CheatShield AI - Advanced Exam Surveillance

Project Documentation

1. Project Overview

Start Date: [3rd -jan 2025]
End Date: [23rd -feb 2025]
Project Status: Completed

CheatShield AI is an advanced cheating detection system that uses AI-powered object detection to monitor exam environments in real-time. This tool can analyze video feeds to detect suspicious activities, such as the use of unauthorized devices (cell phones, tablets, books) and suspicious hand movements indicative of paper exchanges.

2. Requirement Analysis

Functional Requirements

- Detect unauthorized devices (mobile phones, books, tablets, laptops) in an exam setting.
- Identify and track suspicious hand movements indicative of paper exchanges.
- Support both live webcam feeds and pre-recorded videos.
- Provide real-time alerts for detected cheating activities.

Non-Functional Requirements

- The system should process video at a minimum of 30 FPS.
- The detection model should maintain at least 85% accuracy.
- The application should have a user-friendly interface with simple navigation.

3. Software Development Life Cycle (SDLC)

Phase 1: Planning

- Define project objectives and scope.
- Identify key technologies (OpenCV, YOLOv8, Tkinter, etc.).
- Allocate tasks and timeline.

Phase 2: Requirement Analysis

- Gather functional and non-functional requirements.
- Analyze hardware and software dependencies.
- Define data flow and processing pipeline.

Phase 3: Design

- Create system architecture.
- Define UI/UX layout for better usability.
- Develop flowcharts and algorithms for detection logic.

Phase 4: Development

- Implement video processing using OpenCV.
- Integrate YOLOv8 for object and hand movement detection.
- Build a Tkinter-based GUI for user interaction.
- Optimize model accuracy and reduce false positives.

Phase 5: Testing

- Perform unit testing for individual modules.
- Conduct integration testing to ensure seamless functionality.
- Test detection accuracy with real-world video samples.
- Debug and optimize performance.

Phase 6: Deployment

- Package and distribute software.
- Provide installation documentation.
- Deploy the model on local machines or cloud infrastructure.

Phase 7: Maintenance & Updates

- Monitor application performance.
- Collect user feedback and improve features.
- Update detection models with new data.

4. Features

- **Real-Time Video Analysis**: Supports both live camera feeds and pre-recorded videos.
- **AI-Powered Detection**: Uses YOLOv8 for object and movement detection.
- **Multiple Cheating Indicators**: Detects unauthorized devices like mobile phones, laptops, books, and tablets.
- **Hand Movement Tracking**: Flags suspicious hand movements indicating potential paper exchanges.
- **User-Friendly Interface**: Simple UI built using Tkinter.

5. System Requirements

Ensure the following dependencies are installed before running the application:

```
pip install opencv-python numpy tkinter torch ultralytics pillow
```

6. Installation

1. Clone the repository:

2. Install dependencies:
3. Download the YOLOv8 model (if not included):

4.
   ```
   wget
   https://github.com/ultralytics/yolov8/releases/download/v8.0/yolov8n.pt
   ```

7. Usage

Run the Application

To start the application, run:

```
python cheatshield_ai.py
```

Options

- **Upload Video**: Allows you to select and analyze a pre-recorded video.
- **Use Camera**: Uses a live webcam feed to detect cheating in real-time.

8. How It Works

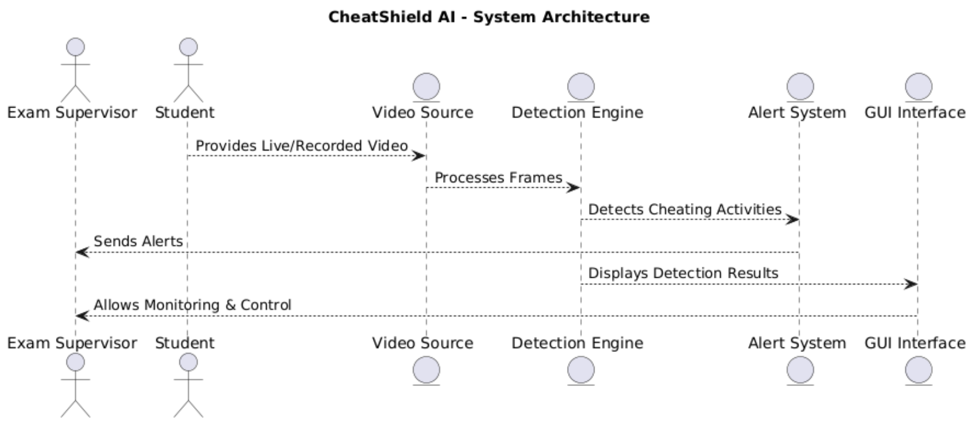
1. The application loads the YOLOv8 model to detect objects in frames.
2. It scans for **faces**, **hands**, and **unauthorized devices**.
3. If a hand moves significantly between two different persons, it flags a **suspicious hand movement**.
4. If a cheating tool (like a mobile phone or book) is detected, it displays an alert.

9. Testing Strategy

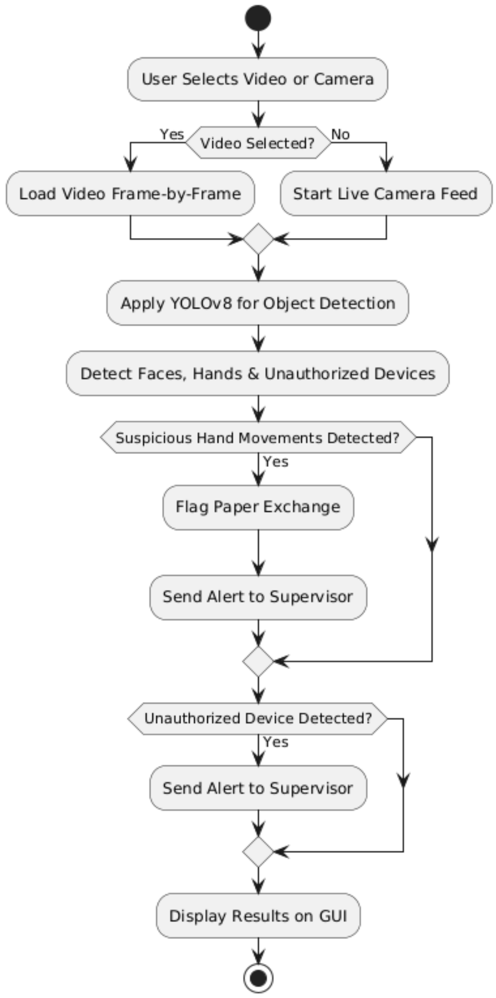
- **Unit Testing**: Verify individual modules like object detection, video processing, and GUI.
- **Integration Testing**: Ensure all components work together without errors.
- **Performance Testing**: Measure real-time processing efficiency.
- **User Acceptance Testing (UAT)**: Collect feedback from exam proctors and administrators.

Uml diagram

Sequential diagram



Activity diagram


User Selects Video or Camera
Yes
Video Selected?
No
Load Video Frame-by-Frame
Start Live Camera Feed
Apply YOLOv8 for Object Detection
Detect Faces, Hands & Unauthorized Devices
Suspicious Hand Movements Detected?
Yes
Flag Paper Exchange
Send Alert to Supervisor
Unauthorized Device Detected?
Yes
Send Alert to Supervisor
Display Results on GUI

Architecture

For the **CheatShield AI - Advanced Exam Surveillance**, the system architecture follows a **Hybrid AI-Based Surveillance Architecture** that includes:

Architecture Type:

- **AI-Powered Real-Time Object Detection** (Using YOLOv8 & OpenCV).

- **Client-Server Architecture** (If cloud-based deployment is used).
- **Standalone Application Architecture** (For local machine processing).

Architecture Components:

1. **User Interface (UI) Layer:**
   - Built using **Tkinter** for GUI.
   - Allows users to select video sources (Live Camera or Uploaded Video).
   - Displays detection alerts.
2. **Processing Layer:**
   - **OpenCV & MediaPipe** handle video frame processing.
   - **YOLOv8** detects unauthorized objects (mobile phones, books, hands).
   - **Custom AI Logic** analyzes hand movements to flag paper exchanges.
3. **Alert System Layer:**
   - Generates alerts when cheating behavior is detected.
   - Displays warning messages in real-time.
4. **Storage & Logging (Optional - If Cloud-Based):**
   - Saves logs of flagged events for later review.
   - Could use **local storage or a cloud-based database**.

Deployment Options:

✅ **Local Machine** (Standalone Application) – Runs on the user's PC.
✅ **Cloud-Based AI Model** – If integrated with a server, could allow remote monitoring.