Riphah International University Lahore, Pakistan

Riphah School of Computing & Innovation

FINAL YEAR PROJECT

PROJECT PROPOSAL & PLAN

HawkEye

Project Team

Student Name	Student ID	Program	Contact Number	Email Address
Daniyal Wajid	48528	BSSE	0309-1840367	48528@students.riphah.edu.pk
Uzair Hassan	48525	BSSE	0325-4039356	48525@students.riphah.edu.pk

Ms. Uzma Kiran (Lecturer)

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
	1.0		Original Draft	
			Changes Based on Feedback from Supervisor	
			Changes Based on Feedback From Faculty	
			Added Project Plan	
			Changes Based on Feedback from Supervisor	

Project Proposal

Project Title: HawkEye

Executive Summary

Hawkeye is an intelligent discipline monitoring system that leverages artificial intelligence and computer vision to maintain campus order and safety. Using live camera feeds, it automatically detects violations such as fighting, possession of prohibited items, or improper uniforms. The system records incidents, identifies the students involved, and updates their disciplinary records. It also automates fines and penalty management to reduce manual supervision. Through a web-based dashboard, administrators can view incidents, manage records, and analyze trends. The project aims to create a safer, technology-driven campus environment through smart surveillance and automation.

1. Introduction

Educational institutions face increasing challenges in maintaining discipline and ensuring safety on campus. Manual supervision is often inefficient and prone to human error. To address this, Hawkeye provides an automated solution that continuously monitors student behavior and identifies rule violations using AI-based video analysis. The system benefits both administrators and students by promoting a disciplined, transparent, and secure environment.

2. Existing System / Competitive Analysis

System / Vendor	Manual Monitoring	Violence / Weapon Detection	Uniform / ID Detection	Multi-Task Behaviour Analytics	Integrates with Existing Cameras / VMS	Student-Record Integration	Policy-Based Penalties
Traditional CCTV	✓	✗	✗	✗	✓	✗	✗
Genetec Security Center	✓	✓	✗	✓	✓	✗	✗
Avigilon (Motorola)	✓	✓	✗	✓	✓	✗	✗
BriefCam Analytics	✓	✓	✗	✓	✓	✗	✗
Irisity (School Guard)	✓	✓	✗	✓	✓	✗	✗
Nyckel AI (Uniform Classifier)	✗	✗	✓	✗	✓	✗	✗
ZeroEyes (Gun Detection)	✓	✓	✗	✗	✓	✗	✗
Omnilert (AI Gun Detection)	✓	✓	✗	✗	✓	✗	✗
Intelgic Vision AEye	✓	✓	✗	✓	✓	✗	✗
HawkEye	✓	✓	✓	✓	✓	✓	✓

3. Problem Statement

There is no automated and intelligent system that continuously monitors student activities on campus to detect disciplinary violations and manage penalties efficiently. Manual supervision is prone to oversight, delayed response, and lack of proper incident records. A smart, real-time AI-based solution is needed to maintain discipline effectively and enhance campus safety.

4. Proposed Solution

Hawkeye provides an AI-driven automated surveillance and discipline management solution for educational campuses. It integrates live CCTV feeds with advanced computer vision models to detect violations such as fighting, possession of weapons, smoking, or improper uniform usage in real time. Once a violation is detected, the system captures image evidence, identifies the students involved through facial recognition or student-ID matching, and automatically logs the incident in the database. Each record includes details such as date, time, location, and violation type.

The system then updates the respective student's disciplinary profile and, based on predefined institutional policies, calculates penalties or fines automatically. Administrators can review all incidents through a centralized dashboard, where they can verify, approve, or adjust penalties. The dashboard also provides analytics to identify recurring offenders and common violation types. Hawkeye aims to minimize manual monitoring, reduce bias, and ensure consistent enforcement of campus rules. The overall solution enhances safety, accountability, and administrative efficiency through AI-powered automation.

5. Scope of the Project

The Hawkeye system is designed to automate campus discipline monitoring through AI-based video analysis and smart data management. The project consists of several major modules, each focusing on a specific functionality that contributes to the overall goal of maintaining a safe and well-disciplined environment.

1. Real-Time Monitoring Module:

This module connects to live CCTV cameras installed across the campus. It uses AI and computer vision algorithms to continuously analyze video feeds for suspicious or prohibited activities such as fighting, possession of weapons, or improper uniforms. It ensures 24/7 surveillance without the need for constant human monitoring.

2. Violation Detection and Logging Module:

Once a violation is detected, this module captures relevant image frames, timestamps, and location details. It automatically logs the incident in the system database along with the type of violation detected. The data is securely stored and can be accessed later for verification and recordkeeping.

3. Student Record and Penalty Management Module:

This module maintains digital profiles for all students, tracking their violation history. Based on predefined rules and policies, it calculates appropriate fines or penalties automatically. Administrators can review and modify penalties and maintain transparent disciplinary records.

4. Administrator Dashboard Module:

This web-based interface allows authorized staff to monitor real-time activity, review logged incidents, and generate analytical reports. The dashboard provides graphical summaries of violations, recurring offenders, and campus hotspots to support data-driven decisions.

5. Notification and Alert Module:

This component generates instant alerts for administrators or security personnel when serious violations are detected. It helps ensure timely intervention and enhances the overall response efficiency of campus authorities.

Excluded Features:

- Integration with national ID or police databases.
- Audio-based violation detection or voice recognition.
- Online payment processing for fines or penalties.

6. System Architectural Design

Hardware Components

1. CCTV / IP Cameras:

These serve as the primary data sources for the system. Cameras installed throughout the campus capture continuous video streams of student activities. Drone feeds can also be integrated for outdoor monitoring of large areas such as playgrounds or parking zones.

2. Server Infrastructure:

GPU-enabled cloud or on-premise servers (e.g., AWS EC2, Google Cloud Compute Engine, or Azure GPU instances) host the AI processing engine. These servers perform real-time video analysis for behavior detection, uniform compliance verification, and identification of prohibited activities.

3. Administrator / Security Officer Devices:

Authorized personnel access the system through desktops, laptops, or tablets. These devices are used for reviewing detected incidents, managing student records, and generating analytical reports via the web-based dashboard.

Software Components

1. AI Processing Engine:

The AI module forms the core of the system. It utilizes deep learning frameworks such as YOLO, OpenCV, and TensorFlow to detect actions or objects in live video feeds. The engine identifies violations including fighting, smoking, or possession of restricted items, and sends the detected events to the backend server.

2. Backend Server: Developed using Flask, FastAPI, or Node.js, the backend handles communication between the AI engine, cloud database, storage, and the web dashboard. It processes event data, saves media files, triggers notifications, and manages user authentication and system commands.

3. Cloud Database: A secure cloud database (e.g., Firebase Firestore or MongoDB Atlas) stores structured information such as detected events, timestamps, violation types, and student disciplinary records. This data supports reporting, analytics, and historical tracking.

4. Cloud Storage: Media files such as images, short video clips, and snapshots of detected incidents are stored in AWS S3 or Google Cloud Storage. This ensures scalability and easy retrieval of media evidence for verification and auditing.

5. Notification Service: An integrated notification module sends real-time alerts via SMS, email, or push notifications to administrators and security personnel when serious violations occur, enabling immediate response.

6. Monitoring Dashboard: The frontend dashboard, developed in React.js (for web) and optionally Flutter (for mobile), provides real-time visualization of camera feeds, incidents, and reports. It allows authorized users to manage cameras, review violation logs, monitor trends, and generate performance analytics.

7. Security Layer: The system employs SSL/TLS encryption, JWT-based authentication, and role-based access control (RBAC) to protect user data, video streams, and administrative operations. This ensures data integrity, privacy, and compliance with institutional security standards.

Network Components

1. Video Stream Transmission:

Live video feeds from campus CCTV or IP cameras are transmitted over a secure network connection to the AI Processing Engine for real-time analysis.

2. Backend Communication Flow:

Detected events are forwarded from the AI engine to the backend server through secure internal APIs. The backend coordinates with the database, storage, and notification modules via encrypted RESTful or WebSocket connections.

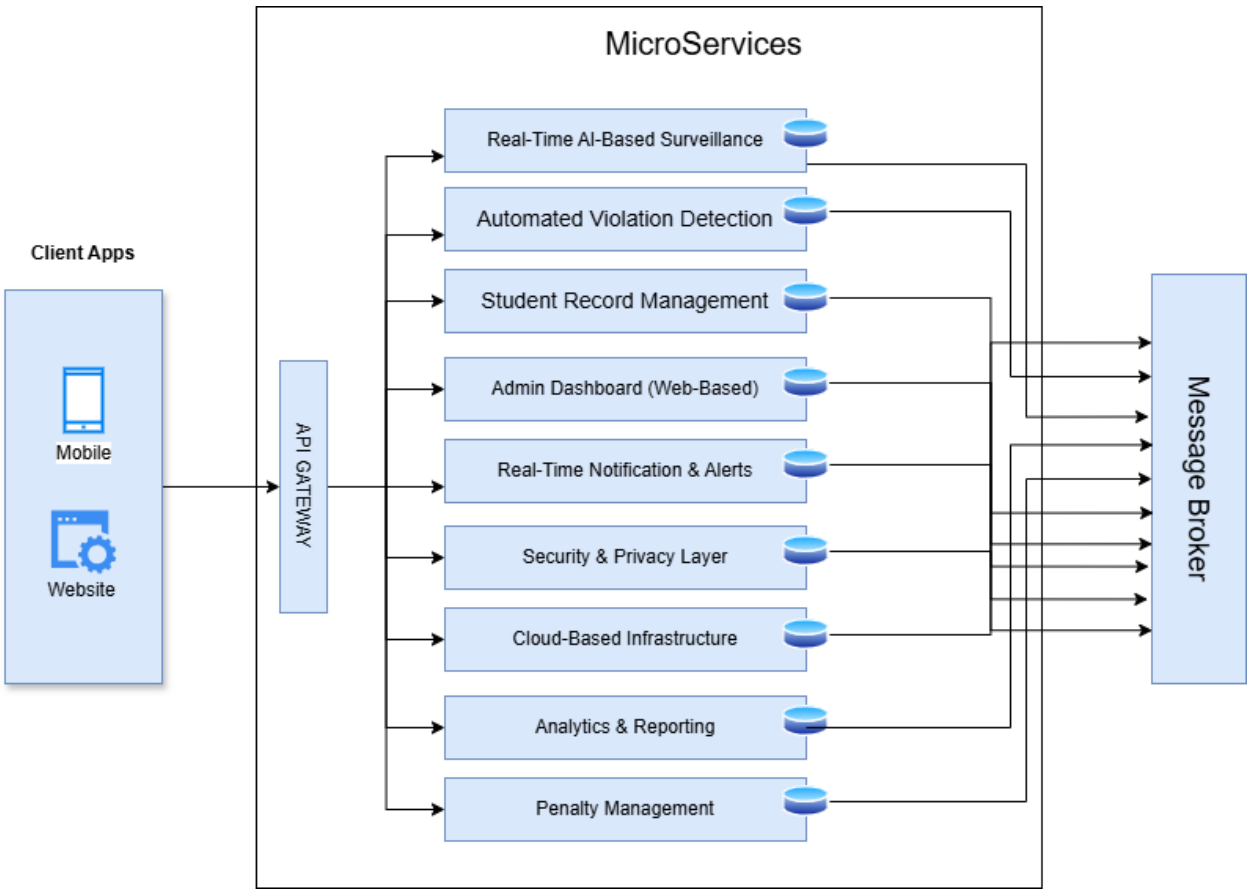
3. User Access:

Administrators and security officers access the HawkEye Monitoring Dashboard through HTTPS-secured web browsers. The interface allows remote monitoring and control from any authorized device with internet access.

4. External Integrations: Future integrations may include external disciplinary management systems or third-party APIs for enhanced student identification or security operations.

5. Data Protection and Privacy: All data transmitted across the network, including video streams, event details, and user credentials, are encrypted using industry-standard protocols. Access is strictly controlled, ensuring only authorized users can view or modify sensitive information.

Microservices Architecture



7. Implementation Tools and Techniques

The implementation of **HawkEye** follows a **modular and agile development approach** to ensure scalability, flexibility, and efficient iteration during the entire project lifecycle. The system integrates artificial intelligence, computer vision, and web technologies to provide a reliable, automated, and intelligent discipline monitoring platform.

The **Agile methodology** is adopted to promote continuous feedback, incremental progress, and early validation of key components such as real-time violation detection, incident logging, and the administrative dashboard. Each sprint focuses on developing and testing specific modules including AI model integration, backend logic, cloud connectivity, and frontend visualization.

Implementation Tools and Technologies

- **Frontend Development:**

  The web-based Monitoring Dashboard is developed using React.js for a responsive and interactive user interface. HTML5, CSS3, and Tailwind CSS are used to design a clean and accessible layout suitable for both desktop and tablet screens.

- **Backend Development:**

  The backend is implemented using Flask, FastAPI, or Node.js to manage communication between the AI engine, database, and user interface. It handles authentication, API endpoints, and data routing efficiently.

- **AI & Computer Vision:**

  The AI engine employs YOLO, OpenCV, and TensorFlow for real-time object detection, behavior recognition, and uniform compliance verification. The models are trained on custom datasets representing campus activities and common violation types.

  - **Database Management:**
    Firebase Firestore or **MongoDB Atlas** is used to store structured data such as detected events, timestamps, student profiles, and violation records. These databases provide high availability, scalability, and secure access control.

  - **Notification Services:**
    The notification system integrates **Firebase Cloud Messaging (FCM)** or **Twilio API** to send instant alerts through email, SMS, or push notifications to campus administrators and security officers.

  - **Security Tools:**
    The platform ensures data protection using **SSL/TLS encryption**, **JWT-based authentication**, and **role-based access control (RBAC)**. These mechanisms prevent unauthorized access and maintain the integrity of sensitive student and incident data.

  - **Version Control:**
    Git and **GitHub** are employed for version management, collaborative coding, and tracking changes throughout the project lifecycle.

8. Project Plan

The **Hawk Eye** project will be executed in multiple well-defined phases to ensure systematic progress and successful completion. The entire project will be managed using an **Agile methodology**, allowing iterative development, continuous improvement, and timely stakeholder feedback. The project spans from **October 2025 to April 2026**, and is divided into two main phases: **Documentation Phase** and **Implementation Phase**.

1. Documentation Phase (October 2025 – January 2026)

This phase focuses on the research, planning, and design aspects of the Hawk Eye system. The key objectives include defining project scope, gathering functional and non-functional requirements, and creating detailed design specifications.

Major tasks in this phase include:

- **Requirement Gathering and Analysis:** Identifying project goals, stakeholders, and system needs.
- **Project Charter & Proposal Creation:** Defining objectives, scope, and constraints.
- **Feasibility Study & Research:** Evaluating technical feasibility and suitable technologies (YOLO/OpenCV for detection, TensorFlow for AI training).
- **System and UML Design:** Developing system architecture, data flow diagrams, and use case models.
- **UI/UX Prototyping:** Designing user-friendly interfaces to ensure smooth interaction.
- **Final Documentation:** Preparing the documentation for approval before moving to implementation.

2. Implementation Phase (February 2026 – April 2026)

This phase involves developing, integrating, and testing all major components of the Hawk Eye system. The development process will follow Agile sprints to ensure incremental progress and quick adaptation to changes.

Key activities include:

- **Module Development:** Implementing backend and frontend components.
- **AI Model Training and Integration:** Training object detection and tracking models using YOLO/OpenCV and integrating them into the system.
- **Database Configuration:** Setting up data storage and retrieval mechanisms.
- **API Development & Integration:** Linking modules and ensuring seamless communication between components.
- **System Testing and Debugging:** Conducting unit testing, integration testing, and performance evaluation.

- **Deployment and Evaluation:** Deploying the final system and ensuring it meets the defined requirements.

3. Final Phase (April 2026)

The last phase includes documentation, presentation, and evaluation of the final system.

Tasks include:

- **User Manual Preparation**
- **Final Report Compilation**
- **Project Presentation and Demonstration**

8.1. Work Breakdown Structure

A **Work Breakdown Structure (WBS)** provides a hierarchical decomposition of the entire Hawk Eye project into manageable sections, covering all essential deliverables from documentation to AI-based system development, testing, and deployment. The tasks are distributed between two team members **Uzair** and **Daniyal** ensuring a balanced workload, accountability, and smooth workflow.

WBS Hierarchy

1. Project Management

1.1 Project Charter & Scope Definition
1.2 Work Breakdown Structure (WBS) Creation
1.3 Roles & Responsibility Matrix
1.4 Schedule & Resource Allocation
1.5 Risk Assessment and Mitigation Planning
1.6 Progress Monitoring and Reporting

2. Reports / Documentation

2.1 Introduction & Problem Statement
2.2 Literature Review / Related Work
2.3 Feasibility Study (Technical & Operational)
2.4 Requirement Gathering and Analysis
2.5 System Design Documentation
2.6 Implementation Strategy & Plan
2.7 Testing and Evaluation Plan
2.8 Final Report and Conclusion
2.9 End User Guide
2.10 Technical & Administrator Documentation

3. System Development

3.1 Development Environment Setup

3.1.1 IDE and Tools Installation (VS Code, Anaconda, etc.)
3.1.2 Version Control Configuration (Git/GitHub)
3.1.3 AI Libraries and Dependencies Installation (YOLO, OpenCV, TensorFlow)
3.1.4 Database Setup and Configuration (Firebase / SQL)

3.2 Detection & Tracking Engine

3.2.1 Dataset Collection & Preprocessing
3.2.2 Object Detection Model Training (YOLO / TensorFlow)
3.2.3 Real-time Tracking Implementation (OpenCV)
3.2.4 Accuracy and Performance Optimization
3.2.5 Model Testing and Validation

3.3 Backend Development

3.3.1 API Design and Development (FastAPI / Flask)
3.3.2 Model Integration with Backend
3.3.3 Data Storage and Retrieval Logic
3.3.4 Authentication and Access Control
3.3.5 Logging and Error Handling Mechanism

3.4 Frontend Development

3.4.1 UI/UX Design and Wireframing
3.4.2 Dashboard Development (React.js / Flutter Web)
3.4.3 Real-time Detection Visualization
3.4.4 User Authentication & Role Management
3.4.5 Reporting and Alerts Module

3.5 Testing and Evaluation

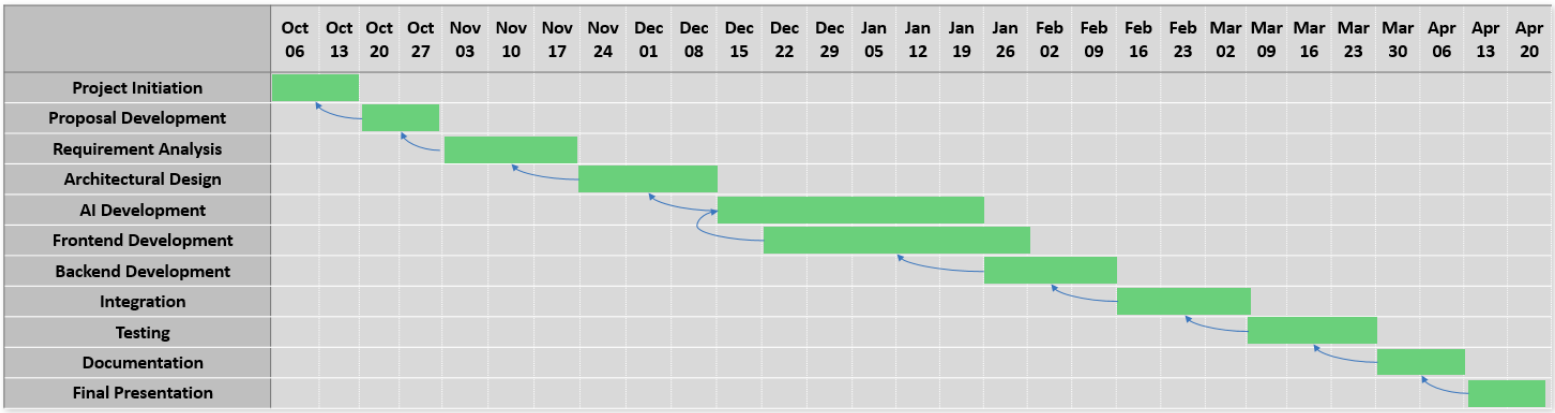
3.5.1 Unit and Integration Testing
3.5.2 Model Accuracy Evaluation
3.5.3 Performance and Load Testing
3.5.4 Usability and UI Testing
3.5.5 Bug Fixing and Improvements

Roles & Responsibility Matrix:

The purpose of roles & responsibility matrix is to identify who will do what.

WBS #	WBS Deliverable	Activity #	Activity to Complete the Deliverable	Duration (# of Days)	Responsible Team Member(s) & Role(s)
1	Project Management	1.1-1.4	Define WBS, assign roles, and manage project schedule	7	Uzair & Daniyal
2	Documentation Introduction	2.1	Write project overview, objectives, and problem definition	8	Uzair
3	Literature Review / Feasibility Study	2.2	Research existing AI surveillance systems and methods	10	Uzair
4	Requirements Analysis	2.3	Identify functional, non-functional, and technical requirements	10	Uzair & Daniyal
5	System Design	2.4	Create system architecture diagrams, data flow, and ER models	12	Daniyal
6	Implementation Planning	2.5	Define development phases, module dependencies, and timeline	6	Both
7	Development Environment Setup	3.1	Install IDEs, configure Git/GitHub, set up YOLO, OpenCV, and database	8	Daniyal
8	Detection & Tracking Engine	3.2	Collect dataset, train YOLO model, and integrate OpenCV	25	Daniyal
9	Backend Development	3.3	Build FastAPI backend, integrate AI model, and manage APIs	20	Uzair
10	Frontend Development	3.4	Design user interface, implement dashboard, and display live detection	20	Uzair
11	Testing & Evaluation	3.5	Conduct unit testing, model accuracy testing, and UI evaluation	15	Both
12	Final Documentation & Reports	2.7-2.10	Prepare final report, admin guide, and technical documentation	15	Uzair
13	Presentation & Submission	----	Prepare final demo presentation and report submission	5	Both

8.2. Gantt Chart



DataSets:

Dataset Name	Purpose	Source
UCF Crime Dataset	Detects violent or abnormal activities like fights or assaults in surveillance footage.	UCF Center for Research in Computer Vision
Hockey Fight Dataset	Used for training fight vs. non-fight detection models.	Kaggle
DeepFashion Dataset	Helps detect and classify clothing to check uniform compliance.	CUHK Multimedia Lab
Fashion-MNIST	Basic clothing recognition and testing.	Zalando Research
COCO Dataset	Detects common objects and background elements.	COCO Consortium
Gun Detection Dataset	Identifies weapons or prohibited items.	Roboflow / Kaggle
VGGFace2 Dataset	Supports student face recognition and identity verification.	University of Oxford
Custom Campus Dataset	Collected from campus cameras for fine-tuning the model.	Created manually

References

1. Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767. Retrieved from https://arxiv.org/abs/1804.02767
2. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934. Retrieved from https://arxiv.org/abs/2004.10934
3. Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools. Retrieved from https://opencv.org

List of Faculty Proposed Changes

Project Title

Proposed Change	Proposed By	Supervisor's Decision
	Name of Faculty Member(s) who proposed this change	Approved/Disapproved and/or Comments

Date: _____ **Supervisor's Signature: _____**

APPROVAL

_____

Project Supervisor

Comments: _____

_____

_____

_____

Name:_____

Date:_____ Signature:_____

Project Manager

Comments: _____

_____

_____

_____

_____

Date:_____ Signature:_____