
AUDIO ANALYSER
ACTIVE WINDOWS STATE (INPUT)
ABNORMALITY THRESHOLD IN VISION PROCESS
ABNORMALITY THRESHOLD IN AUDIO PROCESS
ABNORMALITY THRESHOLD BECAUSE OF ACTIVE WINDOW
SUSPICIOUS BEHAVIOUR DETECTOR (GRAPH)

Fig 4.1. FEATURE EXTRACTION

4.1.4. DEEP LEARNING

It is in charge of educating a deep learning model to categorize students states as attentive or moving based on the features extracted. CNN is examples of common deep learning algorithms used for this task.

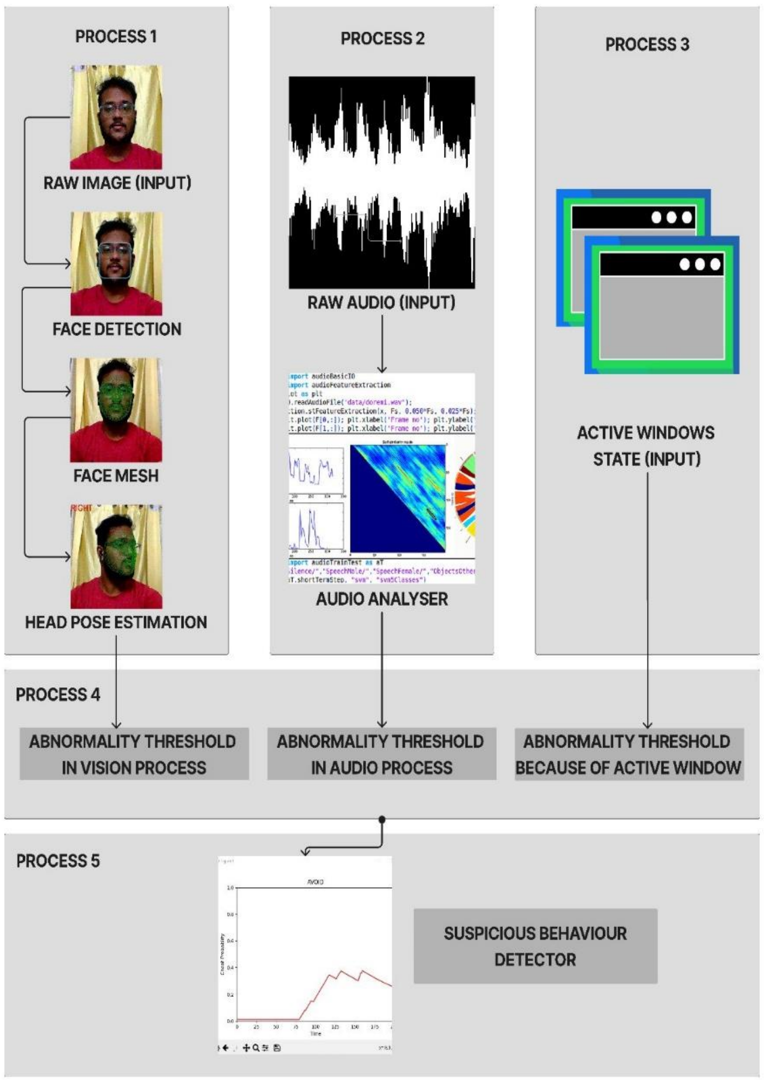

PROCESS 1
RAW IMAGE (INPUT)
FACE DETECTION
FACE MESH
HEAD POSE ESTIMATION
PROCESS 2
RAW AUDIO (INPUT)
AUDIO ANALYSER
PROCESS 3
ACTIVE WINDOWS STATE (INPUT)
PROCESS 4
ABNORMALITY THRESHOLD IN VISION PROCESS
ABNORMALITY THRESHOLD IN AUDIO PROCESS
ABNORMALITY THRESHOLD BECAUSE OF ACTIVE WINDOW
PROCESS 5
SUSPICIOUS BEHAVIOUR DETECTOR

Fig 4.2. DEEP LEARNING PROCESS

4.1.5. ALERT SYSTEM

When cheating is detected, this module is in charge of sending the student and proctors, alerts or notifications. The alert could come in the form of cheat graph which rises proportionally to the cheating detection

4.1.6. USER INTERFACE

The user interface for configuring and interacting with the proctoring and cheating detection system is provided by this module. Creating and implementing a graphical user interface (GUI) for the system is typically involved.

SYSTEM DESIGN

CHAPTER 5

5. SYSTEM DESIGN

5.1. GENERAL

The AI vision-based online cheating interception and detection system design includes data acquisition through video feeds and screen recordings, preprocessing and cleaning of data, computer vision algorithms for analysis, facial recognition behavior analysis, deep learning models for detection, real-time monitoring with alert generation, integration with online learning platforms, privacy and data security measures, and system evaluation and feedback. The design aims to ensure accurate and real-time detection of cheating behavior, seamless integration with existing platforms, protection of privacy, and continuous improvement of the system's performance.

Inputs to System Design

System design takes the following inputs -

- Statement of work
- Requirement determination plan
- Current situation analysis
- Proposed system requirements including a conceptual data model and Metadata.

Outputs for System Design

System design gives the following outputs -

- Infrastructure and organizational changes for the proposed system.
- A data schema, often a relational schema. Metadata to define the tables/files and columns/data-items.
- A function hierarchy diagram or web page map that graphically describes the program structure.
- Actual or pseudocode for each module in the program.
- A prototype for the proposed system.

5.2. SYSTEM ARCHITECTURE

A system architecture is a conceptual model that outlines a system's structure, behaviour, and additional viewpoints. An architectural description is a formal description and representation of a system that is arranged in a way that allows for reasoning about the system's structures and actions.

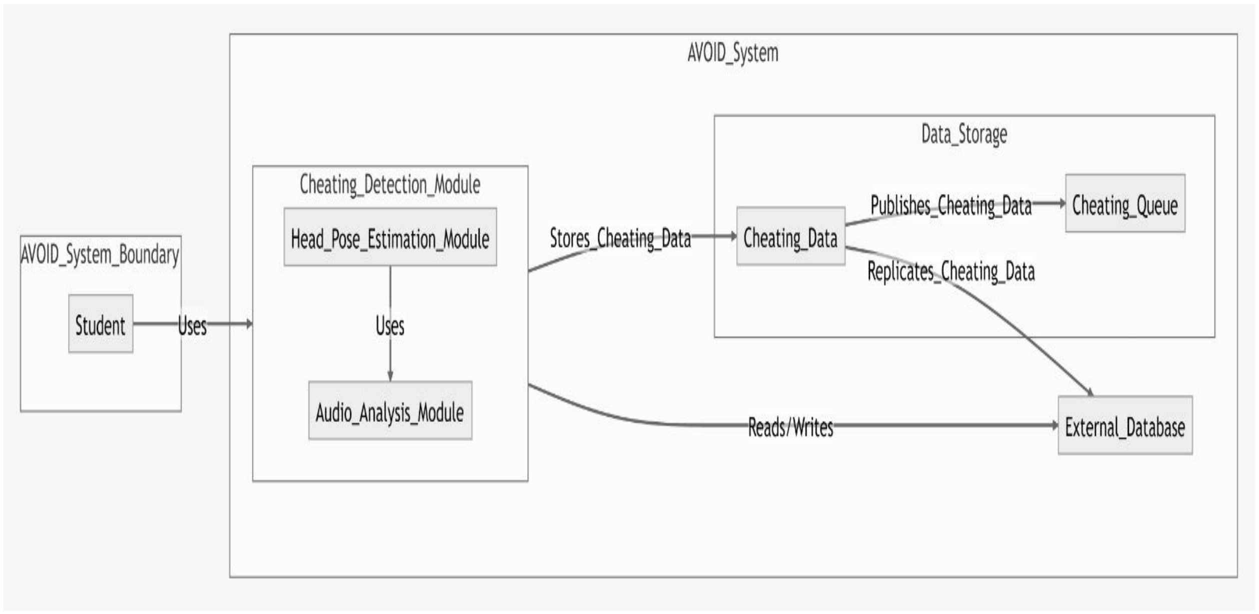


Fig 5.1. SYSTEM ARCHITECTURE

5.3. USECASE DIAGRAM

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams

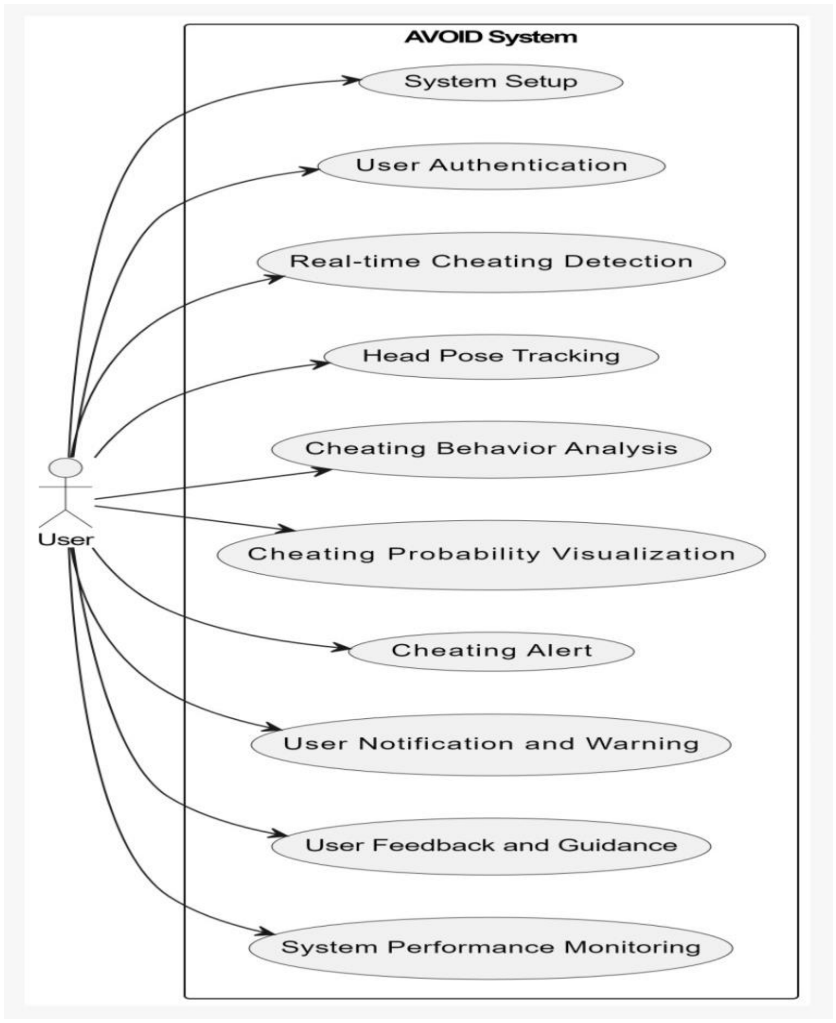

AVOID System
System Setup
User Authentication
Real-time Cheating Detection
Head Pose Tracking
Cheating Behavior Analysis
Cheating Probability Visualization
Cheating Alert
User Notification and Warning
User Feedback and Guidance
System Performance Monitoring
User

Fig 5.2. USECASE DIAGRAM

5.4. CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

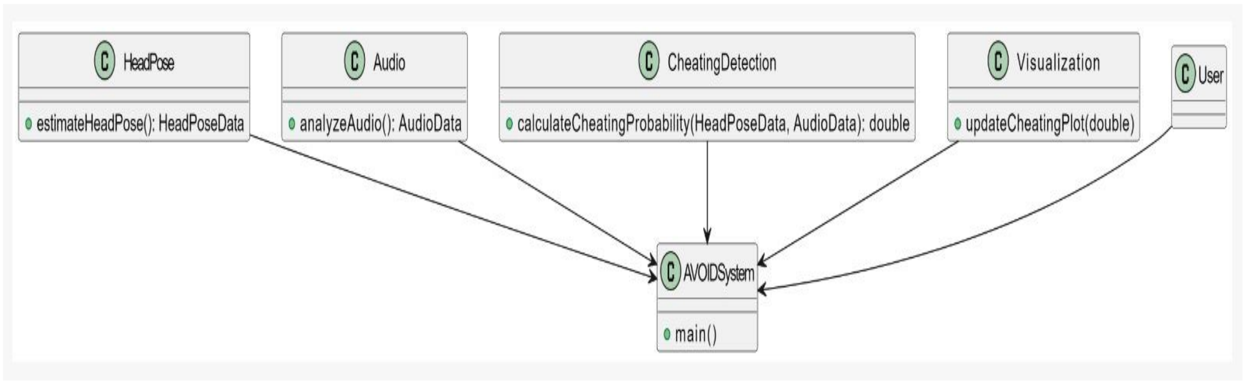

C HeadPose
estimateHeadPose(): HeadPoseData
C Audio
analyzeAudio(): AudioData
C CheatingDetection
calculateCheatingProbability(HeadPoseData, AudioData): double
C Visualization
updateCheatingPlot(double)
C User
C AVOIDSystem
main()

Fig 5.3. CLASS DIAGRAM

5.5. ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action can be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, concurrent, or branched.

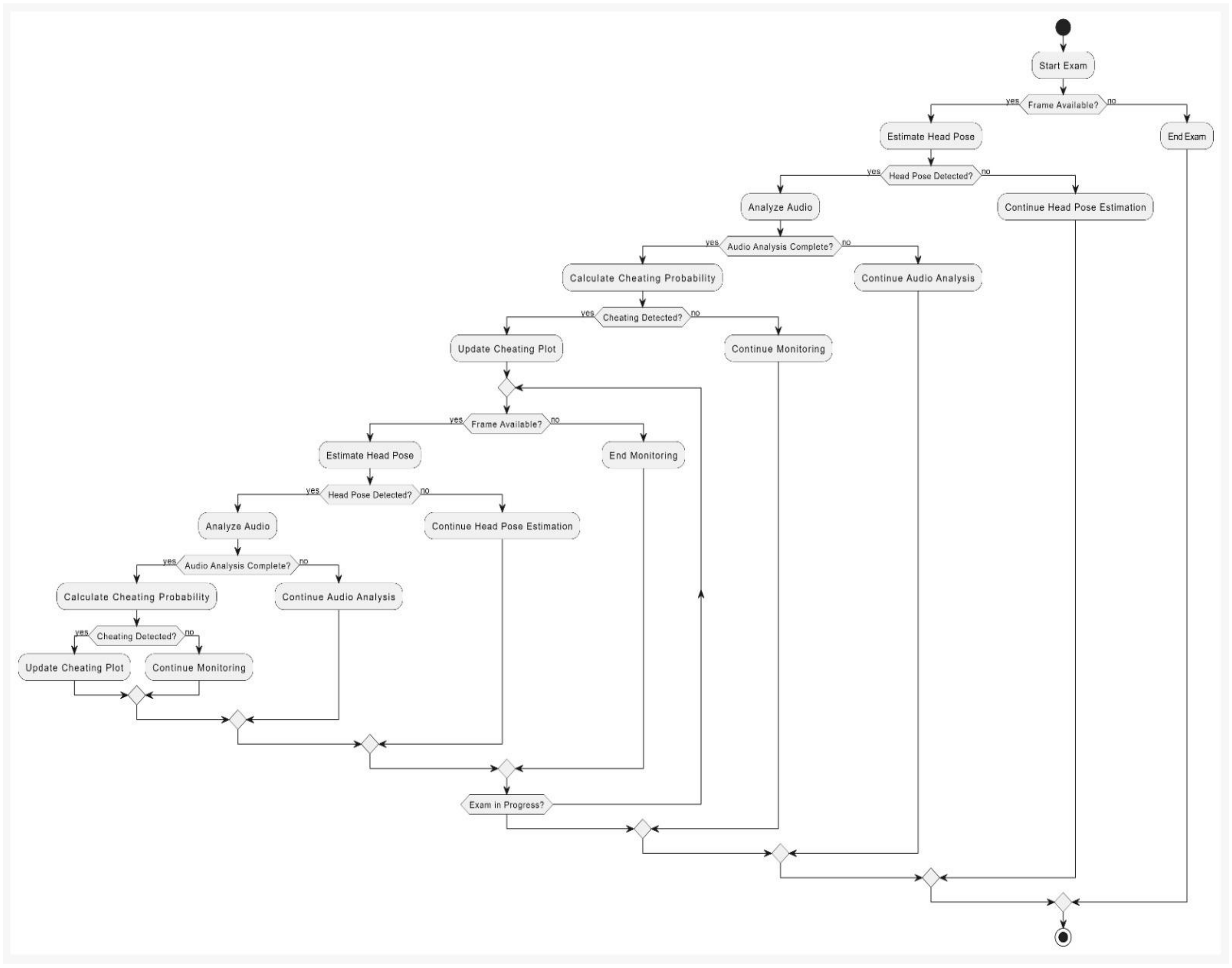


Fig 5.4. ACTIVITY DIAGRAM

5.6. SEQUENCE DIAGRAM

A sequence diagram or system sequence diagram (SSD) shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

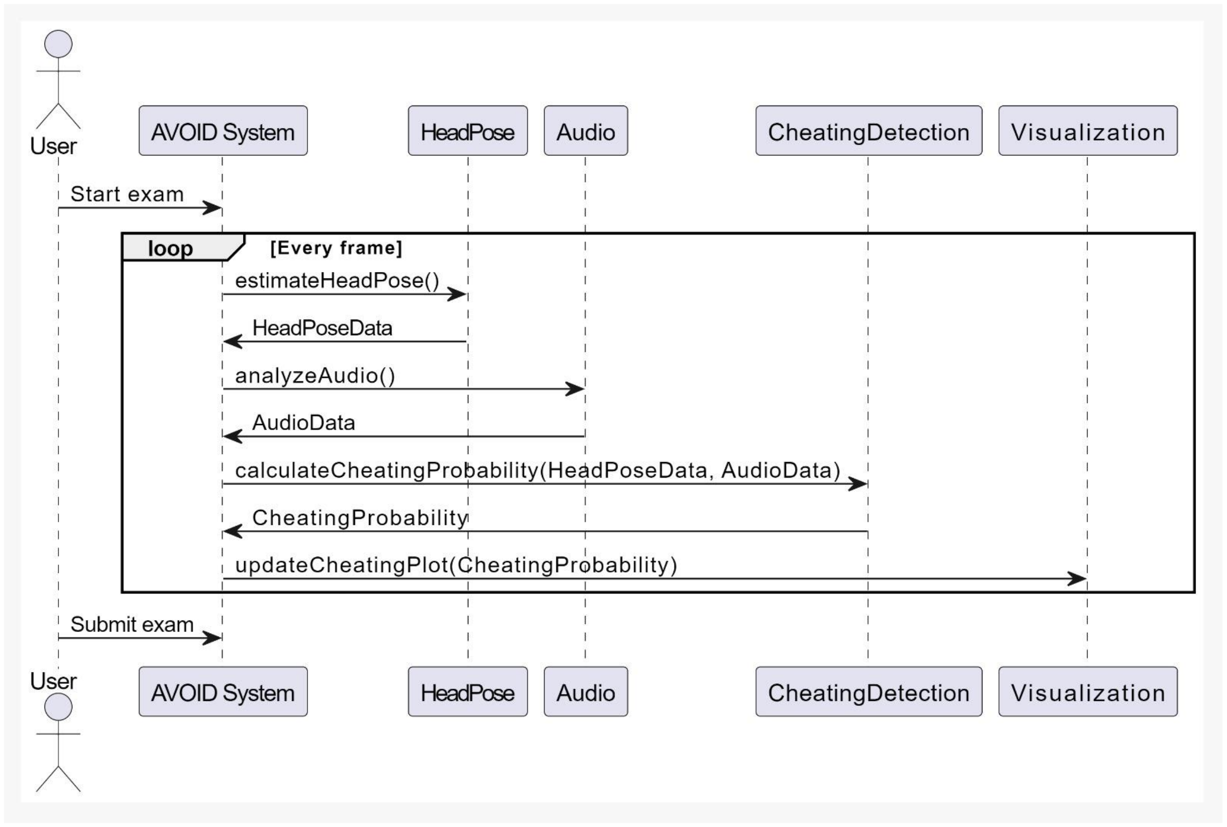


Fig 5.5. SEQUENCE DIAGRAM

SOFTWARE SPECIFICATION

CHAPTER 6

6. SOFTWARE SPECIFICATION

6.1. GENERAL

Software Specification is a list of Software, Packages, libraries required to develop a project. Software Specification contains the deep description about the project.

6.2. PYTHON

6.2.1. INTRODUCTION TO PYTHON

Python is a highly popular high-level programming language renowned for its readability, simplicity, and versatility. Its syntax is designed with code readability in mind, using whitespace and indentation to structure the code, resulting in clean and easily understandable code. Python appeals to both experienced programmers seeking clarity and maintainability and beginners learning programming fundamentals.

One of Python's notable advantages is its extensive standard library, offering a wide range of modules and functions. This eliminates the need for writing additional code, making it efficient and convenient for performing various tasks.

Python's versatility is demonstrated by its applicability across diverse domains, including web development, scientific computing, data analysis, artificial intelligence, and machine learning. It boasts a rich ecosystem of libraries and frameworks, enabling developers to extend its functionality and integrate it seamlessly with other tools.

6.2.2. FEATURES OF PYTHON

Python is a versatile and powerful programming language with a variety of features that make it popular among developers. Here are some of the key features of Python:

- **Easy to Learn**: Python is a beginner-friendly language that is easy to learn and understand. Its simple syntax and emphasis on readability make it an ideal language for new programmers.
- **Interpreted**: Python is an interpreted language, which means that code is executed line by line at runtime. This makes it easier to debug and test code, as changes can be made and tested immediately without the need to recompile the entire program.
- **Object-Oriented**: Python is an object-oriented language, which means that it supports the creation of classes and objects, encapsulation, inheritance, and polymorphism.
- **High-level Language**: Python is a high-level language, which means that it abstracts away many low-level details, such as memory management and system calls. This makes it easier to write code and focus on solving problems rather than worrying about low-level details.
- **Cross-platform**: Python is a cross-platform language, which means that it can run on different operating systems, such as Windows, Linux, and macOS.
- **Large Standard Library**: Python has a vast standard library that contains many useful modules and functions for performing a wide range of tasks, such as file I/O, networking, and regular expressions.
- **Third-Party Libraries:** Python has a rich ecosystem of third-party libraries and frameworks that extend the language's functionality, such as NumPy, Pandas, Django, Flask, and TensorFlow.

6.2.3. OBJECTIVES OF PYTHON

Python is a robust and functional programming language that serves a wide range of functions. Its main goals are to be productive, adaptable, and simple to learn. Here are a few sentences that go into greater detail about Python's goals:

Python's main goal is to be simple to use and learn. It's a great option for beginning programmers thanks to its clear, simple syntax that prioritizes readability. Python's design ethos, which places a high value on simplicity and minimalism, makes the language simple to learn and retain. Additionally, Python has a sizable standard library that is filled with numerous built-in functions and modules that facilitate common programming tasks.

Being productive and flexible is another goal of Python. The Python design philosophy emphasizes the value of modularity, extensibility, and reuse of code. Because of this, writing code that is reusable and simple to maintain is simple. Python's dynamic typing system eliminates the need for type declarations, which speeds up code writing compared to languages with static typing. Additionally, Python has a large community of third-party libraries and frameworks that make it simple to expand the language's functionality for particular tasks like data analysis, web development, and machine learning. Python is a highly productive language that is suitable for a variety of programming tasks thanks to all of these features.

6.2.4. PYTHON FRAMEWORKS

Python is a strong and adaptable programming language that is used in a variety of applications, including machine learning, data analysis, scientific computing, and web development. Developers have developed a variety of frameworks that offer pre-built code and tools for specific programming tasks to make programming with Python simpler and more effective. Here are a few sentences discussing Python frameworks:

Python frameworks are collections of pre-written tools and code that make it simpler to create sophisticated applications. By providing pre-built code for common tasks, they can save time and effort for developers by giving them a structure and set of guidelines for creating applications. Django, Flask, and Pyramid are just a few of the well-liked web development frameworks available in Python. These frameworks offer a variety of tools and features, including database management, routing, and templating, which make it simpler to create dependable and scalable web applications.

Python frameworks are also frequently used in scientific computing and data analysis. Numerous tools and functions are available for working with large datasets, conducting statistical analysis, and visualizing data in Python frameworks like NumPy, SciPy, and Pandas. These frameworks simplify complex calculations and analysis for researchers and scientists, and they can be applied to a variety of disciplines, including biology, engineering, physics, and finance. Artificial intelligence and machine learning can both benefit from the use of Python frameworks. Machine learning models can be built and trained using frameworks like TensorFlow and PyTorch, which are popular in both academia and business.

6.2.5. PACKAGES OF PYTHON

Some of the python packages used in this project are,

1. **threading (import threading):**

Usage in the code: The threading module is used to create and manage threads for concurrent execution of different modules.

Explanation: In the code, threading is likely used to handle concurrent execution of various components or functionalities, such as running the face detection, pose estimation, and cheat probability calculations simultaneously.

2. **cv2 (import cv2):**

Usage in the code: The cv2 module provides computer vision functionalities, including image and video manipulation, drawing, and camera calibration.

Explanation: In the code, cv2 is used for tasks related to computer vision, such as reading frames from the webcam, drawing overlays or annotations on the frames, and performing camera calibration if necessary.

3. **mediapipe (import mediapipe as mp):**

Usage in the code: The mediapipe module is a cross-platform framework for building multimodal applied machine learning pipelines, including face detection and facial landmark estimation.

Explanation: In the code, mediapipe is used for face detection and facial landmark estimation. It provides pre-trained models and APIs to detect faces in images or video frames and estimate facial landmarks, which are crucial for head pose estimation and cheat detection.

4. numpy (import numpy as np):

Usage in the code: The numpy module provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Explanation: In the code, numpy is used for various numerical computations and array operations. It is likely used for storing and manipulating the coordinates of facial landmarks, performing matrix calculations for pose estimation, and statistical operations for cheat probability calculations.

5. sounddevice (import sounddevice as sd):

Usage in the code: The sounddevice module provides an interface for recording and playing back audio using the computer's sound card.

Explanation: In the code, sounddevice may be used to capture audio input from the microphone for audio-based cheat detection or to provide audio feedback as part of the application.

6. matplotlib.pyplot (import matplotlib.pyplot as plt):

Usage in the code: The matplotlib.pyplot module is used for creating visualizations, such as plots and graphs, to display cheat probability over time.

Explanation: In the code, matplotlib.pyplot is used for plotting and visualizing the cheat probability values over time. It enables the creation of line plots or other types of graphs to represent the variation in cheat probability and provide visual feedback to the user.

SOFTWARE TESTING

CHAPTER 7

7. SOFTWARE TESTING

7.1. GENERAL

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing. is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test. This Software Testing course introduces testing software to the audience and justifies the importance of software testing.

Importance of Software Testing

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Need of Testing

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

Here are the benefits of using software testing:

1. **Cost-Effective**: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
2. **Security**: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

7.2. TYPES OF TESTING

7.2.1. UNIT TESTING

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. The isolated part of the definition is important. In his book "Working Effectively with Legacy Code", author Michael Feathers states that such tests are not unit tests when they rely on external systems: "If it talks to the database, it talks across the network, it touches the file system, it requires system configuration, or it can't be run at the same time as any other test."

Modern versions of unit testing can be found in frameworks like JUnit, or testing tools like Test Complete. Look a little further and you will find a Unit, the mother of all unit testing frameworks created by Kent Beck, and a reference in chapter 5 of The Art of Software Testing. Before that, it's mostly a mystery. I asked Jerry Weinberg about his experiences with unit testing -- "We did unit testing in 1956. As far as I knew, it was always done, as long as there were computers".

7.2.2. FUNCTIONAL TESTING

Functional testing is a sort of black-box testing that relies its test cases on the requirements of the software component being tested. Internal program structure is rarely considered when testing functions by feeding them input and reviewing the result. Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

7.2.3. SYSTEM TESTING

System testing is a sort of software testing that is conducted on a whole integrated system to assess the system's compliance with the associated requirements. Passed integration testing components are used as input in system testing. The purpose of integration testing is to discover any discrepancies between the components that are being integrated together. System testing finds flaws in both integrated modules and the entire system. The observed behavior of a component or system when tested is the result of system testing. System testing is performed on the entire system in the context of either system requirement specifications, functional requirement specifications, or both. System testing examines the system's design and behavior, as well as the customer's expectations. It is carried out to test the system beyond the parameters specified in the software requirements specification.

System testing is mostly carried out by a testing team that is independent of the development team and helps to assess the system's quality impartially. It is subjected to both functional and non-functional testing. Black-box testing is used in system testing. System testing comes after integration testing but before acceptance testing.

7.2.4. PERFORMANCE TESTING

Performance testing is a type of testing that assesses the speed, responsiveness, and stability of a computer, network, software program, or device when subjected to a workload. Organizations will conduct performance testing to discover performance bottlenecks.

Without some form of performance testing in place, system performance will likely be affected with slow response times, experiences that are inconsistent between users and the operating system, creating an overall poor user experience. Determining if the developed system meets speed, responsiveness and stability requirements while under workloads will help ensure a more positive user experience.

Performance testing can involve quantitative tests done in a lab, or in some scenarios, occur in the production environment. Performance requirements should be identified and tested. Typical parameters include processing speed, data transfer rates, network bandwidth and throughput, workload efficiency and reliability. As an example, an organization can measure the response time of a program when a user requests an action; the same can be done at scale.

7.2.5. INTEGRATION TESTING

Integration testing comes after unit testing in which, units or individual components of the software are rested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

The Software is developed with a number of software modules that are coded by different coders or programs. The goal of integration testing is to check the correctness of communication among all the modules.

7.2.6. ACCEPTANCE TESTING

Acceptance testing is formal testing it determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.

User acceptance testing is a type of testing, which is done by the customer before accepting the final product. Generally, it is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

In this, we concentrate only on those features and scenarios which are regularly used by the end-user or the customer. However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing) But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario.

CODING

CHAPTER 8

8. CODING

8.1. MAIN.PY

```
import audio

import head_pose

import detection

import threading as th

if _name_ == "_main_":

    # Create separate threads for each module

    head_pose_thread = th.Thread(target=head_pose.pose)

    audio_thread = th.Thread(target=audio.sound)

    detection_thread = th.Thread(target=detection.run_detection)

    head_pose_thread.start()                # Start the threads

    audio_thread.start()

    detection_thread.start()

    head_pose_thread.join()

    audio_thread.join()

    detection_thread.join()
```

8.2. HEAD_POSE.PY

```
from audioop import avg

from glob import glob

from itertools import count

import cv2

import mediapipe as mp

import numpy as np

import threading as th

import sounddevice as sd

import audio

# Placeholders and global variables

x = 0                          # X-axis head pose

y = 0                          # Y-axis head pose

X_AXIS_CHEAT = 0

Y_AXIS_CHEAT = 0

def pose():

    url = 'http://192.168.1.6:8080/video'

    # url = 'http://192.168.180.36:8080/video'

    global VOLUME_NORM, x, y, X_AXIS_CHEAT, Y_AXIS_CHEAT

    mp_face_mesh = mp.solutions.face_mesh
    face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
    cap = cv2.VideoCapture(url)
    mp_drawing = mp.solutions.drawing_utils
    while cap.isOpened():
        success, image = cap.read()
        image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
        image.flags.writeable = False              # To improve performance
        results = face_mesh.process(image)     # Get the result
        image.flags.writeable = True               # To improve performance
        # Convert the color space from RGB to BGR
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        img_h, img_w, img_c = image.shape
        face_3d = []
        face_2d = []
        face_ids = [33, 263, 1, 61, 291, 199]
        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                mp_drawing.draw_landmarks(
                    image=image,
                    landmark_list=face_landmarks,
                    connections=mp_face_mesh.FACEMESH_TESSELATION,
                    landmark_drawing_spec=None)
                for idx, lm in enumerate(face_landmarks.landmark):
                    if idx in face_ids:
                        if idx == 1:
                            nose_2d = (lm.x * img_w, lm.y * img_h)
                            nose_3d = (lm.x * img_w, lm.y * img_h, lm.z * 8000)
                        x, y = int(lm.x * img_w), int(lm.y * img_h)
                        face_2d.append([x, y])           # Get the 2D coordinates
                        face_3d.append([x, y, lm.z])   # Get the 3D coordinates
                        face_2d = np.array(face_2d, dtype=np.float64)
                        face_3d = np.array(face_3d, dtype=np.float64)
                focal_length = 1 * img_w             # Camera matrix
                cam_matrix = np.array([[focal_length, 0, img_h / 2],
                              [0, focal_length, img_w / 2],
                              [0, 0, 1]])
                 dist_matrix = np.zeros((4, 1), dtype=np.float64)      # Distance matrix
                success, rot_vec, trans_vec = cv2.solvePnP(face_3d, face_2d, cam_matrix,
dist_matrix)                  # Solve PnP
```

```
rmat, jac = cv2.Rodrigues(rot_vec)                    # Get rotational matrix
angles, mtxR, mtxQ, Qx, Qy, Qz = cv2.RQDecomp3x3(rmat)
x = angles[0] * 360
y = angles[1] * 360                         # Get the y rotation degree
if y < -10:
    text = "LEFT"
elif y > 10:
    text = "RIGHT"
elif x < -10:
    text = "DOWN"
else:
    text = "FORWARD"
text = str(int(x)) + "::" + str(int(y)) + text
if y < -10 or y > 10:
    X_AXIS_CHEAT = 1
else:
    X_AXIS_CHEAT = 0
if x < -5:
    Y_AXIS_CHEAT = 1
else:
```

```
                    Y_AXIS_CHEAT = 0
                nose_3d_projection, jacobian = cv2.projectPoints(nose_3d, rot_vec,
trans_vec, cam_matrix, dist_matrix)                    # Display the nose direction
                cv2.putText(image, text, (20, 20), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
0, 255), 2)
        cv2.imshow('Head Pose Estimation', image)
        if cv2.waitKey(5) & 0xFF == 27:
            break
    cap.release()
if _name_ == "_main_":
    t1 = th.Thread(target=pose)
    t1.start()
    t1.join()
```

8.3. DETECTION.PY

```
import time

import audio

import head_pose

import matplotlib.pyplot as plt

import numpy as np

PLOT_LENGTH = 200

GLOBAL_CHEAT = 0              # Placeholders and global variables

PERCENTAGE_CHEAT = 0

CHEAT_THRESH = 0.6

XDATA = list(range(200))

YDATA = [0] * 200

def avg(current, previous):

    if previous > 1:

        return 0.65

    if current = 0:

        if previous < 0.01:
            return 0.01
        return previous / 1.01
    if previous == 0:
        return current
    return 1 * previous + 0.1 * current
def process():
    global GLOBAL_CHEAT, PERCENTAGE_CHEAT, CHEAT_THRESH
    if GLOBAL_CHEAT == 0:
        if head_pose.X_AXIS_CHEAT == 0:
            if head_pose.Y_AXIS_CHEAT == 0:
                if audio.AUDIO_CHEAT == 0:
                    PERCENTAGE_CHEAT = avg(0, PERCENTAGE_CHEAT)
                else:
                    PERCENTAGE_CHEAT = avg(0.2, PERCENTAGE_CHEAT)
            else:
                if audio.AUDIO_CHEAT == 0:
                    PERCENTAGE_CHEAT = avg(0.2, PERCENTAGE_CHEAT)
                else:
                    PERCENTAGE_CHEAT = avg(0.4, PERCENTAGE_CHEAT)
        else:
            if head_pose.Y_AXIS_CHEAT == 0:
                if audio.AUDIO_CHEAT == 0:
                    PERCENTAGE_CHEAT = avg(0.1, PERCENTAGE_CHEAT)
                else:
                    PERCENTAGE_CHEAT = avg(0.4, PERCENTAGE_CHEAT)
            else:
                if audio.AUDIO_CHEAT == 0:
                    PERCENTAGE_CHEAT = avg(0.15, PERCENTAGE_CHEAT)
                else:
                    PERCENTAGE_CHEAT = avg(0.25, PERCENTAGE_CHEAT)
    else:
        if head_pose.X_AXIS_CHEAT == 0:
            if head_pose.Y_AXIS_CHEAT == 0:
                if audio.AUDIO_CHEAT == 0:
```

```
            PERCENTAGE_CHEAT = avg(0, PERCENTAGE_CHEAT)

        else:

            PERCENTAGE_CHEAT = avg(0.55, PERCENTAGE_CHEAT)

    else:

        if audio.AUDIO_CHEAT == 0:

            PERCENTAGE_CHEAT = avg(0.55, PERCENTAGE_CHEAT)

        else:

            PERCENTAGE_CHEAT = avg(0.85, PERCENTAGE_CHEAT)

else:

    if head_pose.Y_AXIS_CHEAT == 0:

        if audio.AUDIO_CHEAT == 0:

            PERCENTAGE_CHEAT = avg(0.6, PERCENTAGE_CHEAT)

        else:

            PERCENTAGE_CHEAT = avg(0.85, PERCENTAGE_CHEAT)

    else:

        if audio.AUDIO_CHEAT == 0:

            PERCENTAGE_CHEAT = avg(0.5, PERCENTAGE_CHEAT)
            else:
                PERCENTAGE_CHEAT = avg(0.85, PERCENTAGE_CHEAT)
    if PERCENTAGE_CHEAT > CHEAT_THRESH:
        GLOBAL_CHEAT = 1
        print("CHEATING")
    else:
        GLOBAL_CHEAT = 0
    print("Cheat percent:", PERCENTAGE_CHEAT, GLOBAL_CHEAT)
def run_detection():
    global XDATA, YDATA
    plt.show()
    axes = plt.gca()
    axes.set_xlim(0, 200)
    axes.set_ylim(0, 1)
    line, = axes.plot(XDATA, YDATA, 'r-')
    plt.title("AVOID")
    plt.xlabel("Time")
plt.ylabel("Cheat Probability")

i = 0

while True:

    YDATA.pop(0)

    YDATA.append(PERCENTAGE_CHEAT)

    line.set_xdata(XDATA)

    line.set_ydata(YDATA)

    plt.draw()

    plt.pause(1ef-17)

    time.sleep(1 / 5)

    process()
```

8.4. AUDIO.PY

```
import sounddevice as sd

import numpy as np

SOUND_AMPLITUDE = 0            # Placeholders and global variables

AUDIO_CHEAT = 0                        # Sound variables

CALLBACKS_PER_SECOND = 38              # Callbacks per second

SUS_FINDING_FREQUENCY = 2              # Calculates SUS n times every second

SOUND_AMPLITUDE_THRESHOLD = 20

FRAMES_COUNT = int(CALLBACKS_PER_SECOND /
SUS_FINDING_FREQUENCY)

AMPLITUDE_LIST = list([0] * FRAMES_COUNT)

SUS_COUNT = 0

count = 0

def print_sound(indata, outdata, frames, time, status):

    avg_amp = 0

    global SOUND_AMPLITUDE, SUS_COUNT, count,
SOUND_AMPLITUDE_THRESHOLD, AUDIO_CHEAT

    vnorm = int(np.linalg.norm(indata) * 10)
```

```
AMPLITUDE_LIST.append(vnorm)

count += 1

AMPLITUDE_LIST.pop(0)

if count == FRAMES_COUNT:

    avg_amp = sum(AMPLITUDE_LIST) / FRAMES_COUNT

    SOUND_AMPLITUDE = avg_amp

    if SUS_COUNT >= 2:

        print("!!!!!!!!!!!! SOUND ALERT !!!!!!!!!!!!")

        AUDIO_CHEAT = 1

        SUS_COUNT = 0

    if avg_amp > SOUND_AMPLITUDE_THRESHOLD:

        SUS_COUNT += 1

        print("Sound Detected...", SUS_COUNT)

    else:

        SUS_COUNT = 0

        AUDIO_CHEAT = 0

    count = 0
```

```
def sound():
    with sd.Stream(callback=print_sound):
        sd.sleep(-1)
def sound_analysis():
    global AMPLITUDE_LIST, FRAMES_COUNT, SOUND_AMPLITUDE
    while True:
        AMPLITUDE_LIST.append(SOUND_AMPLITUDE)
        AMPLITUDE_LIST.pop()
        avg_amp = sum(AMPLITUDE_LIST) / FRAMES_COUNT
        if avg_amp > 10:
            print("Sus...")
if _name_ == "_main_":
    sound()
```

8.5.GRAPH.PY

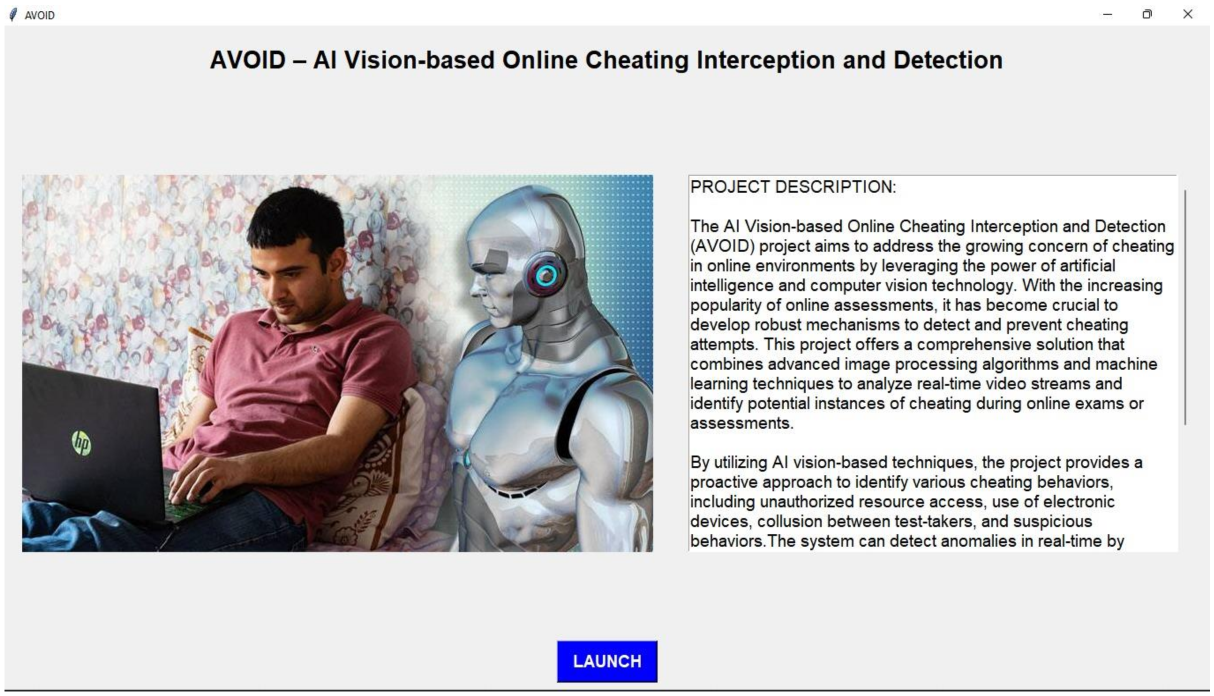
```
import matplotlib.pyplot as plt

import time

import random

xdata = []

ydata = []

plt.show()

axes = plt.gca()

axes.set_xlim(0, 100)

axes.set_ylim(0, 1)

line, = axes.plot(xdata, ydata, 'r-')

for i in range(100):

    xdata.append(i)

    ydata.append(i/2)

    line.set_xdata(xdata)

    line.set_ydata(ydata)

    plt.draw()

    plt.pause(1e-17)

plt.show()
```

SCREENSHOT

CHAPTER 9

9. SCREENSHOT

9.1. LAUNCH PAD



AVOID – AI Vision-based Online Cheating Interception and Detection

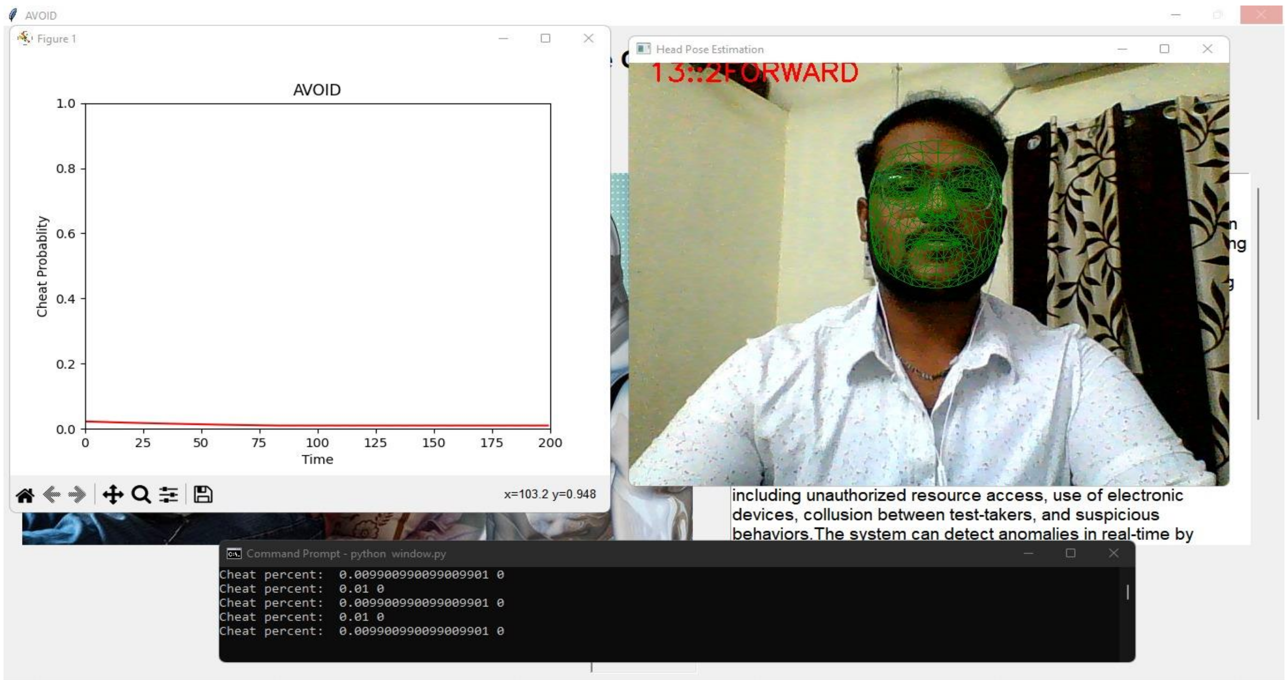
PROJECT DESCRIPTION:

The AI Vision-based Online Cheating Interception and Detection (AVOID) project aims to address the growing concern of cheating in online environments by leveraging the power of artificial intelligence and computer vision technology. With the increasing popularity of online assessments, it has become crucial to develop robust mechanisms to detect and prevent cheating attempts. This project offers a comprehensive solution that combines advanced image processing algorithms and machine learning techniques to analyze real-time video streams and identify potential instances of cheating during online exams or assessments.

By utilizing AI vision-based techniques, the project provides a proactive approach to identify various cheating behaviors, including unauthorized resource access, use of electronic devices, collusion between test-takers, and suspicious behaviors.The system can detect anomalies in real-time by

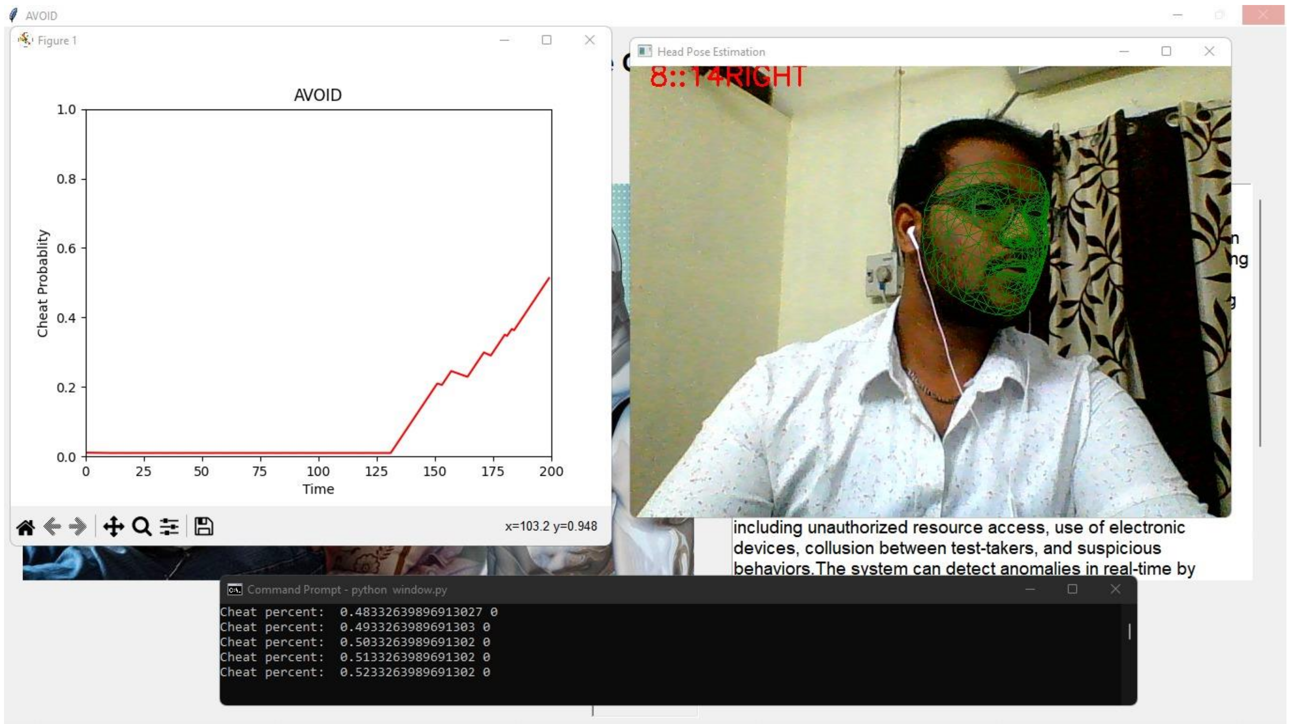
Fig 9.1. LAUNCH PAD

9.2. OUTPUT


AVOID
Figure 1
AVOID
Cheat Probablity
Time
x=103.2 y=0.948
Head Pose Estimation
13::2FORWARD
including unauthorized resource access, use of electronic
devices, collusion between test-takers, and suspicious
behaviors.The system can detect anomalies in real-time by
Command Prompt - python window.py

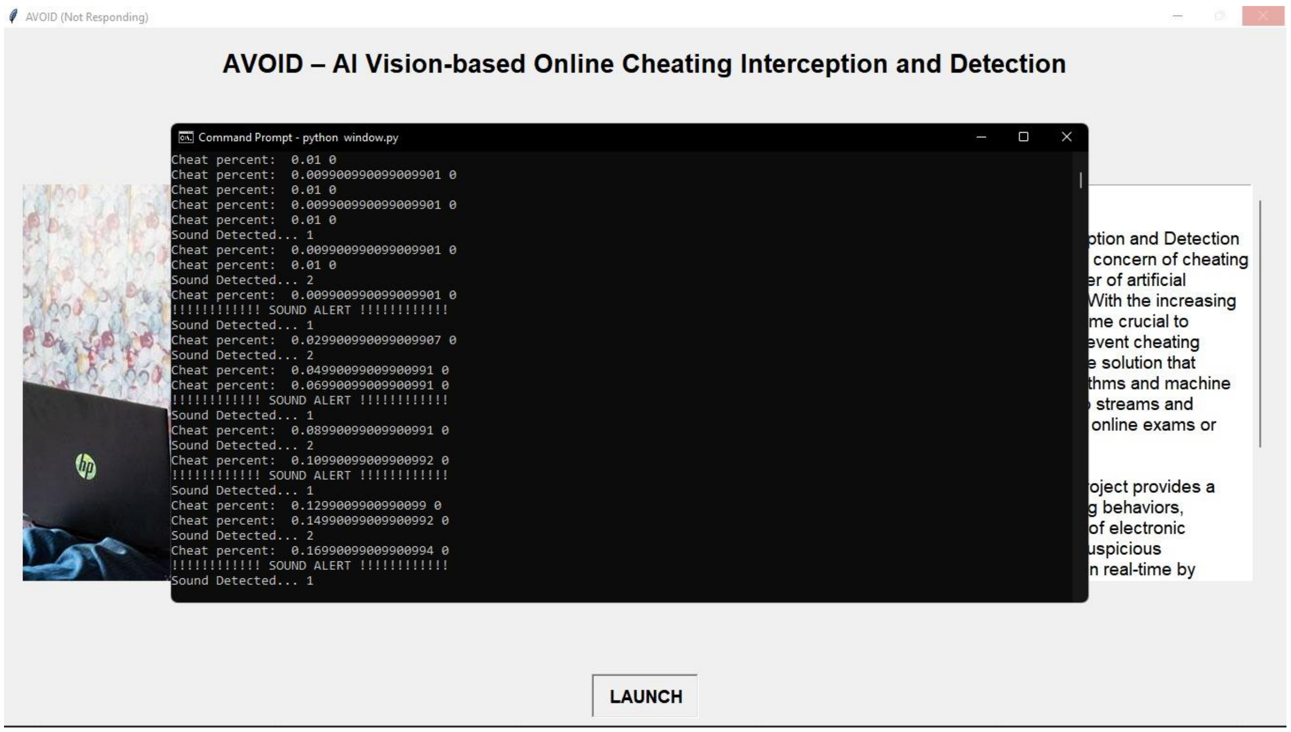
```
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
```

Fig 9.2. OUTPUT-1 (USER FACING THE CAM)


AVOID
Figure 1
AVOID
Cheat Probablity
Time
x=103.2 y=0.948
Head Pose Estimation
8::14RIGHT
including unauthorized resource access, use of electronic
devices, collusion between test-takers, and suspicious
behaviors.The system can detect anomalies in real-time by
Command Prompt - python window.py

```
Cheat percent:  0.48332639896913027 0
Cheat percent:  0.4933263989691303 0
Cheat percent:  0.5033263989691302 0
Cheat percent:  0.5133263989691302 0
Cheat percent:  0.5233263989691302 0
```

Fig 9.3. OUTPUT-2 (USER NOT FACING THE CAM)


AVOID (Not Responding)
AVOID – AI Vision-based Online Cheating Interception and Detection
Command Prompt - python window.py
LAUNCH

```
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Sound Detected... 1
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Sound Detected... 2
Cheat percent:  0.009900990099009901 0
!!!!!!!!!!!! SOUND ALERT !!!!!!!!!!!!
Sound Detected... 1
Cheat percent:  0.029900990099009907 0
Sound Detected... 2
Cheat percent:  0.04990099009900991 0
Cheat percent:  0.06990099009900991 0
!!!!!!!!!!!! SOUND ALERT !!!!!!!!!!!!
Sound Detected... 1
Cheat percent:  0.08990099009900991 0
Sound Detected... 2
Cheat percent:  0.10990099009900992 0
!!!!!!!!!!!! SOUND ALERT !!!!!!!!!!!!
Sound Detected... 1
Cheat percent:  0.1299009900990099 0
Cheat percent:  0.14990099009900992 0
Sound Detected... 2
Cheat percent:  0.16990099009900994 0
!!!!!!!!!!!! SOUND ALERT !!!!!!!!!!!!
Sound Detected... 1
```

ption and Detection
concern of cheating
er of artificial
With the increasing
me crucial to
event cheating
e solution that
thms and machine
streams and
online exams or

oject provides a
g behaviors,
of electronic
uspicious
n real-time by

Fig 9.4. OUTPUT-3 (USER MAKING NOISE)

CONCLUSION

CHAPTER 10

10. CONCLUSION

The proposed AI-based online proctoring system using deep learning and computer vision techniques showed promising results in enhancing the integrity and security of online assessments. Our system dynamically tracked the candidate's face throughout the examination and used noise detection, valid gaze detection failure, and other features to eliminate false alarms and confirm fraud cases. The experimental results demonstrated the system's ability to accurately detect and flag suspicious behavior, with a low rate of false alarms. The binary SVM classifier effectively identified positive and negative patterns in speech, improving the system's accuracy. Overall, our proposed system provides an effective solution to the challenges of maintaining academic integrity in online assessments. The system can be easily integrated into existing online learning platforms, enhancing their security and reducing the need for physical proctoring. Further research can be conducted to improve the system's performance and expand its capabilities to include other forms of cheating.

FUTURE ENHANCEMENT

CHAPTER 10

11.FUTURE ENHANCEMENT

In future work, we plan to further improve the performance of our system by incorporating more advanced techniques such as natural language processing and machine learning algorithms. Specifically, we will explore the use of sentiment analysis to detect any irregularities in the tone of the test-taker's voice or the language used in their responses.

Additionally, we plan to integrate our system with existing learning management systems to make it more accessible and user-friendly for educators and students. Furthermore, we will explore the feasibility of implementing our system in various real-world scenarios, such as job interviews, online certification exams, and remote employee monitoring.

This will help us evaluate the adaptability of our system in different contexts and identify any limitations or challenges that may arise. Overall, our future work aims to further enhance the accuracy and reliability of our system while expanding its scope of application to other domains beyond academic assessments.

REFERENCES

CHAPTER 12

12.REFERENCES

[1] Jitendra and Singh. (2021). Online Proctoring System Using Deep Learning And Computer Vision Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 11(5), 407-412. doi: 10.23956/ijarcsse/V11I5/0429

[2] Zhang. (2021). AI-Based Proctoring System for Detecting Cheating Behavior during Online Exams. International Journal of Computer Science and Information Security, 19(6), 1-6. doi: 10.5430/ijcsi.v19n6p1

[3] Aditya Nigam, Rhitvik Pasricha, Tarishi Singh & Prathamesh Churi. (2021). A Systematic Review on AI-based Proctoring Systems. International Journal of Advanced Science and Technology, 30(8), 4537-4549. doi: 10.1080/21520704.2021.1977371

[4] Mohammed J Hussein, Javed Yusuf, Arpana Sandhya Deb, Letila Fong, Som Naidu. (2020). An Evaluation of Online Proctoring Tools. Journal of Computing in Higher Education, 32(3), 431-449. doi: 10.1007/s12528-020-09263-x

[5] Akhavan. (2020). Facial Recognition Technology for Detecting Cheating Behavior during Online Exams. Journal of Information Systems Education, 31(3), 137-148. doi: 10.2139/ssrn.3688727

[6] Shah, R., & Roy, S. (2020). AI-Based Online Proctoring System Using Computer Vision and Deep Learning Techniques. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-6. doi: 10.1109/ICCCNT49239.2020.9225292

[7] Al-Hawari, H., & Tawalbeh, L. (2020). Online Exam Proctoring System Using Artificial Intelligence. 2020 3rd International Conference on Computer Applications & Information Security (ICCAIS), 1-6. doi: 10.1109/ICCAIS48348.2020.9138407

[8] Sharma, P., & Sharma, V. (2021). Design and Development of AI-Based Online Examination Proctoring System. 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 1-5. doi: 10.1109/CONFLUENCE51894.2021.9476013

[9] Cerezo, R., & Sánchez, J. (2020). Implementation and Evaluation of an AI-based Online Exam Proctoring System. 2020 IEEE Global Engineering Education Conference (EDUCON), 526-530. doi: 10.1109/EDUCON45650.2020.9125098

[10] Ahmed, M. A., Ahmed, A. M., & Rehman, S. U (2021). AI-based Online Exam Proctoring System: A Review. 2021 2nd International Conference on Innovative Research in Engineering and Science (ICIREAS), 1-6. doi: 10.1109/ICIREAS51982.2021.9456035