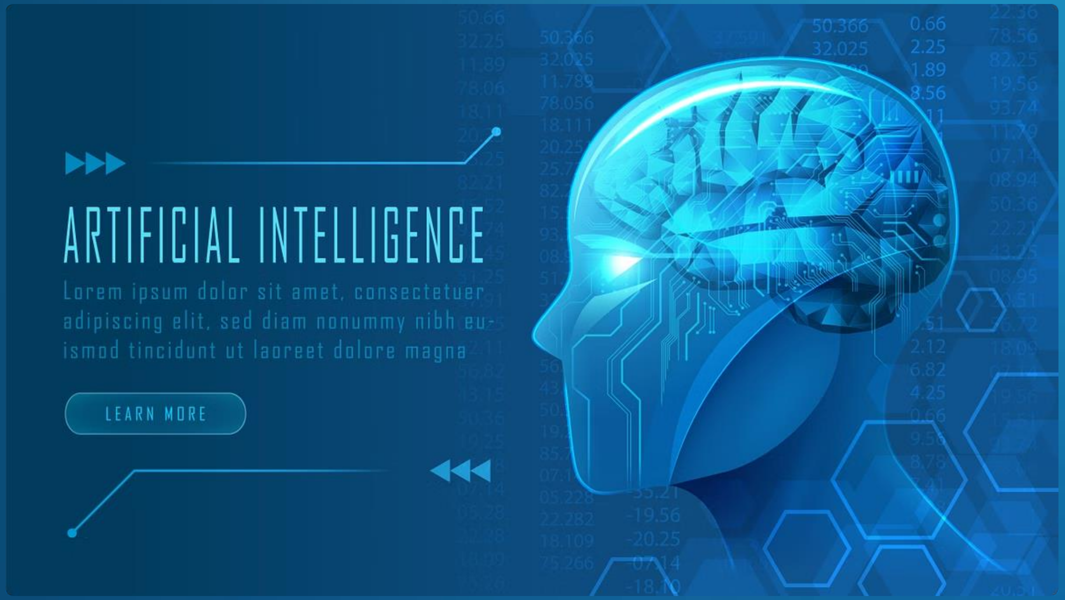

ARTIFICIAL INTELLIGENCE
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna
LEARN MORE

# AI-Based Proctoring System For Secure Assesments

Introducing an innovative AI-driven proctoring system for secure skill assessments. Developed by the pink_sweat team, this solution addresses remote testing challenges. It uses advanced AI to ensure fairness and accuracy in skill evaluations.

Title: AI-Based Proctored Skill Test

Subtitle: AI-Driven Proctoring System for Secure Skill Assessments

Presented by: pink_sweat

Team Members: Hitesh Kumar , Amarnath Kumar, Sourav Kumar Verma

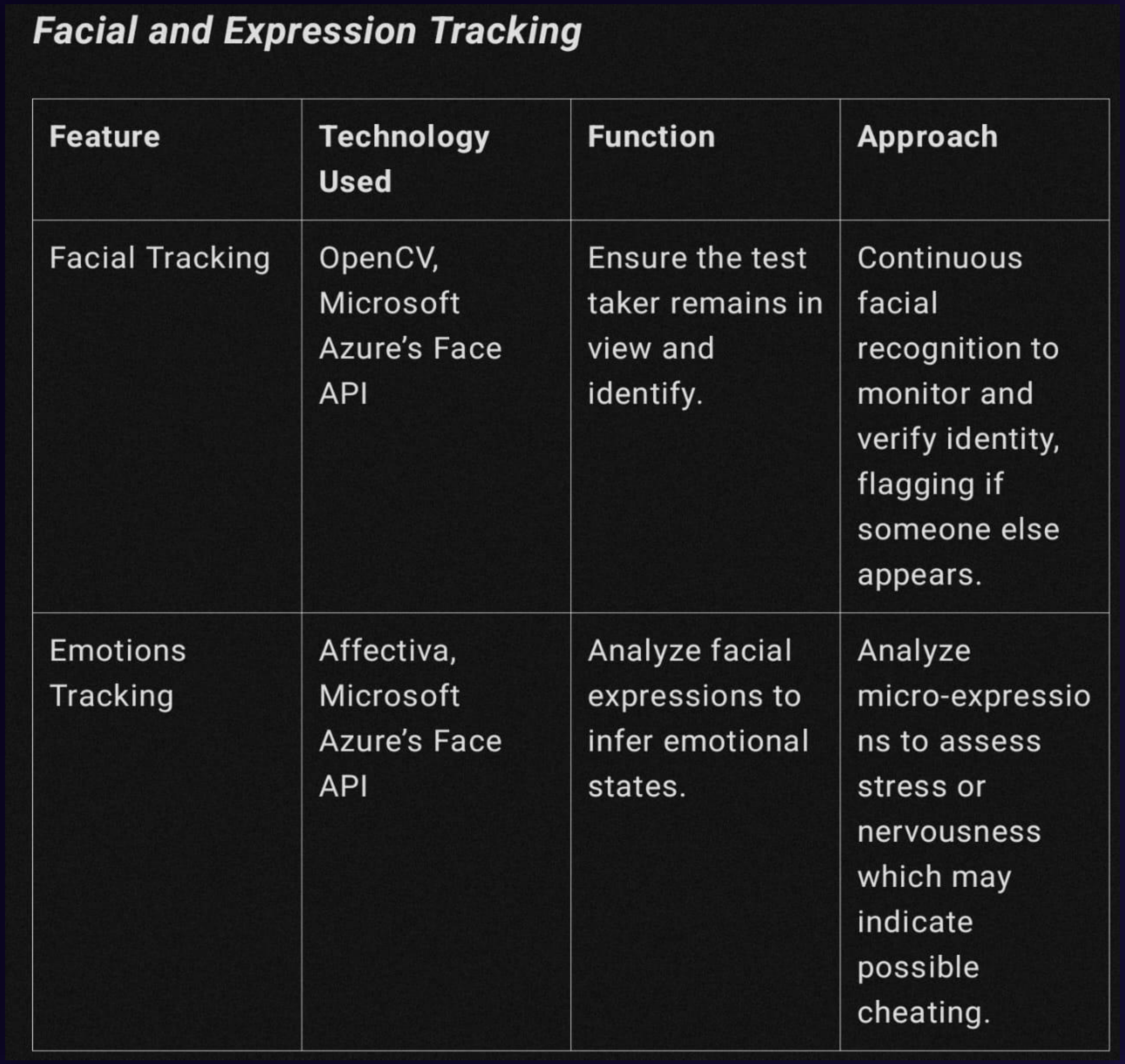
Hackathon: Hack2Hire by SkillMingle.in

Date: 20/10/2024

# Problem Statement Title:

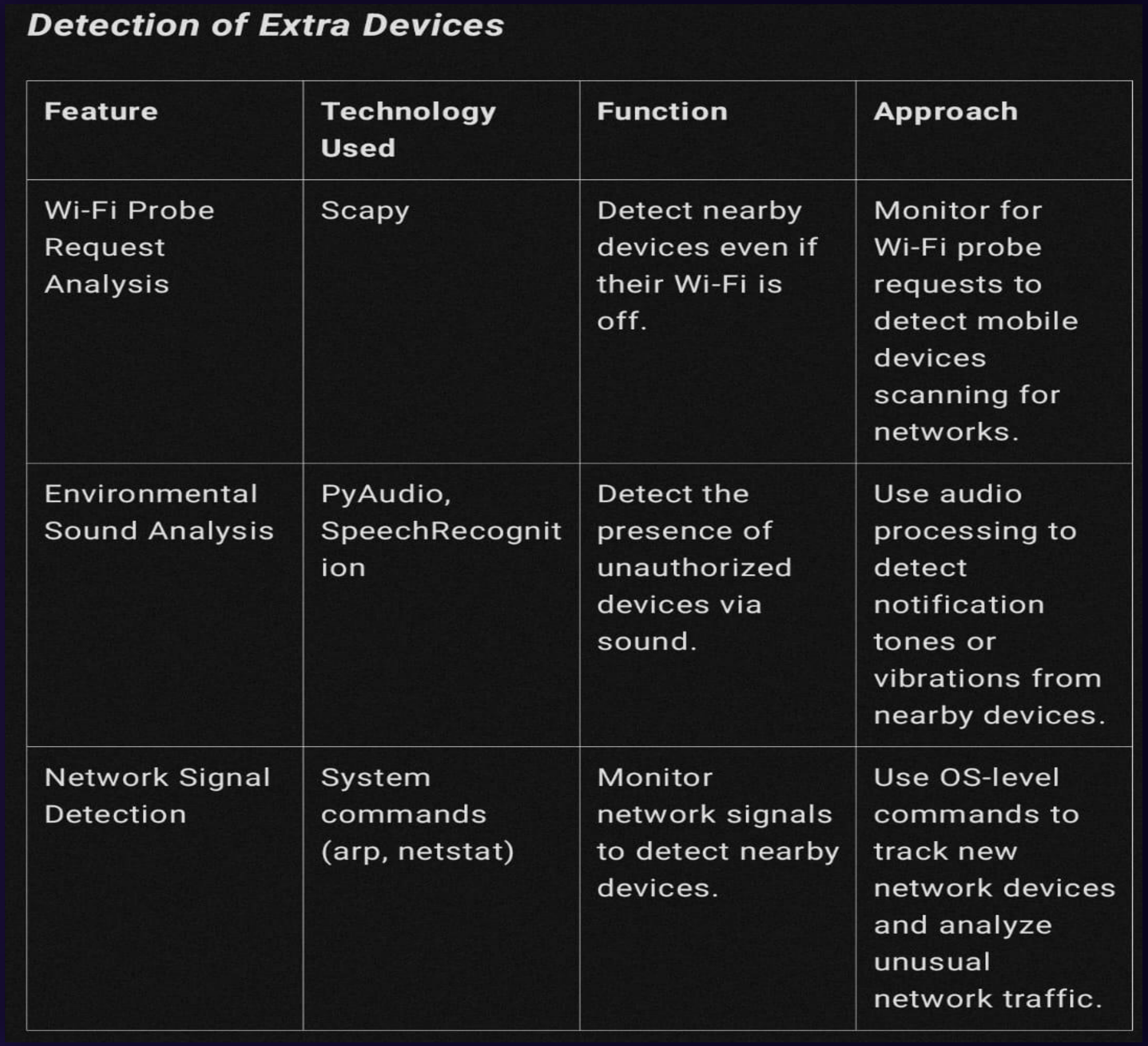
The Challenge in Remote Skill Assessment Content: Ensuring fairness, security, and accuracy in skill tests is difficult with traditional proctoring methods. Objective: Build an AI-powered tool that monitors test-takers in real-time and prevents cheating, ensuring the integrity of skill assessments.

# Uniqueness of the Solution:



***Facial and Expression Tracking***

| Feature | Technology Used | Function | Approach |
|---|---|---|---|
| Facial Tracking | OpenCV, Microsoft Azure's Face API | Ensure the test taker remains in view and identify. | Continuous facial recognition to monitor and verify identity, flagging if someone else appears. |
| Emotions Tracking | Affectiva, Microsoft Azure's Face API | Analyze facial expressions to infer emotional states. | Analyze micro-expressions to assess stress or nervousness which may indicate possible cheating. |



***Detection of Extra Devices***

| Feature | Technology Used | Function | Approach |
|---|---|---|---|
| Wi-Fi Probe Request Analysis | Scapy | Detect nearby devices even if their Wi-Fi is off. | Monitor for Wi-Fi probe requests to detect mobile devices scanning for networks. |
| Environmental Sound Analysis | PyAudio, SpeechRecognition | Detect the presence of unauthorized devices via sound. | Use audio processing to detect notification tones or vibrations from nearby devices. |
| Network Signal Detection | System commands (arp, netstat) | Monitor network signals to detect nearby devices. | Use OS-level commands to track new network devices and analyze unusual network traffic. |

Design At: https://www.canva.com/design/DAGUG2srFM8/rCYEXGUBSj_CtE0ZZDuANQ/view?utm_content=DAGUG2srFM8&utm_campaign=designshare&utm_medium=link&utm_source=editor

# Uniqueness of the Solution:

1. Wi-Fi Probe Requests Analysis (Even if Wi-Fi is Off)

How it Works: This method relies on the fact that many mobile devices, even when their WiFi is turned off, periodically send out probe requests to discover familiar networks. These signals can be detected by monitoring devices.

Method: By using system APIs or network monitoring tools like scapy, you can capture these WiFi probe requests. If unexpected probe requests are detected, it suggests that a mobile device is nearby, possibly searching for a network.

2. Environmental Sound Analysis for Phone Notifications

How it Works: This method utilizes the microphone of a computer or a device to pick up sounds from the environment. Phones often emit notification sounds, vibration noises, or specific ringtone sounds.

Method: Continuous monitoring of environmental sounds through microphones using libraries like SpeechRecognition or PyAudio. An AI sound classification model can analyze these sounds and identify specific notification or ringtone sounds, suggesting the use of a phone nearby.

3. Detect Nearby Network Signals Using the OS Network Stack

How it Works: Even if WiFi or Bluetooth is off on a phone, other network signals or interactions might still occur that can be detected using the primary device’s operating system network stack.

Method: Utilize system-level commands like arp -a to detect new devices in the local network environment or netstat to analyze unusual IP connections or device handshakes. A script can run periodically to check for these signals, indicating nearby devices that might be connecting to cellular networks or briefly activating WiFi.

# Key Features of our Approach

1

## Eyeball and Lips Movement Detection

AI algorithms track eye and lip movements to prevent external communication and distractions. Pre-trained facial landmark detection models and custom models to detect abnormal movements or distractions

2

## Facial Emotion Recognition

Deep learning models analyze test-takers' emotional states and stress levels in real-time. Convolutional Neural Networks (CNN) trained on datasets like FER-2013 to detect emotions (stress, confusion, etc.)

3

## Screen and Audio Monitoring

Comprehensive monitoring of screen activity and audio environment ensures test integrity. Rule-based systems for detecting unauthorized windows or external device usage and Speech-to-text or real-time sound analysis to detect human speech or background noise during the test

4

## Device and Port Checks

Advanced systems detect unauthorized devices and monitor port usage during assessments. Anomaly detection models to flag unauthorized device usage (like USBs or external hardware)

# Architecture & System Design

## Detailed View of Approach

Eyeball and Lips Movement Detection:

- **Technology:** OpenCV, Dlib, or MediaPipe.
- **ML Approach:** Pre-trained facial landmark detection models and custom models to detect abnormal movements or distractions.

Facial Emotion Recognition:

- **Technology:** TensorFlow, Keras, or PyTorch for deep learning models.
- **ML Approach:** Convolutional Neural Networks (CNN) trained on datasets like FER-2013 to detect emotions (stress, confusion, etc.).

Screen and System Monitoring:

- **Technology:** Python libraries like **PyAutoGUI** or **Pynput** for screen monitoring, and **psutil** to check for extra processes and ports.
- **ML Approach:** Rule-based systems for detecting unauthorized windows or external device usage.

Audio Monitoring:

- **Technology:** PyAudio or SpeechRecognition.
- **ML Approach:** Speech-to-text or real-time sound analysis to detect human speech or background noise during the test.

Nearby Device Detection:

- **Technology:** Wi-Fi or Bluetooth signal scanning tools to detect unauthorized devices.
- **ML Approach:** Signal pattern recognition algorithms to distinguish between test-taker’s authorized devices and nearby external devices.

Ports Monitoring for External Devices:

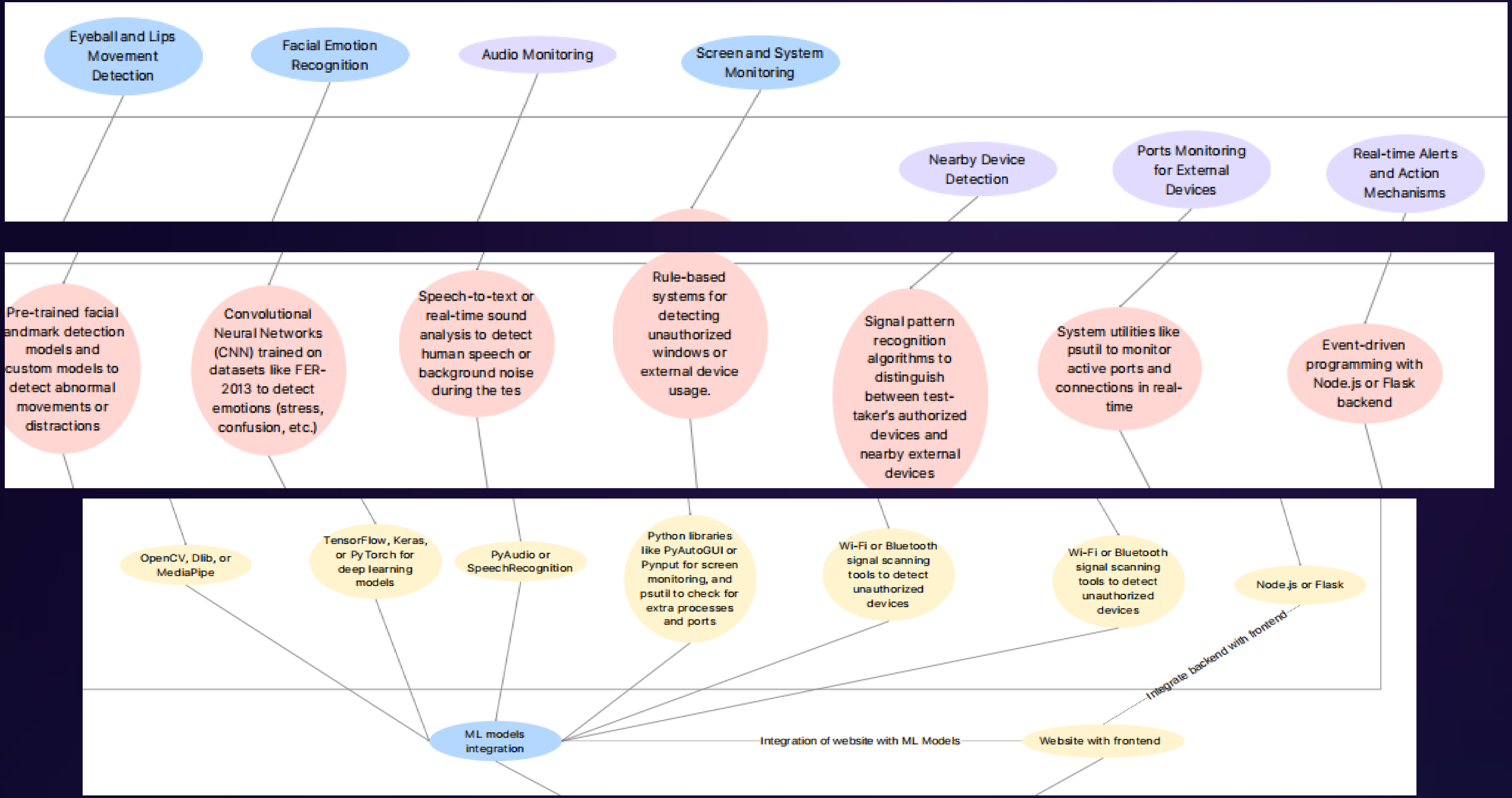
- **Technology:** System utilities like **psutil** to monitor active ports and connections in real-time.
- **ML Approach:** Anomaly detection models to flag unauthorized device usage (like USBs or external hardware).

Real-time Alerts and Action Mechanisms:

- **Technology:** Event-driven programming with **Node.js** or **Flask** backend.
- **ML Approach:** Trigger alert systems based on anomaly detection, suspicious behavior patterns, and security risks.

# Architecture & System Design

Design At: https://www.canva.com/design/DAGUG2srFM8/rCYEXGUBSj_CtE0ZZDuANQ/view?utm_content=DAGUG2srFM8&utm_campaign=designshare&utm_medium=link&utm_source=editor



Eyeball and Lips Movement Detection

Facial Emotion Recognition

Audio Monitoring

Screen and System Monitoring

Nearby Device Detection

Ports Monitoring for External Devices

Real-time Alerts and Action Mechanisms

Pre-trained facial andmark detection models and custom models to detect abnormal movements or distractions

Convolutional Neural Networks (CNN) trained on datasets like FER-2013 to detect emotions (stress, confusion, etc.)

Speech-to-text or real-time sound analysis to detect human speech or background noise during the tes

Rule-based systems for detecting unauthorized windows or external device usage.

Signal pattern recognition algorithms to distinguish between test-taker's authorized devices and nearby external devices

System utilities like psutil to monitor active ports and connections in real-time

Event-driven programming with Node.js or Flask backend

OpenCV, Dlib, or MediaPipe

TensorFlow, Keras, or PyTorch for deep learning models

PyAudio or SpeechRecognition

Python libraries like PyAutoGUI or Pynput for screen monitoring, and psutil to check for extra processes and ports

Wi-Fi or Bluetooth signal scanning tools to detect unauthorized devices

Wi-Fi or Bluetooth signal scanning tools to detect unauthorized devices

Node.js or Flask

Integrate backend with frontend

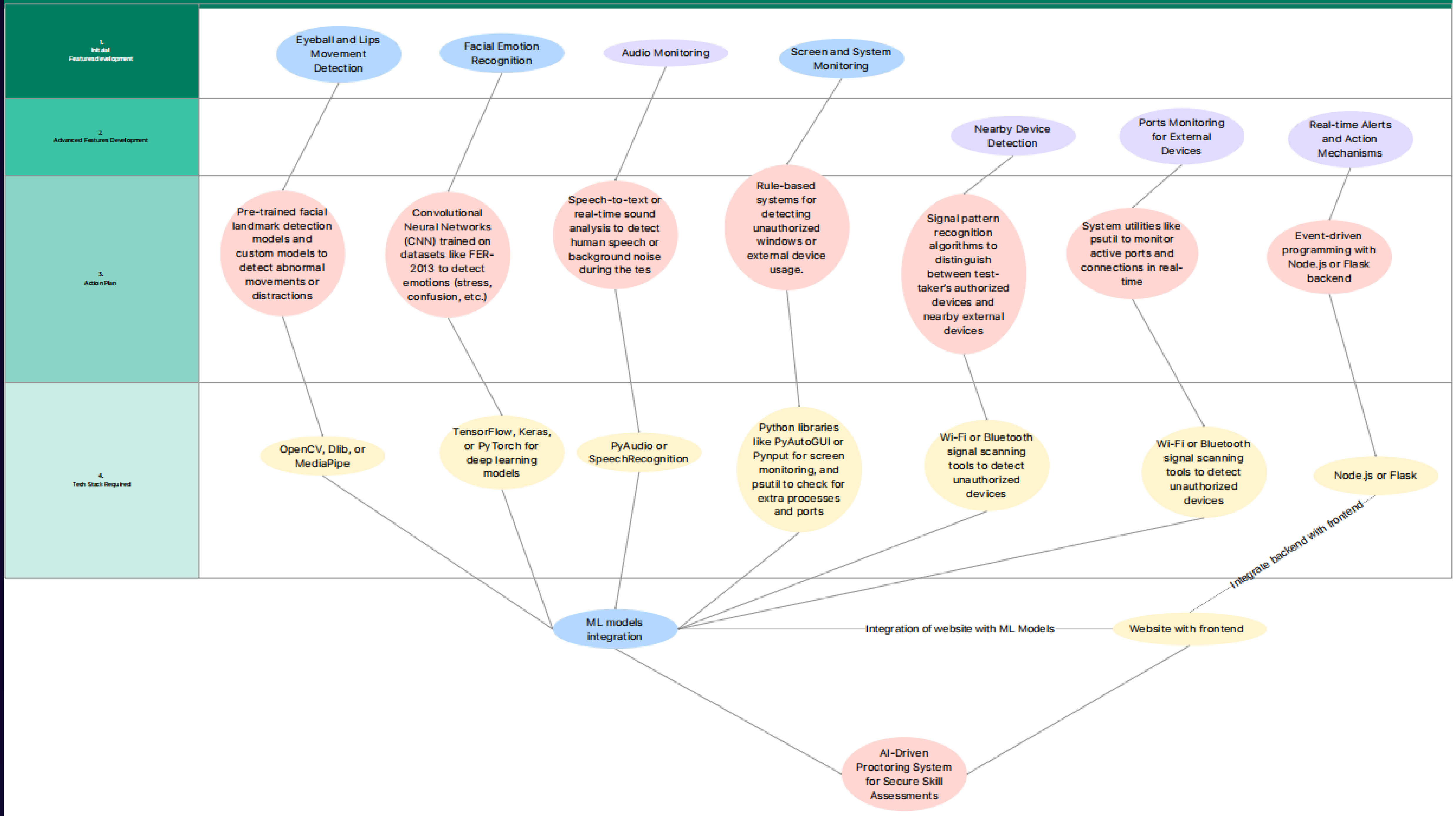
ML models integration

Integration of website with ML Models

Website with frontend

# Architecture and Workflow Overview

AI-Driven Proctoring System for Secure Skill Assessments
BY PINK_SWEAT



1. Initial Features development

Eyeball and Lips Movement Detection

Facial Emotion Recognition

Audio Monitoring

Screen and System Monitoring

2. Advanced Features Development

Nearby Device Detection

Ports Monitoring for External Devices

Real-time Alerts and Action Mechanisms

3. Action Plan

Pre-trained facial landmark detection models and custom models to detect abnormal movements or distractions

Convolutional Neural Networks (CNN) trained on datasets like FER-2013 to detect emotions (stress, confusion, etc.)

Speech-to-text or real-time sound analysis to detect human speech or background noise during the tes

Rule-based systems for detecting unauthorized windows or external device usage.

Signal pattern recognition algorithms to distinguish between test-taker's authorized devices and nearby external devices

System utilities like psutil to monitor active ports and connections in real-time

Event-driven programming with Node.js or Flask backend

4. Tech Stack Required

OpenCV, Dlib, or MediaPipe

TensorFlow, Keras, or PyTorch for deep learning models

PyAudio or SpeechRecognition

Python libraries like PyAutoGUI or Pynput for screen monitoring, and psutil to check for extra processes and ports

Wi-Fi or Bluetooth signal scanning tools to detect unauthorized devices

Wi-Fi or Bluetooth signal scanning tools to detect unauthorized devices

Node.js or Flask

Integrate backend with frontend

ML models integration

Integration of website with ML Models

Website with frontend

AI-Driven Proctoring System for Secure Skill Assessments

# System Process and Audio Monitoring

1

## Screen Monitoring

PyAutoGUI and Pynput track screen activity and enforce full-window test mode.

2

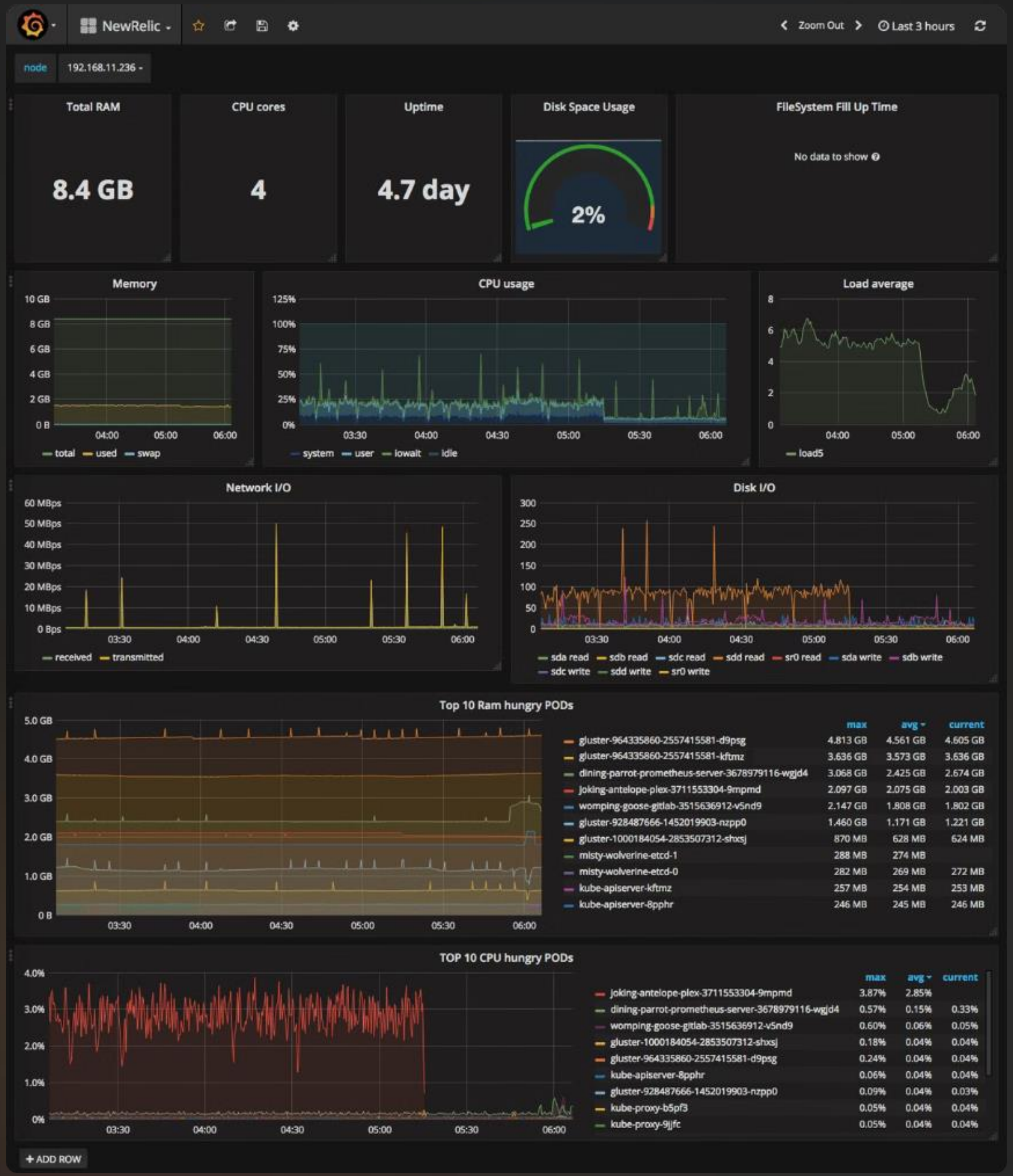
## Process Check

Psutil library monitors system processes to detect unauthorized applications during tests.

3

## Audio Analysis

PyAudio and SpeechRecognition perform real-time sound analysis to detect external conversations.



Total RAM: 8.4 GB

CPU cores: 4

Uptime: 4.7 day

Disk Space Usage: 2%

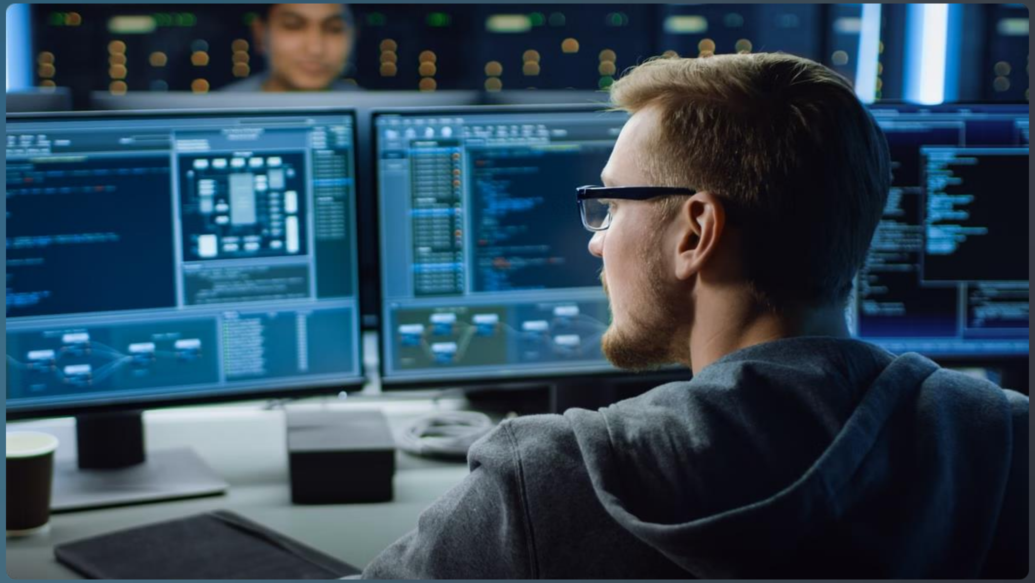
FileSystem Fill Up Time: No data to show

Top 10 Ram hungry PODs

| | max | avg | current |
|---|---|---|---|
| gluster-964335860-2557415581-d9psg | 4.813 GB | 4.561 GB | 4.605 GB |
| gluster-964335860-2557415581-kftmz | 3.636 GB | 3.573 GB | 3.636 GB |
| dining-parrot-prometheus-server-3678979116-wgjd4 | 3.068 GB | 2.425 GB | 2.674 GB |
| joking-antelope-plex-3711553304-9mpmd | 2.097 GB | 2.075 GB | 2.003 GB |
| womping-goose-gitlab-3515636912-v5nd9 | 2.147 GB | 1.808 GB | 1.802 GB |
| gluster-928487666-1452019903-nzpp0 | 1.460 GB | 1.171 GB | 1.221 GB |
| gluster-1000184054-2853507312-shxsj | 870 MB | 628 MB | 624 MB |
| misty-wolverine-etcd-1 | 288 MB | 274 MB | |
| misty-wolverine-etcd-0 | 282 MB | 269 MB | 272 MB |
| kube-apiserver-kftmz | 257 MB | 254 MB | 253 MB |
| kube-apiserver-8pphr | 246 MB | 245 MB | 246 MB |

TOP 10 CPU hungry PODs

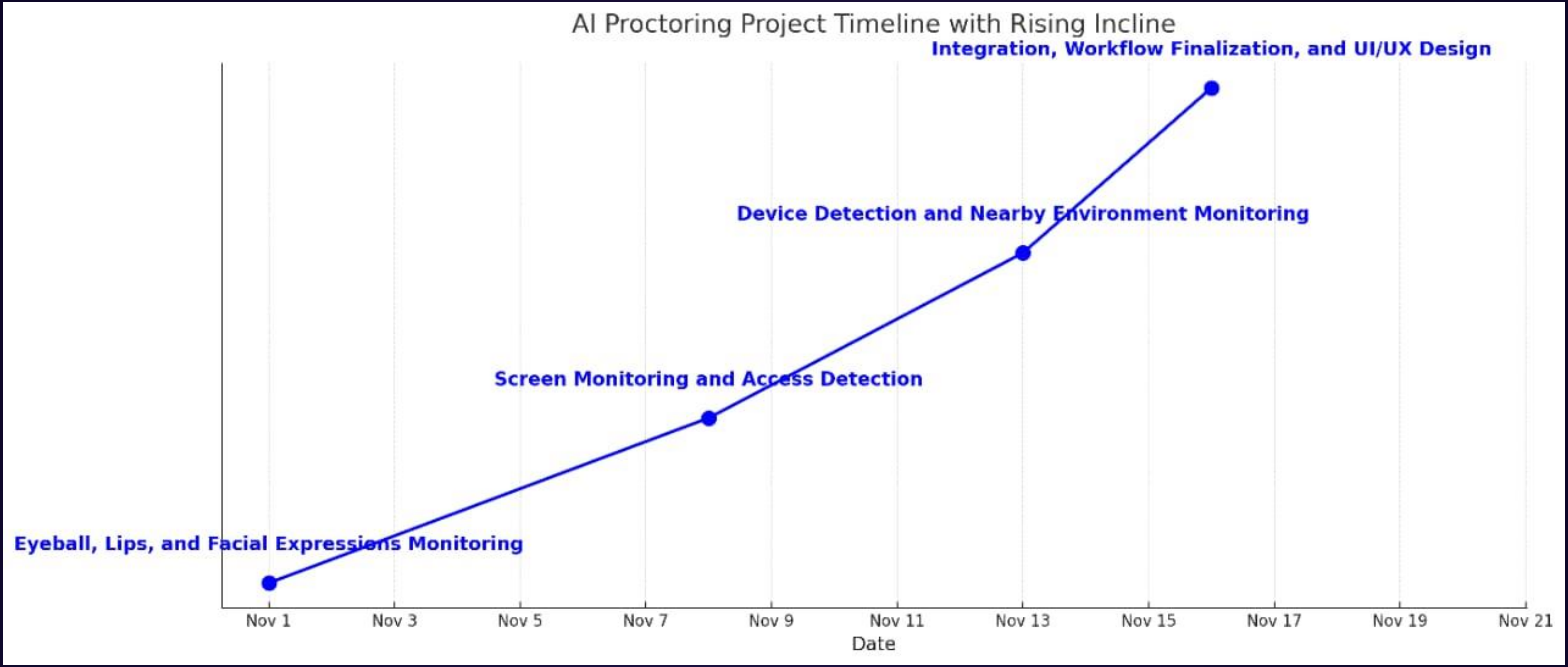
| | max | avg | current |
|---|---|---|---|
| joking-antelope-plex-3711553304-9mpmd | 3.87% | 2.85% | |
| dining-parrot-prometheus-server-3678979116-wgjd4 | 0.57% | 0.15% | 0.33% |
| womping-goose-gitlab-3515636912-v5nd9 | 0.60% | 0.06% | 0.05% |
| gluster-1000184054-2853507312-shxsj | 0.18% | 0.04% | 0.04% |
| gluster-964335860-2557415581-d9psg | 0.24% | 0.04% | 0.04% |
| kube-apiserver-8pphr | 0.06% | 0.04% | 0.04% |
| gluster-928487666-1452019903-nzpp0 | 0.09% | 0.04% | 0.03% |
| kube-proxy-b5pf3 | 0.05% | 0.04% | 0.04% |
| kube-proxy-9jjfc | 0.05% | 0.04% | 0.04% |



# Device Detection and Port Monitoring of Test Taker

| Technology | Function | ML Approach |
|---|---|---|
| Wi-Fi/Bluetooth Scanning | Detect Nearby Devices | Signal Pattern Recognition |
| Psutil | Monitor Active Ports | Anomaly Detection |
| Node.js/Flask | Real-time Alerts | Event-driven Anomaly Triggers |

# Timeline


AI Proctoring Project Timeline with Rising Incline
Integration, Workflow Finalization, and UI/UX Design
Device Detection and Nearby Environment Monitoring
Screen Monitoring and Access Detection
Eyeball, Lips, and Facial Expressions Monitoring
Nov 1
Nov 3
Nov 5
Nov 7
Nov 9
Nov 11
Nov 13
Nov 15
Nov 17
Nov 19
Nov 21
Date

# Future Improvements: Adaptive Systems

1 **Enhanced Behavioral Analysis:**

Implement more sophisticated AI models to detect not just eye or lip movements, but more subtle behavioral patterns, such as typing cadence or body posture, to further prevent cheating.

2 **Voice and Audio Analysis:**

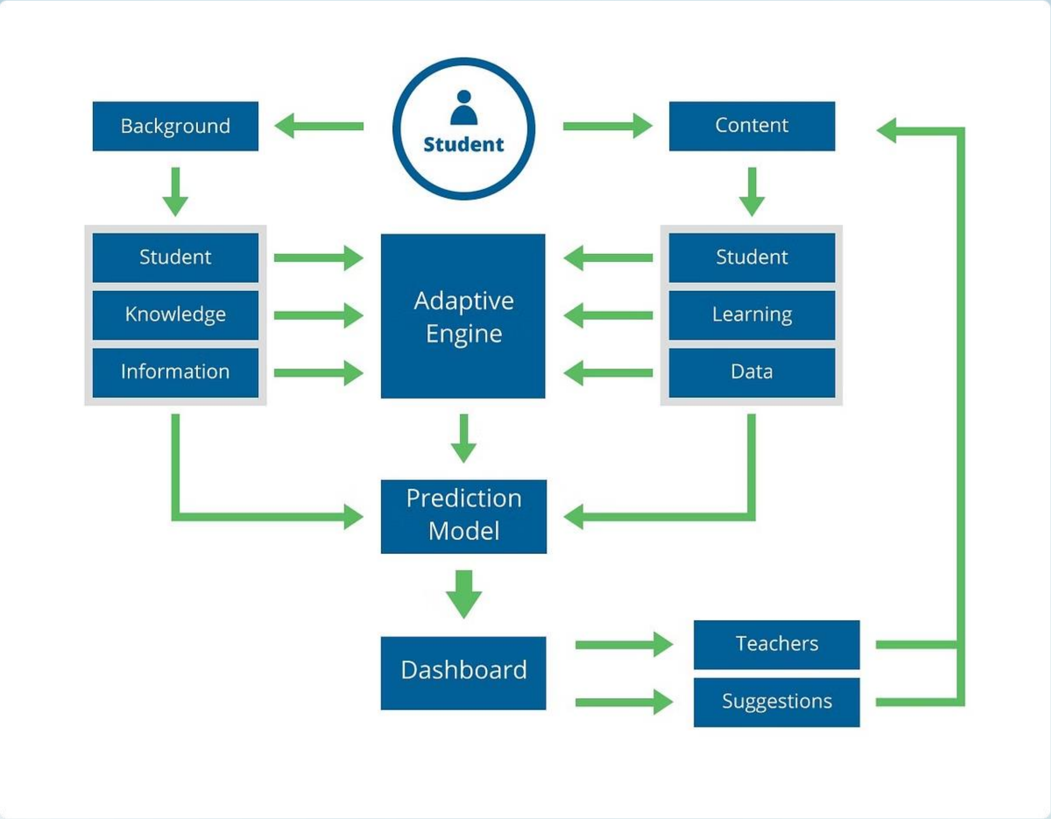
Integrate advanced **Natural Language Processing (NLP)** techniques to analyze voice for conversational cues or external help. This could involve sentiment analysis or detecting pauses that indicate cheating attempts.

3 **Integration with Augmented Reality (AR) for 360-degree Monitoring:**

Use AR/VR tools to provide 360-degree monitoring of the environment around the test-taker, ensuring no external objects or persons are providing help.

4 **Blockchain for Data Integrity:**

Implement **Blockchain technology** to ensure the integrity and authenticity of proctoring data, providing a tamper-proof log of the test session that is transparent and auditable.



Title: AI-Based Proctored Skill Test

Subtitle: AI-Driven Proctoring System for Secure Skill Assessments

Presented by: pink_sweat

Team Members: Hitesh Kumar , Amarnath Kumar, Sourav Kumar Verma

Hackathon: Hack2Hire by SkillMingle.in

Date: 20/10/2024