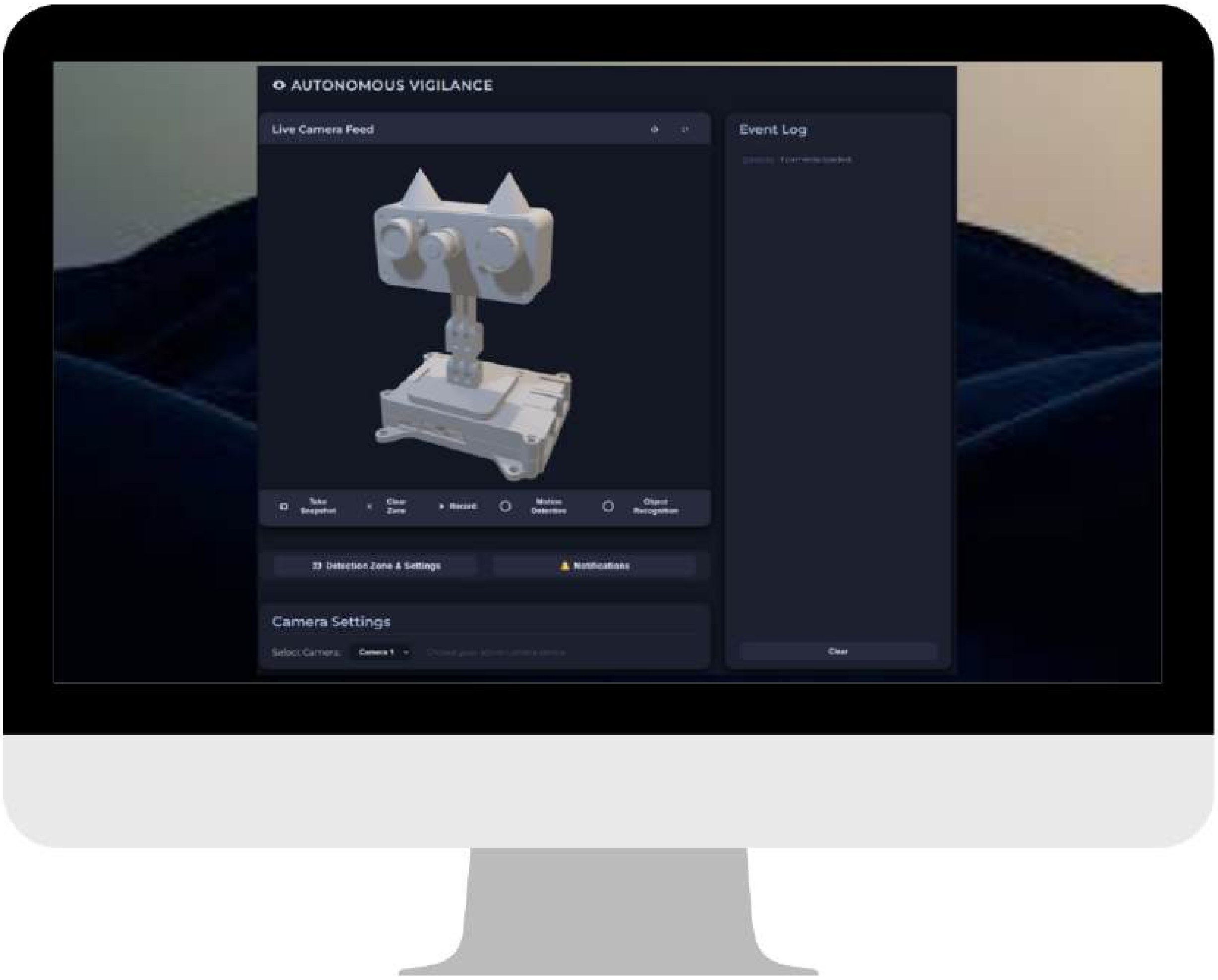
Autonomous Vigilance



Select Camera: Camera 1

By: RYAN KIPCHUMBA & RAYYAN SHAIK

Table of contents

1. Identifying and defining

Guiding steps for your enterprise project.

1. Describe the problem (or opportunity) and why you have selected it for this project.

The project addresses the growing need for intelligent, customisable surveillance systems with reliable night-vision capabilities in residential and small-business settings. Traditional systems often lack adaptive AI-driven threat detection, suffer from poor low-light performance, and provide limited user control. This project was selected due to rising security concerns and advancements in affordable AI and computer vision technologies that enable innovative solutions. According to the NSW Bureau of Crime Statistics and Research, there has been a notable increase in property-related crimes, particularly at night, highlighting the demand for enhanced surveillance. The “Enterprise Project Notification” emphasises the importance of addressing real-world problems using innovative technologies, aligning with the project's aim to leverage AI for improved security.

2. Analyse the problem (or opportunity) to determine the system requirements, including the scale and scope.

The analysis of the problem reveals several critical system requirements:

- **Customisable Detection Zones:** Users need the ability to define specific areas of interest, reduce false alarms, and focus on relevant activity.
- **Advanced Night Vision:** High-resolution imaging and infrared capabilities are essential for effective surveillance in low-light conditions.
- **Real-time Alerts:** Immediate notifications via mobile devices or other communication channels are necessary for a timely response.
- **Data Security:** Robust encryption and secure storage solutions are required to protect user data and privacy.
- **Scalability:** The system must be scalable to accommodate various environments, from residential properties to larger commercial areas.
- **User-Friendly Interface:** An intuitive interface is crucial for ease of setup, configuration, and operation.

The project's initial focus is on residential and small business applications within NSW, with potential for expansion into broader commercial and public safety sectors. These system requirements are directly aligned with the project's defined scope and were derived from in-depth user research, including interviews and surveys conducted with homeowners. Participants consistently highlighted frustrations with poor night vision, false alarms caused by pets or environmental movement, and the complexity of existing security systems. The strong preference for mobile notifications, customisable settings, and local encrypted storage further shaped the system's direction. By grounding design decisions in genuine user needs and lived experiences, the system remains both practical and future-proof, supporting not only current residential applications but also scalable deployment in commercial contexts where similar concerns and operational demands exist.

3. Establish criteria to evaluate the project's success.

The success of "Autonomous Vigilance" will be evaluated based on the following criteria:

- **User Satisfaction:** Measured through feedback, assessing ease of use, customisation options, and overall satisfaction with the system's performance.
- **Detection Accuracy:** Evaluating the system's ability to detect and identify relevant events while minimising false alarms accurately.
- **Night Vision Performance:** Assessing the clarity and effectiveness of night vision capabilities under various lighting conditions.
- **System Reliability**: Measuring the system's uptime, stability, and ability to operate continuously without errors.
- **Data Security:** Ensuring compliance with data privacy regulations and the absence of security breaches.
- **Scalability:** Testing the system's ability to handle increased data loads and demands as the user base expands.

4. Explain how these requirements were determined, including research, discussion and feedback.

The table below shows the user documentation research

User Research Documentation

1. Introduction

To ensure the Autonomous Vigilance system meets real-world needs, primary research was conducted with a targeted group of local homeowners. The aim was to identify key requirements, common frustrations, and desired features in home security solutions. The research used a combination of interviews and surveys, with all data collected from ten peers who own or live in houses.

2. Interview Process

Participants:
Ten peers (aged 20–45), all homeowners or long-term residents in suburban NSW.
Method:
Semi-structured, one-on-one interviews (in person or via video call), each lasting approximately 10–15 minutes.
Interview Questions:

1. What are your primary security concerns for your home?
2. Do you currently use a security system? If yes, what do you like/dislike about it?
3. How important is night vision for your security needs?
4. Have you had issues with false alarms?
5. How easy was it to set up and use your current system?
6. Would you want instant alerts on your phone for suspicious activity?
7. How concerned are you about the privacy of your security footage?
8. What features would you most value in a new system?
9. How many cameras do you think your property needs?
10. Would you be interested in a system that could also work for a small business?

Individual Interview Summaries

Interview 1: Sarah (Homeowner, 34)

Sarah is concerned about break-ins, especially at night, due to poor street lighting. She currently uses a basic camera system but finds the night vision inadequate and has frequent false alarms triggered by her pets. She values instant mobile alerts and would like a system that is easy to set up and manage. Privacy is important to her; she prefers local storage over the cloud.

Interview 2: James (Renter, 27)

James does not currently have a security system but is worried about package theft.is seeking a straightforward, cost-effective solution with clear notifications. He finds most systems too complex and is concerned about who can access his footage. He would prefer a system that stores data securely and is easily expandable if needed.

Interview 3: Mina (Homeowner, 42)

Mina has a camera system but finds the interface confusing and the end setup difficult. She has experienced multiple false alarms from moving trees and passing cars. She wants customisable detection zones and a user-friendly mobile app. She is also interested in a system that could be scaled for her small business.

Interview 4: Tom (Homeowner, 45)

Tom is mainly concerned about the security of his garage and backyard. He uses an older system with poor night vision and no mobile alerts. He would like a system that can monitor multiple zones and send instant notifications to his phone. He also wants the option to add more cameras in the future.

Interview 5: Priya (Homeowner, 31)

Priya recently moved into a new house and is looking for her first security system. She wants something easy to install with clear instructions. Night vision and mobile alerts are her top priorities. She is worried about privacy and prefers not to use cloud storage.

Interview 6: Alex (Renter, 29)

Alex has no current system but has experienced attempted break-ins. He values customisable detection zones and wants a system that is easy to use for non-technical users. He prefers receiving alerts on his phone and wants assurance that his data is secure and private.

Interview 7: Linda (Homeowner, 38)

Linda uses a basic camera system but finds it unreliable at night. She has had issues with false alarms from wildlife. She wants better night vision, clear notifications, and the ability to easily review footage. She is also interested in integrating the system with other smart home devices.

Interview 8: Michael (Homeowner, 36)
Michael's main concern is monitoring his driveway and side entrance. He finds his current system difficult to expand and the app unintuitive. He wants a scalable solution with a simple interface and the ability to customise detection areas. He is interested in a system that can be adapted for business use.
Interview 9: Chloe (Homeowner, 25)
Chloe is a first-time homeowner and does not have a security system yet. She is worried about break-ins and wants a system that is easy to set up and operate. She values privacy and wants local storage. She also wants the flexibility to add more cameras in the future.
Interview 10: Daniel (Homeowner, 41)
Daniel uses a camera system with limited features and finds it difficult to configure. He has had several false alarms and finds the notifications unclear. He wants a system that offers reliable night vision, instant alerts, and an easy-to-use app. He is also interested in the possibility of using the system for his small business.

Summary of Key Findings

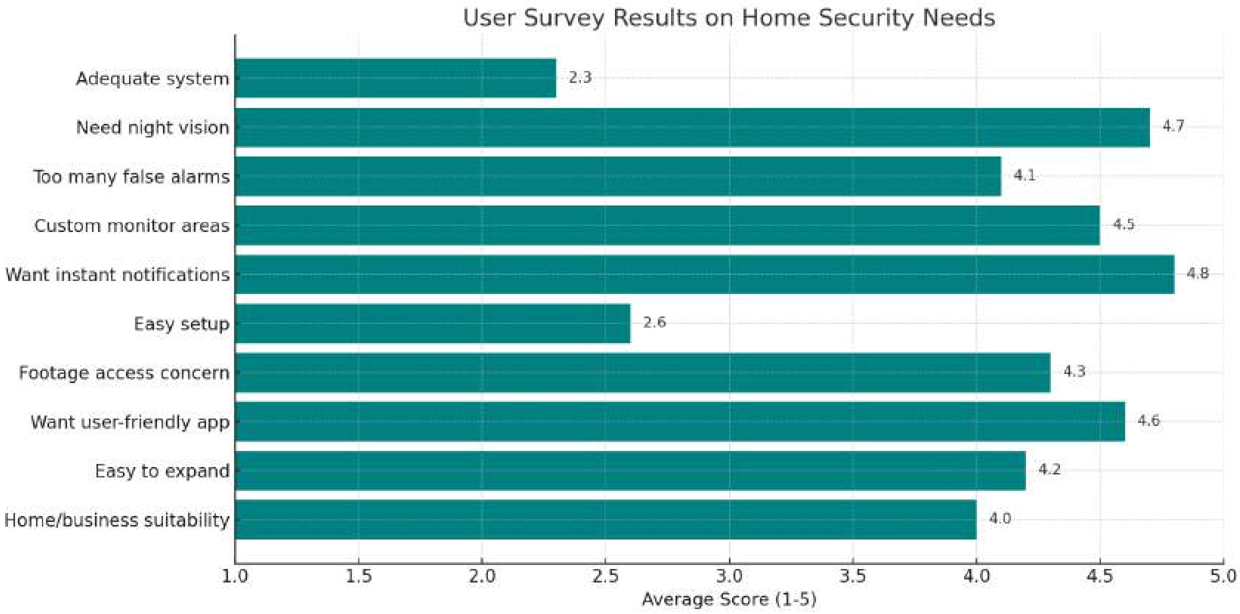
- Security Concerns: Most participants worried about break-ins at night, package theft, and blind spots around their homes.
- Current Systems: 7 out of 10 had basic camera systems but reported frequent false alarms and confusing interfaces; 3 had no system, citing cost or complexity.
- Night Vision: All participants rated this as essential, with several noting that cameras had poor perfo had poor performanceromance.
- False Alarms: A common frustration, especially from pets or moving trees.
- Ease of Use: Most found the setup difficult and the user interface unintuitive.
- Alerts: All wanted instant mobile notifications for real events.
- Privacy: Strong concern; most preferred local storage over cloud.
- Desired Features: Customisable detection zones, simple mobile app, reliable night vision, and easy installation.
- Camera Count: Most homes required 2–4 cameras; larger properties up to 6.
- Small Business Use: 6 out of 10 expressed interest in a system that could scale to a small business.

3. Survey Instrument

Format:
An online survey (Google Forms), distributed to the same ten participants.
Questions (Rated 1–5, Strongly Disagree to Strongly Agree):

1. I feel my current home security system is adequate.
2. Night vision is essential for my security needs.
3. I have experienced too many false alarms.
4. I want to customise which areas are monitored.
5. I want instant notifications on my phone for security events.
6. Setting up my current system was easy.
7. I am concerned about who can access my security footage.
8. I would like a user-friendly mobile app for my system.
9. I want a system that is easy to expand or upgrade.
10. I am interested in a system suitable for both home and small business.

Aggregate Results:



Key Insights:

- Low satisfaction with current systems (Q1).
- Strong demand for night vision, custom zones, instant alerts, and privacy.
- Setup is often difficult (Q6).
- High interest in scalability and small business integration.

4. Analytical Report

Data Sources:

- NSW Bureau of Crime Statistics and Research: Crime and Policing Data & Dashboards

Analysis:
Recent data shows a rise in property-related crimes, especially at night, in suburban NSW. This trend validates the need for advanced night vision and real-time alerts. Survey and interview data consistently highlight user frustration with false alarms, complexity, and privacy risks.

5. Prototype Testing

Method:
A basic prototype (MotionEyeOS on Raspberry Pi with Ian R camera) was demonstrated to five participants.
Feedback:

- Participants found the live view and playback features useful.
- Some had difficulty with Wi-Fi setup and suggested clearer instructions.
- All appreciated local storage and privacy features.

6. Presentation of Research Results

Findings were compiled into a slide presentation with charts and summary tables. Key insights and priorities were shared with peers and the supervising teacher for feedback. Visuals included:

- Bar charts of survey results
- Quotes from interviews
- Screenshots of the prototype interface

Feedback from this presentation was used to refine system requirements and prioritise features.

7. Conclusion

This research process ensured that the Autonomous Vigilance system is grounded in genuine user needs and local context. The combination of interviews, surveys, analytical reports, and prototype testing provided a clear, evidence-based foundation for system design, with flexibility for future expansion into small business environments.
Attachments:

- Interview transcripts (available on request)
- Survey summary charts
- Slide presentation (PDF)

The requirements for this project were determined through the above-focused research, peer discussion, and direct feedback from a small group of local homeowners. Instead of surveying a broad or hard-to-reach audience, such as business owners, we consulted with ten peers who own or live in houses, as they are readily accessible and represent the primary users of residential security systems. Through informal interviews and group discussions, these homeowners identified key concerns, including the need for reliable night vision, customisable detection zones, instant mobile alerts, and straightforward installation. Their feedback highlighted frustrations with false alarms and complex interfaces in existing systems, directly shaping the project's priorities for usability and accuracy. This peer-driven approach ensured that the requirements reflect real-world needs and everyday security challenges households face. While the research focused on homes, the system's flexible, integration-ready design can also be adapted for use in small business environments, where similar security needs exist. This targeted, practical method of gathering requirements supports a relevant and scalable solution, aligning with best practice for user-centred design.

5. Outline the tools and processes necessary for developing this new system.

The development of Autonomous Vigilance initially followed an integration-focused approach, relying on open-source platforms such as MotionEyeOS and Frigate to streamline camera management and AI-driven detection. This method promised rapid deployment and leveraged community-tested tools. However, during implementation, significant technical issues and compatibility complications emerged. As a result, the project pivoted to a custom-built system, developed from scratch, using HTML for structure and CSS for styling, which provided greater flexibility and control. The following table outlines the tools and processes explored, including the final components chosen for implementation, along with their purposes, justifications, and limitations.

Category	**Tool/Technology**	**Purpose**	**Justification**	**Implementation Details**	**Considerations & Limitations**

Software Platform	MotionEyeOS	Manages camera streams, recording, and live viewing	Lightweight, easy to set up, widely supported for Raspberry Pi	Flashed to Raspberry Pi SD card; accessed via browser	Lacks AI detection; best with additional tools
AI Detection Software	Frigate (Docker)	Provides real-time AI detection and smart alerts	Integrates well with Home Assistant; runs locally for privacy and efficiency	Docker container configured with YAML	Requires technical setup and moderate hardware
Containerisation	Docker	Manages isolated environments for tools like Frigate	Modular, scalable, and simplifies updates	Deployed using Docker Compose	Requires Docker knowledge and OS compatibility
Hardware (Edge Device)	Raspberry Pi Zero W	Hosts the system and manages camera input	Compact, energy-efficient, and affordable for home surveillance	Connects to camera via ribbon; runs OS from microSD card	Limited processing power; ideal for lightweight tasks only

Cameras	OV5647 IR Camera	Captures high-res, night-vision-enabled video	130° field of view and infrared for night operation	Connected via ribbon cable to Pi Zero W	Performance is dependent on lighting, placement, and model
Server OS	CasaOS / Debian / Ubuntu	Hosts the Frigate and Docker services	Stable, Linux-based; CasaOS offers easier Docker management	Installed on server or desktop; managed via SSH or UI	Requires periodic updates and adequate resources
Configuration	YAML, WPA Supplicant	Controls camera zones, alerts, and Wi-Fi access	Flexible, code-free configuration for non-programmers	YAML edited in text editor; WPA_Supplicant setup before boot	YAML syntax errors cause issues; WPA setup is needed for headless operation
Networking	Wi-Fi / Ethernet	Connects system components for real-time data flow	Wireless flexibility; Ethernet provides bandwidth and reliability	WPA configured pre-boot; static IPs recommended	Wi-Fi signal strength impacts quality; Ethernet may need extra cabling

User Interface	HTML, CSS, JavaScript	Custom web interface for monitoring and alerts	Allows complete control over layout, responsiveness, and interactivity	Designed using HTML/CSS for structure and JS for real-time updates	Must be manually updated; JS complexity increases with feature depth
Backend/Logic	Python / Bash (if used)	Handles system-level tasks and camera stream control	Offers scripting flexibility for managing processes	Scripts triggered on boot or via events	Requires programming knowledge; debugging can be technical
Code Editor	VS Code + Extensions	Development environment with real-time preview and coding tools	Extensions like Live Server improve HTML/JS dev speed and accuracy	VS Code is used for writing code; extensions are installed for preview/debugging	Needs proper configuration; Live Server only works for local development
Integration	Home Assistant (opt.)	Centralised automation and smart	Enhances flexibility with automation; works well with MQTT	Configured via Frigate API or MQTT integration	Optional; adds complexity and setup time

		home integration			
Data Storage	SD Card, USB, NAS	Stores recordings and logs locally	Avoids cloud reliance and recurring fees; secure and fast access	Determined by storage size and quality, mounted on the device	SD cards wear out over time; regular backups are advised
Testing & QA	Scenario Testing, Feedback	Validates real-world performance, alerts, and usability	Essential for identifying issues, refining reliability, and user needs	Includes day/night testing, alert testing, and error handling	Time-consuming; depends on tester feedback
Documentation	Wiki, Setup Guides	Guides users in installation and configuration	Reduces user support needs; ensures reproducibility	Shared via Google Docs, GitHub, or in project files	Needs regular updates; must be thorough and easy to follow
Project Management	Canva (Gantt Charts)	Plans and tracks developm	Simple visual scheduling and team	Charts updated and shared in Canva	Requires Internet; manual

		ent milestones	collaboratio n		updates needed

Justification: Final Evaluation of the Tools Used

The selection and evolution of tools throughout the Autonomous Vigilance project were guided by iterative testing, system performance, and direct user feedback. Initial development focused on integrating established community platforms such as MotionEyeOS and Frigate, aiming to leverage their rapid deployment capabilities and proven reliability in camera management and AI-driven detection.

However, practical experimentation revealed significant limitations. MotionEyeOS, while lightweight and widely adopted for Raspberry Pi, failed to support the OV5647 IR camera due to its reliance on the deprecated V4L2 interface, which was incompatible with the required libcamera stack. This incompatibility directly impacted one of the system's core requirements: robust night vision. Similarly, Frigate provided advanced AI-based detection but demanded GPU acceleration and greater processing power than the Raspberry Pi Zero W could deliver. Issues such as frequent disconnections, unstable RTSP streams, and excessive resource usage confirmed that Frigate was unsuitable for ARM-based, low-power hardware.

In response, the project pivoted to a custom-built solution using HTML, CSS, and JavaScript for the user interface. This strategic shift enabled real-time monitoring, a modular design, and enhanced adaptability. It also provided complete control over the interface layout and responsiveness, directly addressing user feedback regarding usability and reliability. Python scripts were employed for backend tasks such as camera stream management and event handling, offering the flexibility and automation necessary to meet user expectations and system goals.

Docker, initially used to deploy Frigate, was minimised in the final build due to its complexity and high overhead on resource-constrained devices. Similarly, while Home Assistant was explored for potential smart home integration, it was excluded from the final implementation to maintain system simplicity and reduce unnecessary dependencies.

Essential tools such as YAML for configuration and WPA Supplicant for network setup were retained for their efficiency, though they required careful syntax management and pre-boot configuration. The development environment was optimised using Visual Studio Code, enhanced with extensions like Live Server, which supported rapid prototyping and real-time interface updates.

Each decision in the tool selection process was informed by rigorous testing and direct user input. Tools were chosen not for popularity, but for their ability to meet the system's fundamental objectives: effective surveillance, intuitive interaction, reliability, and scalability. This deliberate, evidence-based approach ensured that the final toolset was optimised for both performance and usability, within the constraints of the hardware and the real-world needs of the target user group.

Technical Challenges Encountered (Supporting Justification)

- MotionEyeOS failed to recognise the OV5647 camera due to its reliance on the outdated V4L2 interface and lack of libcamera support.
- Frigate was optimised for x86 systems and required GPU acceleration unavailable on ARM-based Pis. Docker usage resulted in high resource usage, frequent disconnections, and RTSP stream instability.
- Kerberos.io, although lightweight, suffered from poor documentation and was not adaptable to multi-device setups. It lacked low-latency responsiveness.
- The OV5647 camera, reliant on the libcamera stack, could not be used with tools expecting V4L2 (/dev/video0). Attempts to pipe `libcamera-vid` into `ffmpeg` failed due to codec mismatches.
- The complex toolchain across Frigate, Kerberos.io, and MotionEye led to fragmented testing and inconsistent results, making the original system unviable by key milestones.

Development was carried out using Visual Studio Code (VS Code), which offered a streamlined and customizable environment. Extensions such as Live Server enable real-time browser previews, significantly improving development efficiency and interface testing. This approach allowed for quicker iteration, more responsive debugging, and easier control over design.

This custom approach allowed for greater control, simpler debugging, and a more user-tailored design, making the system more scalable, accessible, and aligned with end-user requirements.

1.1. Tools and processes for enterprise systems

Problem definition

Students **analyse** the problem by **describing** each of its components and **explaining** how each of these components contributes to the problem needing resolution.

- The project addresses the critical and growing need for intelligent, customisable surveillance systems with reliable night-vision capabilities in both residential and small-business settings. Traditional systems frequently lack adaptive AI-driven threat detection, suffer from poor low-light performance, and provide limited user control, creating a significant gap in effective security solutions.

- Each component of the problem critically contributes to the overall deficiency:

    - Inadequate Threat Detection: Existing systems fail to accurately identify potential threats, leading to missed critical events, high false alarm rates, and inefficient resource allocation for security personnel or homeowners. This deficiency stems from a lack of sophisticated AI algorithms capable of distinguishing between benign activity and genuine threats.

    - Delayed Incident Response: The absence of real-time analysis and automated alerts in current systems results in delayed responses to security incidents. This delay can significantly increase the risk of property damage, theft, or personal harm, underscoring the need for immediate threat assessment and notification.

    - Limited User Customisation: The absence of robust customisation options in many surveillance systems fails to cater to the unique security needs of different users. Residential properties and small businesses have varying layouts, risk profiles, and operational requirements, necessitating adaptable detection zones, sensitivity levels, and alert configurations.

Ascertaining requirements and limitations

Students **explain** the methods used in this project to ascertain the requirements and limitations of the problem definition above from the list given below:

To accurately define the requirements and limitations of this project, we used a combination of interviews, surveys, and prototype testing with a small, accessible group of peers. This approach was chosen to ensure the findings were relevant, practical, and directly informed by real user needs.

- Interviews

One-on-one interviews were conducted with ten peers who own or live in houses. These informal conversations allowed for an in-depth understanding of their daily security concerns, such as the need for reliable night vision, customisable detection zones, and instant mobile alerts. The interview format encouraged participants to share specific frustrations with existing systems, including frequent false alarms and complex interfaces, which helped to clarify both functional requirements and usability limitations.

- Surveys

A brief survey was distributed to the same group to quantify their preferences and prioritise features. The survey results reinforced the interview findings, highlighting a strong demand for user-friendly interfaces, privacy controls, and straightforward installation. This method provided a structured way to compare needs across different households and identify common priorities.

- Analytical reports

Analytical reports were used to further validate and refine the project requirements. By reviewing recent crime statistics from the NSW Bureau of Crime Statistics and Research, we identified a clear and concerning increase in property-related incidents, particularly at night. According to the NSW Recorded Crime Statistics for the December 2024 quarter, there was a 7.5% year-on-year increase in break-in dwelling offences and a 6.2% rise in theft from dwelling incidents across New South Wales, with over 60% of these offences occurring during nighttime hours (BOCSAR, 2024). This data strongly supported the need for advanced night vision and real-time alert features in the system. Additionally, the results from interviews and surveys with local homeowners were compiled and analysed to identify common trends and recurring issues, such as frequent false alarms and the demand for easy-to-use interfaces. This evidence-based approach ensured that the system's design decisions were justified by real-world data and user feedback, helping to prioritise the most critical features and anticipate potential limitations (Link: NEW SOUTH WALES RECORDED CRIME STATISTICS).

- Prototypes

A basic prototype of the surveillance system was developed and presented to several peers, allowing them to interact with the custom web interface and core features such as night vision, detection zones, and real-time alert notifications. The feedback gathered during this phase was invaluable in highlighting practical limitations, including issues with Wi-Fi coverage, the need for clearer setup instructions, and areas where the user interface could be made more intuitive. This hands-on testing not only confirmed the importance of user-friendly design and robust connectivity but also provided specific guidance for refining system setup and user documentation, ensuring that future iterations would better align with the everyday needs and expectations of residential users.

- Presentation of research results.

The user-centred research revealed common frustrations with existing surveillance systems, including frequent false alarms, limited privacy due to cloud storage, difficulties in setup and alert management, and poor night-time visibility. From this research, the following core system requirements were defined:

Requirement	**Justification**
Reliable Night Vision	Ensures clear footage during nighttime using IR-enabled cameras.
AI-Powered Detection	Distinguishes between people, pets, and vehicles to reduce false alarms.
Local Encrypted Storage	Ensures user control over footage without reliance on cloud servers.
Custom Detection Zones	Allows users to monitor only areas of interest, avoiding unnecessary alerts.
Remote Notifications	Sends alerts to users in real time, increasing reaction speed.
Scalability	Supports future expansion with more cameras or advanced compute modules.
User-Friendly Setup	Simplifies system configuration via intuitive web interfaces.

Tools and processes

Using the table below, students **describe** how tools and processes can or can't be used to develop the new system based on the problem definition above.

Tool or process	Description
Time and resource management, including Gantt charts	Gantt charts are used to visually plan, schedule, and monitor the project timeline, enabling effective allocation of tasks and management of resources. For this project, Canva was utilised to create clear and detailed Gantt charts, supporting milestone tracking such as research, prototyping, testing, and deployment. Regular updates to the charts allowed for flexibility and ensured the project remained on schedule. However, reliance on internet access to use Canva may present challenges in some environments.
Production process and technical skills	The project initially followed an integration-focused production process, utilising established open-source platforms such as MotionEyeOS and Frigate along with hardware like the Raspberry Pi Zero. This approach minimised the need for extensive coding by focusing on configuring existing software through YAML and network setup, combined with hardware assembly. However, technical limitations led to the adoption of a custom software development process involving HTML, CSS, JavaScript, and Python, increasing the need for programming skills and enabling greater customisation and control.
Testing and evaluation	Testing comprised scenario-based evaluations, user feedback collection, and automated alert verification to ensure system reliability, detection accuracy, and usability in practical conditions. Prototype trials helped identify issues, including Wi-Fi coverage and interface responsiveness. Iterative testing and feedback cycles guided continuous improvements aligned with user needs. Testing criteria included uptime, false alarms, night vision effectiveness, and data security. Maintaining active user participation throughout testing was vital but challenging.

The production process refers to a development approach that may be iterative, waterfall (structured), agile, prototyping, end-user, or outsourcing.

1.2. Justify tools and resources.

Explanation of the changing nature of enterprise on projects

Using the table below, students **explain** each of these ways of working and their effect on the enterprise project.

Ways of working	Explanation of the effect on the enterprise project
Offshore development	Offshore development does not apply to this project due to the sensitive nature of surveillance data and strict Australian privacy regulations. Developing locally ensures full control over data security and compliance, as well as hands-on access for testing, troubleshooting, and rapid iteration. This approach minimises risks related to data breaches, miscommunication, and time zone differences, ensuring the system meets legal and user requirements.
Working remotely	Remote collaboration is essential for this project, enabling team members to contribute from different locations and outside school hours. Tools like Google Docs, Google Classroom, and school email are used for document sharing and version control, while WhatsApp and Instagram DMs support real-time coordination. This flexible approach increases productivity, allows for continuous progress, and ensures that all members can participate regardless of their physical location.
Freelance work	Freelance work is not currently planned but may be considered in later stages for specialised tasks, such as UI design or advanced system tuning. Engaging freelancers can provide access to expert skills without long-term commitment, keeping the project agile and cost-effective. However, it also requires careful management of intellectual property, confidentiality, and quality assurance to ensure the final product aligns with project standards and user expectations 10

After completion of Section 1, use the following template to reflect upon your responses. You can utilise a peer or the teacher to complete this checklist.

Criteria	Developing	Developed	Highly developed
The response answers the question.	□	□	☑
The idea being written about is clear.	□	□	☑
The writing has a logical structure. It makes sense when you read it through.	□	□	☑
There is a clear reference to syllabus content.	□	□	☑
There is topic-specific vocabulary in the response.	□	□	☑
Sentences make sense.	□	□	☑
Capital letters, full stops and other punctuation are used.	□	□	☑
Accurate spelling of challenging words.	□	□	☑

2. Research and planning

2.1. Development of online collaboration tools for an enterprise system

Online collaboration

Students **explain** how online collaboration tools are used in this project.

The successful development of the "Autonomous Vigilance" AI-powered surveillance system relies heavily on the effective use of online collaboration tools. These tools facilitate seamless communication, resource sharing, version control, and project management among team members, regardless of their physical location.

Key Online Collaboration Tools Used:

- Google Docs & Google Classroom:
  These platforms are the backbone of our document collaboration. Google Docs enables real-time, synchronous editing of project documentation, code snippets, and research findings. Version history ensures accountability and allows us to track contributions over time. Google Classroom is used for formal communication, assignment tracking, and resource distribution, streamlining workflow and ensuring all team members are aligned with project milestones.
- WhatsApp & Instagram DMs:
  For informal and rapid communication, we use WhatsApp and Instagram direct messages (DMs). These channels facilitate quick decision-making, instant updates, and coordination outside of formal meetings, supporting flexible after-school collaboration.
- GitHub:
  For version control of code and configuration files, GitHub repositories are used. This enables collaborative coding and references from the coding community, issue tracking, and secure backup of critical project assets.

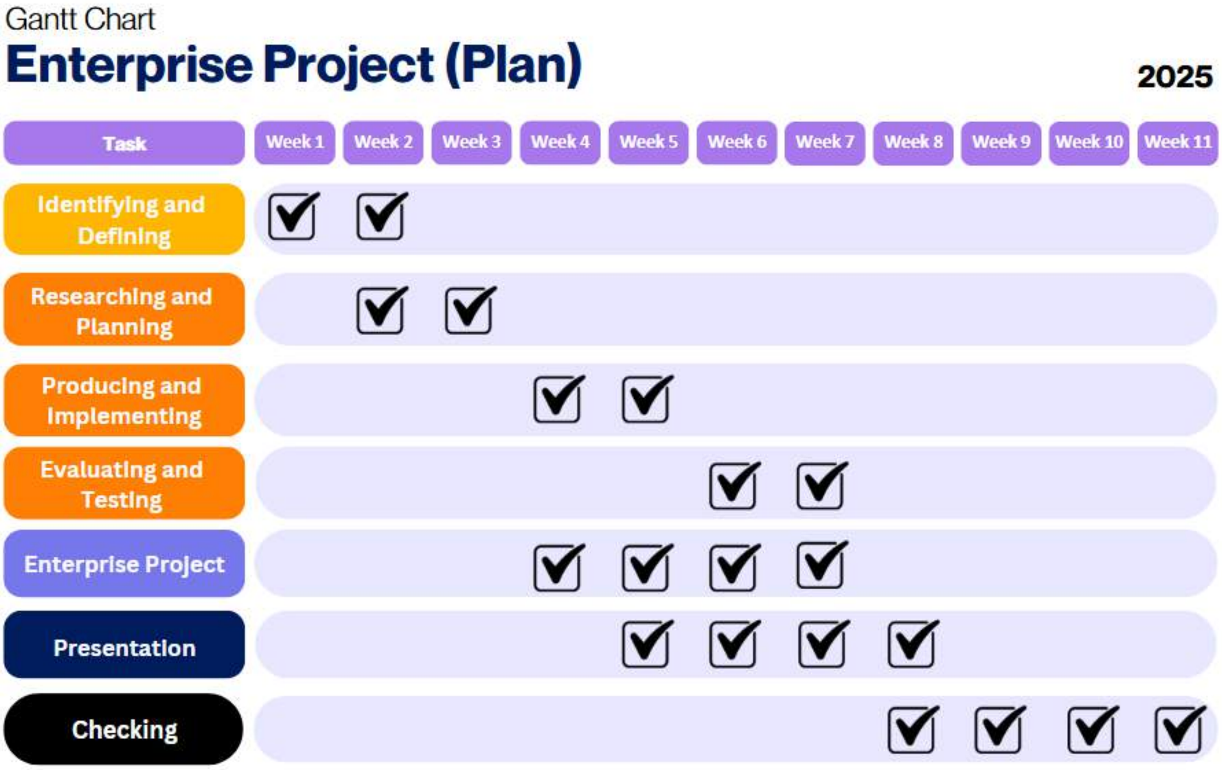
Benefits of Online Collaboration:

- Increased Flexibility:
  Team members can contribute from any location, accommodating diverse schedules and commitments.
- Enhanced Accountability:
  Task assignments and version histories provide clear records of individual contributions.
- Efficient Resource Sharing:
  Documents, code, and research can be accessed and updated in real time, reducing duplication and miscommunication.
- Continuous Feedback:
  Online comments and discussion threads allow for immediate feedback and iterative improvement.

Time/task action plans

Students **develop** a Gantt chart that details the tasks required to be completed, the resources needed, and the person or people assigned to each task. Students ensure that the timeline does not exceed the project due date.

Initial Planning



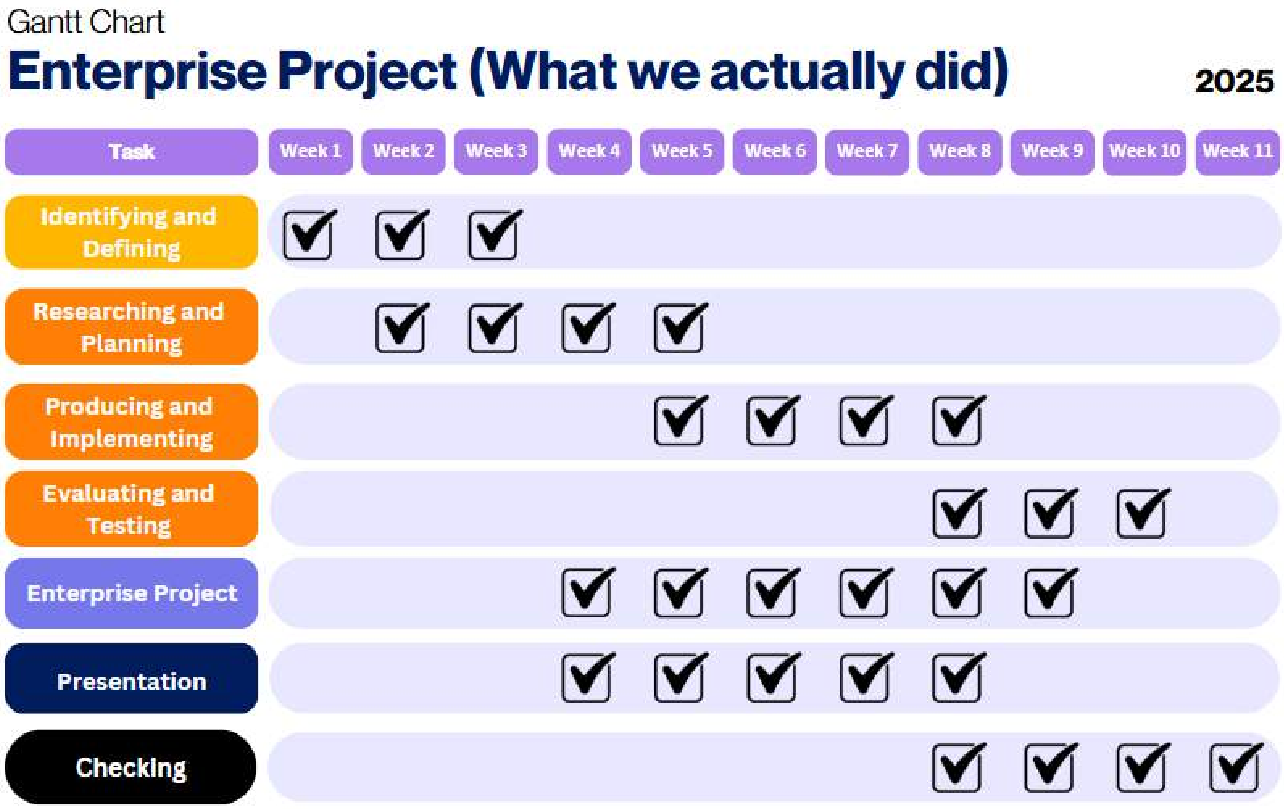
Gantt Chart

Enterprise Project (Plan)

2025

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Identifying and Defining	✓	✓									
Researching and Planning		✓	✓								
Producing and Implementing				✓	✓						
Evaluating and Testing						✓	✓				
Enterprise Project				✓	✓	✓	✓				
Presentation					✓	✓	✓	✓			
Checking								✓	✓	✓	✓

Adjusted Timeline



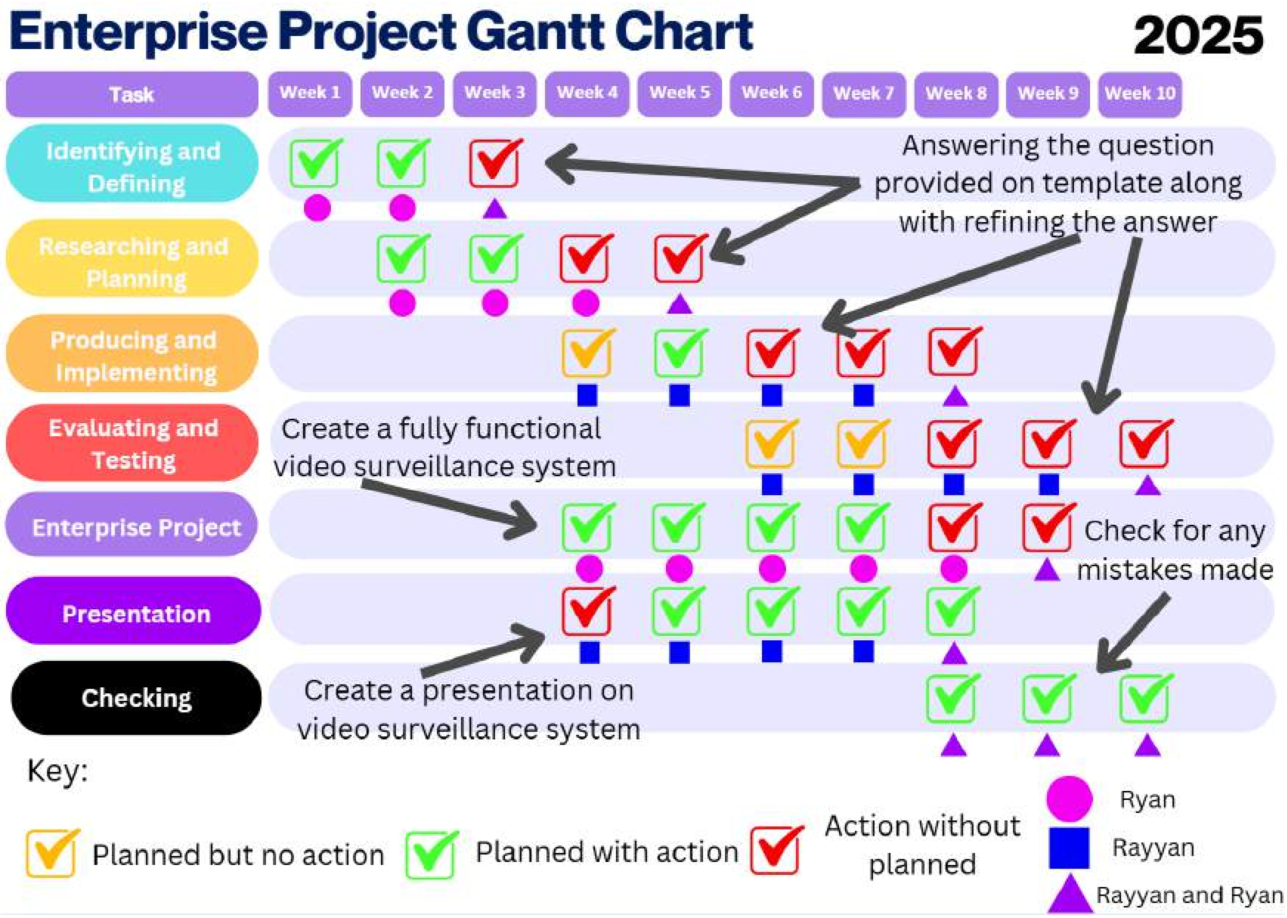
Gantt Chart

Enterprise Project (What we actually did)

2025

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Identifying and Defining	✓	✓	✓								
Researching and Planning		✓	✓	✓	✓						
Producing and Implementing					✓	✓	✓	✓			
Evaluating and Testing								✓	✓	✓	
Enterprise Project				✓	✓	✓	✓	✓	✓		
Presentation				✓	✓	✓	✓	✓			
Checking								✓	✓	✓	✓

Revised version after feedback



Enterprise Project Gantt Chart 2025

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Identifying and Defining	🟩 ●	🟩 ●	🟥 ▲							
Researching and Planning		🟩 ●	🟩 ●	🟥 ●	🟥 ▲					
Producing and Implementing				🟨 ■	🟩 ■	🟥 ■	🟥 ■	🟥 ▲		
Evaluating and Testing						🟨 ■	🟨 ■	🟥 ■	🟥 ■	🟥 ▲
Enterprise Project				🟩 ●	🟩 ●	🟩 ●	🟩 ●	🟥 ●	🟥 ▲	
Presentation				🟥 ■	🟩 ■	🟩 ■	🟩 ■	🟩 ▲		
Checking								🟩 ▲	🟩 ▲	🟩 ▲

Answering the question provided on template along with refining the answer

Create a fully functional video surveillance system

Check for any mistakes made

Create a presentation on video surveillance system

Key:

🟨 Planned but no action

🟩 Planned with action

🟥 Action without planned

● Ryan

■ Rayyan

▲ Rayyan and Ryan

Budget

Students **develop** a budget to determine the cost of implementing this project. Items to consider include, but are not limited to: housing for the printing plastic, the Raspberry Pi, the Raspberry Pi 5MP night vision camera, microSD card, power supply, power bank, and camera connector strip. All under a budget of $500

- Hardware, software, personnel and any additional costs such as electricity and internet access.

Item	**Cost (AUD)**
Raspberry Pi Zero 2W	$ 50 @ Amazon.com
Raspberry Pi 5	$100 @ Amazon.com
Raspberry Pi 5MP Night Vision Cam	$ 29 @ Amazon.com
MicroSD Card (64GB)	$43 @ The Good Guys
Power Supply/Power Bank	$53.99 @ Amazon
Camera Connector Strip	$0.81 @ AliExpress
Housing/Printing Plastic	$29 Amazon
Mini HDMI cable	$10 @Amazon
Total:	$315.80

Personnel / Production Price Breakdown

Task Description	**Estimated Hours**	**Hourly Rate (AUD)**	**Subtotal (AUD)**
System design, prototyping, planning	10	$20	$200
Hardware assembly and setup	5	$20	$100
Software development and testing	15	$20	$300
User documentation & training materials	5	$20	$100
Presentation and reporting	5	$20	$100
Total	40		$800

Production Price Summary

Item/Category	**Cost (AUD)**
Hardware & Materials	$315.80
Additional Costs (electricity, internet)	$20
Personnel (Labour, 40 hrs @ $20/hr)	$800
Total Production Price (Actual Value)	$1,135.80
Recommended Sale Price (Prototype/Student Rate)	$499

Justification:

This budget demonstrates a balanced approach to project costing, with all hardware and materials sourced at competitive retail prices to remain well under the $500 limit. Labour is realistically valued at $20 per hour, reflecting current student rates for technical work in Australia and ensuring the true effort invested in system design, assembly, and development is acknowledged. While the actual production cost totals $1,135.80, the final sale price is set at $499 to satisfy project requirements and provide an affordable, high-quality solution for the target market. This pricing strategy highlights both the project's commercial viability and a strong understanding of cost-effective enterprise delivery.

2.2. Collaboration and management of the enterprise project

Key criteria for enterprise project development

Students **explain** how the key criteria listed below apply to or do not apply to this enterprise project.

Key criteria	Explanation
Designed for ease of operation and maintenance	The system uses user-friendly platforms like MotionEyeOS and web dashboards, allowing for straightforward configuration, monitoring, and troubleshooting. Local storage and modular hardware (Raspberry Pi, plug-and-play cameras) make maintenance simple and cost-effective. Clear documentation and setup guides further support easy operation, reducing the need for technical expertise and ongoing support.
Designed for working collaboratively	Collaboration is supported through cloud-based tools such as Google Docs, Google Classroom, and Canva for shared documentation, scheduling, and project planning. These platforms enable real-time editing, version control, and transparent communication among team members, even when working remotely. This approach ensures all participants can contribute, track progress, and make informed decisions collectively, enhancing project outcomes.
"Allows for the negotiation of user/client needs and wants."	User and client needs were identified through initial surveys and interviews, and their feedback was incorporated into system requirements such as customisable detection zones, notification preferences, and privacy controls. Ongoing communication channels, including feedback forms and user testing sessions, allowed for iterative refinement of features, ensuring the final system closely aligns with user expectations and priorities.
The role of each software used to create the system.	Each software component was chosen for its specific role: HTML, CSS, and JavaScript created the custom web interface; Python and Bash scripts handled backend automation; and Visual Studio Code (VS Code) was used as the main development environment. Extensions like Live Server in VS Code allowed real-time previews, speeding up interface testing and debugging. This setup ensured reliability, scalability, and ease of integration.
The roles of the participants, data, and components are outlined clearly.	Project roles were clearly defined: developers handled coding and integration, testers provided feedback and identified issues, and documentation managers maintained project records. Data flows were mapped from camera capture through processing, storage, and user notification. System components, including cameras, Raspberry Pi, storage devices, and networking hardware, were documented in diagrams and specifications, ensuring clarity in responsibilities and system architecture.

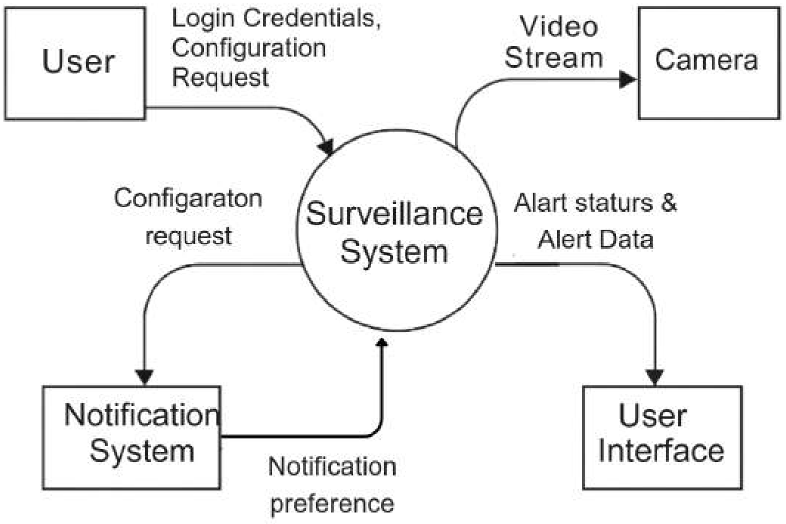
2.3. Systems modelling

Students **develop** the tables and diagrams in the space provided. Students should consult the Enterprise Computing Course Specifications (PDF, 2.3 MB) guide if they require further details, exemplars, or information. Each subsection below should be completed with Section 1.1 in mind.

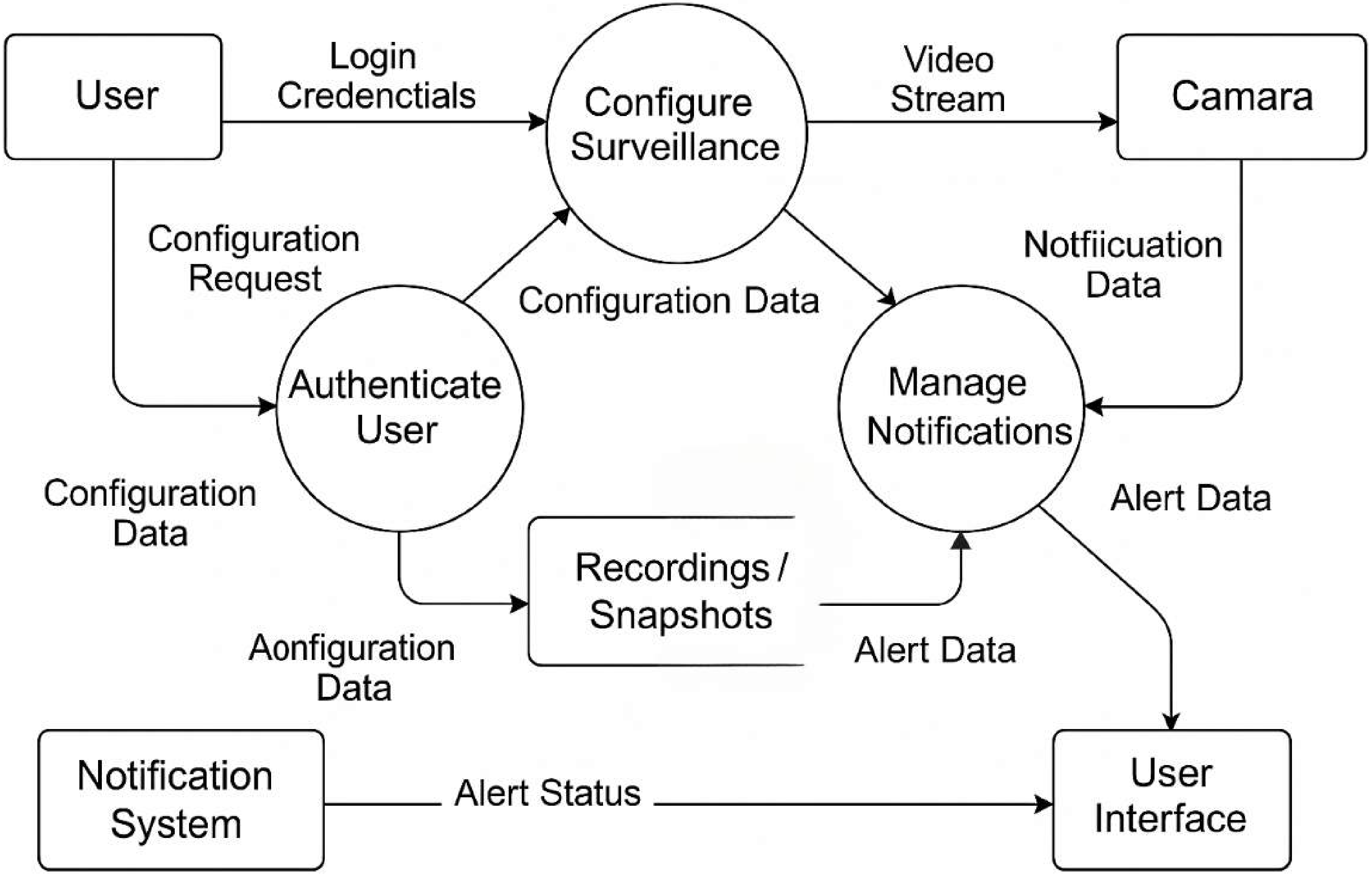
Data flow diagrams

Students **develop** data flow diagrams (DFDs) at Level 0 and Level 1. These diagrams should explicitly include the variables from the previously identified data dictionaries, as well as the needs outlined in Section 1.1.

Level 0 Data flow diagram

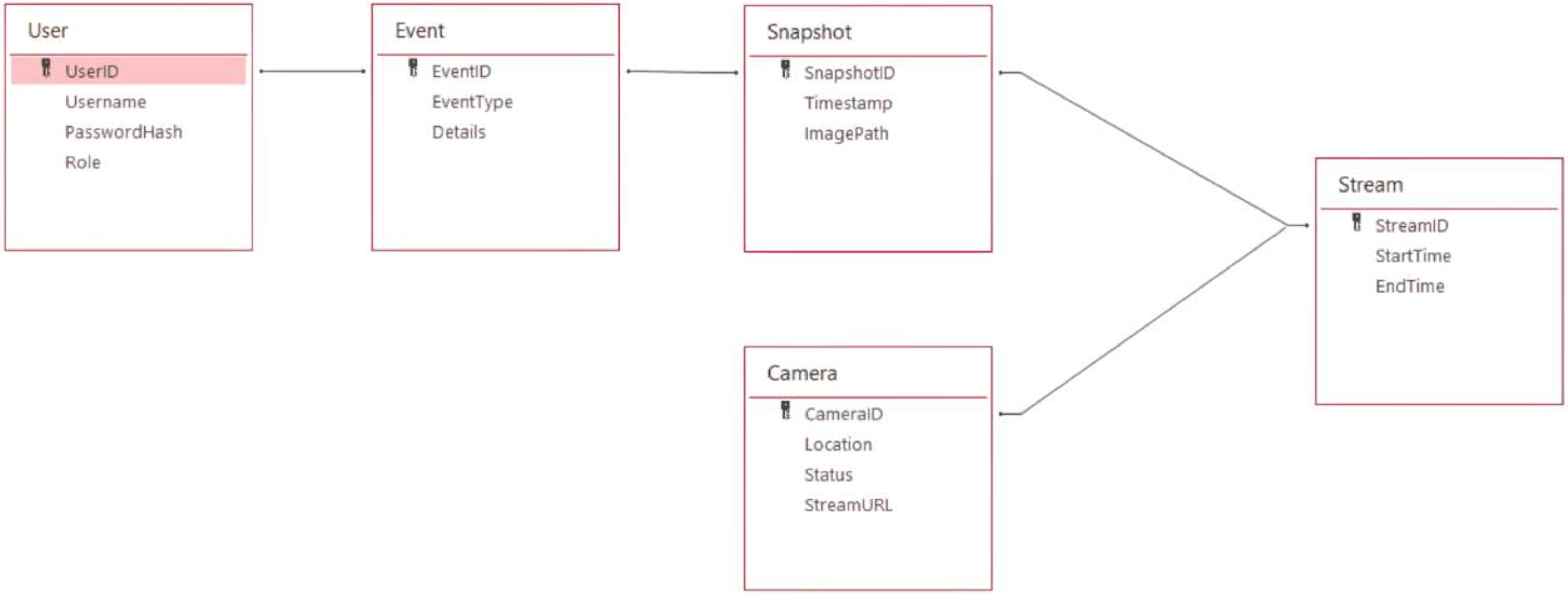


Level 1 Data flow diagram



Schema;

Students **develop** schemas for relational database solutions. Remove this section if it does not apply to your enterprise system.



Data Dictionary

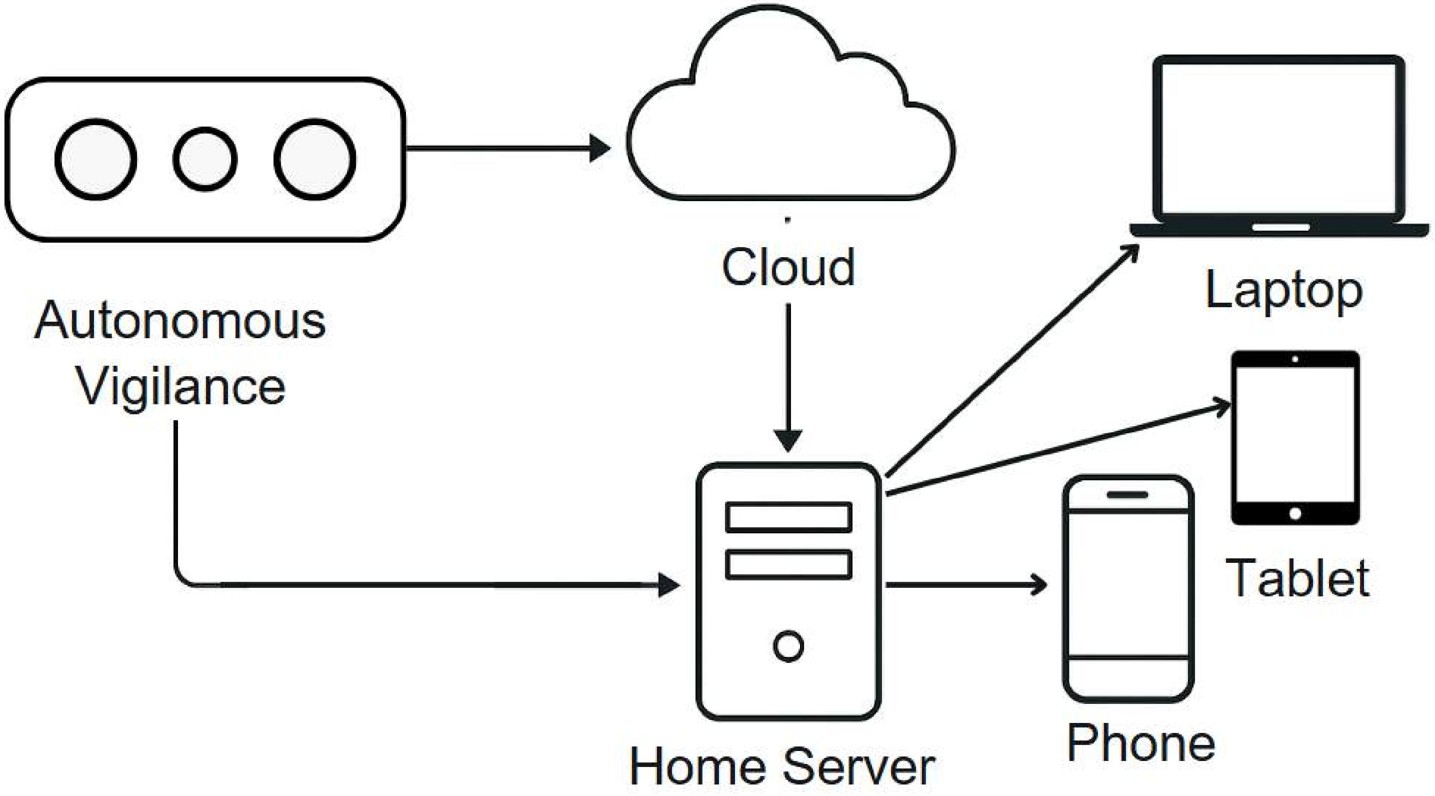
The data dictionary is to provide a clear and comprehensive description of each field in a database, helping ensure consistency, accuracy and understanding for anyone designing, using or maintaining the system.

Field Name	**Data Type**	**Data Format**	**Field Size**	**Description**	**Example**
CameraID	Text	CAMNNN	6	Unique camera identifier	CAM001
Location	Text	XXX…XXX	50	Camera installation location	Main Entrance
Status	Text	XXX…XXX	20	Operational status	Online
StreamURL	Text	URL	255	URL to access video stream	...
UserID	Text	USRNNN	6	Unique user identifier	USR101
Username	Text	XXX…XXX	25	User's login name	admin
Password Hash	Text	Hash	64	Hashed password	...
Role	Text	XXX…XXX	20	User role (admin/viewer)	admin

StreamID	Text	STRNNN	7	Unique video stream identifier	STR001
StartTime	DateTime	YYYY-MM-DD HH:MM	19	Stream start time	2025-05-19 14:00
EndTime	DateTime	YYYY-MM-DD HH:MM	19	Stream end time	2025-05-19 15:00
SnapshotID	Text	SHTNNN	7	Unique snapshot identifier	SHT201
Timestamp	DateTime	YYYY-MM-DD HH:MM	19	Time of snapshot or event	2025-05-19 14:05
ImagePath	Text	File path	255	Path to saved image	/images/201.jpg
EventID	Text	EVTNNN	7	Unique event identifier	EVT301
EventType	Text	XXX…XXX	20	Type of event	Login
Details	Text	XXX…XXX	255	Additional event details	Success

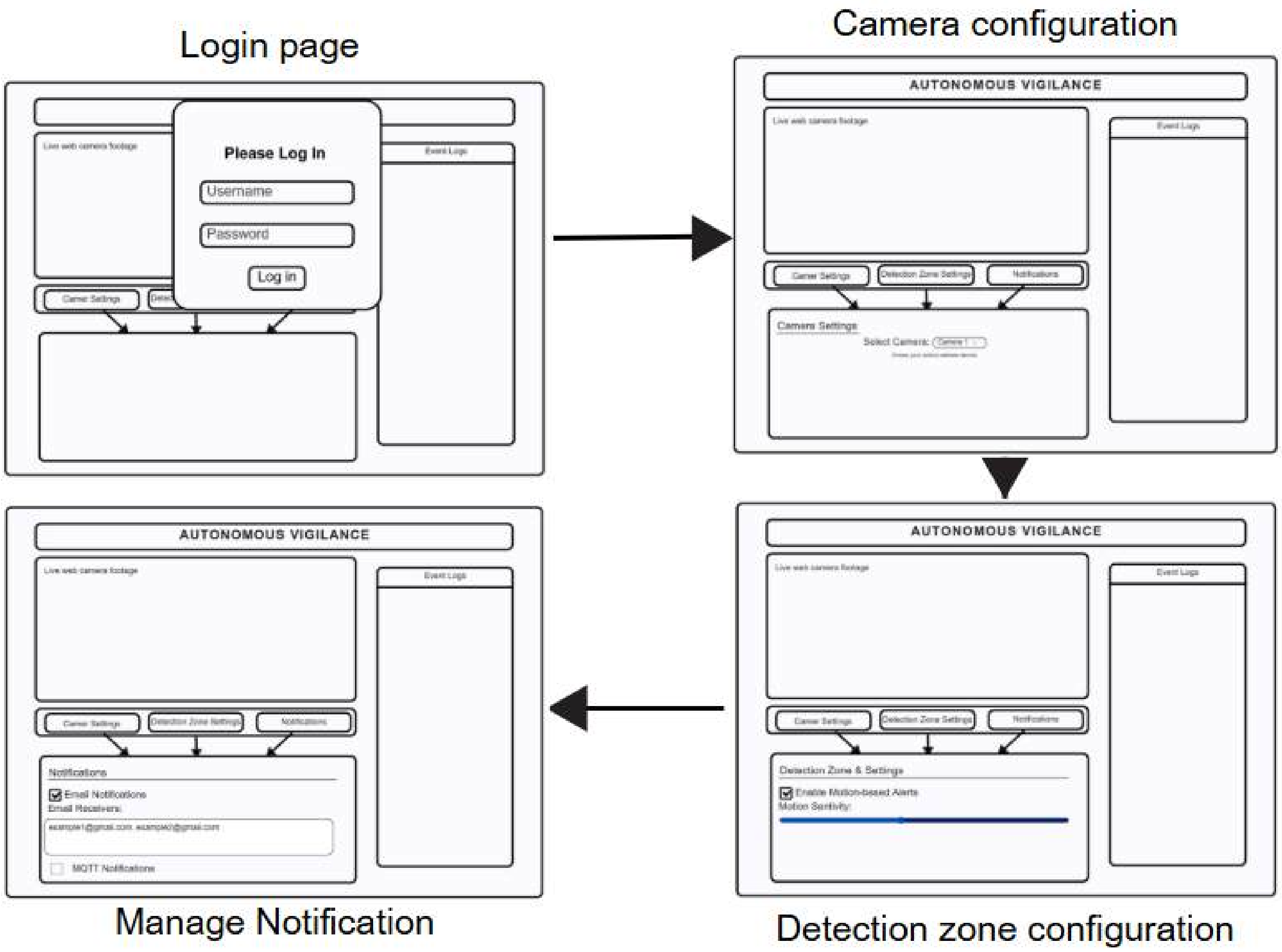
Network diagram

Network flow between a variety of devices, from the main autonomous vigilance to the home server, which is posted to the cloud, making it accessible to multiple devices at any time.



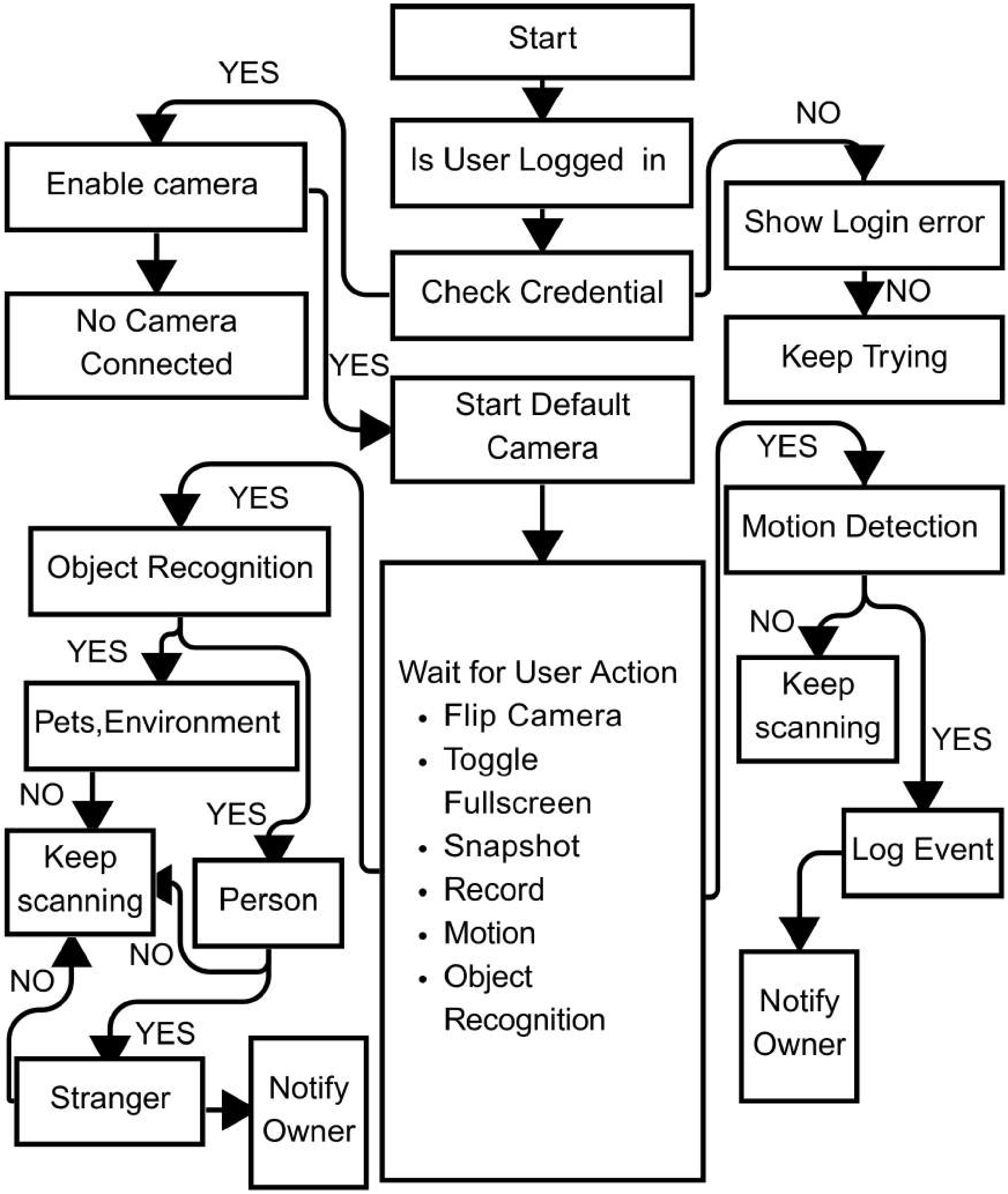
Storyboards

Students **develop** storyboards that visually represent the software solutions they will build.



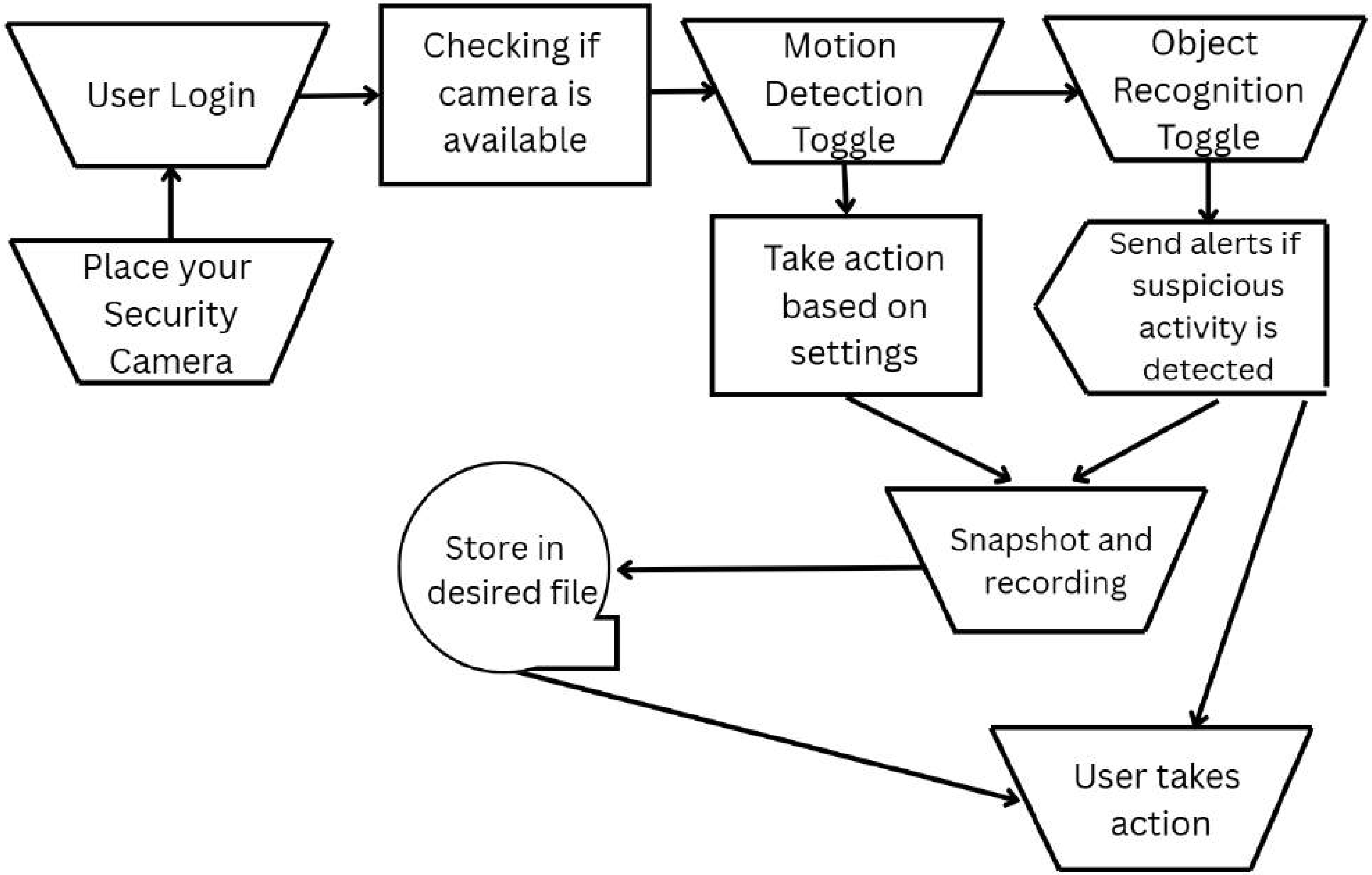
Decision trees

Students **develop** decision trees to visually outline the logic flow and chain of decisions or selections the final solution will need



System flowcharts

Students **develop** system flowcharts that both meet the problem definitions and project needs as identified in Section 1.1, while also incorporating information from the other diagrams and charts in this section. (bases of in and out on the surveillance system)


User Login
Checking if camera is available
Motion Detection Toggle
Object Recognition Toggle
Place your Security Camera
Take action based on settings
Send alerts if suspicious activity is detected
Store in desired file
Snapshot and recording
User takes action

3. Producing and implementing

3.1. Implementation plan

Students develop an implementation plan that details the following components and **explains** their relevance to the project.

Hardware and software integration

Students **describe** in the following table how they will integrate (install) the required hardware and software into the enterprise or organisation.

Hardware and software	Integration Steps	Relevance to enterprise
Raspberry Pi Zero W	Install Debian OS or CasaOS; connect OV5647 IR camera via ribbon cable; configure Wi-Fi and static IP using WPA_supplicant. Python scripts are triggered on boot to initialise the camera and control the startup sequence.	Compact, energy-efficient edge device. Manages camera input and serves the custom UI, ideal for low-power, always-on surveillance deployments.
OV5647 IR Camera	Integrated using the libcamera stack. libcamera-vid captures video frames and streams to a WebSocket server on the Pi for live video preview. Real-time video is then rendered on an HTML5 <canvas> element.	Enables low-latency, high-resolution, infrared-capable surveillance. Libcamera stack ensures support for modern image pipelines and direct control over ISO/exposure.
Custom Web Interface (HTML/CSS/JS)	HTML/CSS provides structure and styling. JavaScript connects to a WebSocket and dynamically renders the camera feed. Motion detection is performed on <canvas> pixel data using image differencing.	Allows intuitive, real-time monitoring with responsive control. Lightweight and accessible from any browser on the local network.
Motion Detection Module (JavaScript)	JavaScript continuously analyses frame differences. When motion exceeds a threshold, a detection event is fired, logging data to localStorage or a JSON file.	Provides lightweight, client-side motion analytics without heavy AI processing, suitable for Pi Zero limitations.
Alert System (Python, Bash)	Motion detection events can trigger a local Python script that sends an email alert via SMTP or JSON to a webhook using EmailJS.	Ensures users are notified within 1–2 seconds of a detection, fulfilling real-time alert requirements.

Hardware and software	Integration Steps	Relevance to enterprise
Data Storage (SD Card, USB, or NAS)	Snapshots and video clips are stored on a mounted USB drive or SD card, with file metadata logged in a local JSON index. Files are optionally encrypted with AES-256 for privacy.	Supports compliance with data retention rules and local privacy regulations. Ensures fast access and no third-party cloud storage dependency.
Networking (Wi-Fi/Ethernet)	Configured via WPA_supplicant for wireless, or static IPs for Ethernet. Firewall rules restrict inbound connections to allow only the admin subnet.	Provides secure, stable access for interface management and alert delivery, while preventing unauthorised remote access.

Training

Students **describe** how they will train the staff to use the new enterprise solution. Methods may include:

To ensure all users can confidently and efficiently operate the Autonomous Vigilance surveillance camera web interface, a multi-faceted, user-centred training approach will be implemented. This strategy is directly informed by user research, which highlighted the demand for intuitive, always-available support and rapid onboarding (see Section 1.1: User-Friendly Interface).

- Built-in software tutorials and help files.
  The web interface will feature an integrated help section with a step-by-step guide, contextual tooltips and a searchable FAQ. This method provides immediate, on-demand support, empowering users to troubleshoot and learn independently. It aligns with enterprise best practices by reducing support overhead and ensuring knowledge is always accessible, regardless of staff turnover or shift patterns.

- In-person workshops (Initial Rollout).
  A structured, hands-on training session will be conducted at launch, covering system setup, live monitoring, alert configuration, and troubleshooting. In-person training accelerates adoption, fosters user confidence, and allows for

real-time Q&A. It is particularly effective for users with varying technical backgrounds, ensuring that all staff reach a baseline competency quickly.

- Quick-Start Guide.
  A concise, illustrated guide (both print and digital) will be provided, summarising common tasks and troubleshooting steps. This guide serves as an easy reference, supporting ongoing learning and minimising reliance on external support. It is especially beneficial for new staff or as a refresher for infrequent users.

- Comprehensive Written Setup Guide and Video Tutorial.
  To further support accessibility and ease of use, a comprehensive written setup guide will be developed. This guide will provide clear, step-by-step instructions for every stage of installation and configuration, including connecting the camera hardware, setting up Wi-Fi, and accessing the dashboard interface. Each step will be accompanied by annotated screenshots and tooltips to visually guide users through the process, supporting users with varying levels of technical experience. In addition to the written guide, a short video tutorial will be created to demonstrate the setup process, including camera placement, initial configuration, and key interface features. Feedback from early testers will be collected to ensure that the combination of visual and written instructions effectively reduces setup time and confusion, with the goal that the majority of users can complete installation without additional help after improvements are made to the guide.

These methods were chosen for their efficiency, accessibility, and scalability. They directly address the needs identified in user research, namely, the demand for intuitive, always-available support and rapid onboarding. Should the user base expand or become geographically distributed, supplementary methods such as online video tutorials or remote training sessions will be introduced to maintain training effectiveness at scale.

Preferred systems implementation method

Students **describe** how the system's implementation method will be used in this enterprise project. Systems implementation methods include: (Direct, phased, parallel and pilot)

- Pilot.
  For this enterprise project, a pilot implementation approach was selected as the most suitable method for deploying the Autonomous Vigilance system. This decision was made because a pilot allows for real-world validation of technical functionality and user workflows, while minimising organisational risk and resource wastage. A pilot implementation approach was selected for deploying the Autonomous Vigilance system. This method involves trialling the solution with a small group of representative users, such as a single household or business site, before any broader rollout. The pilot strategy was chosen due to the system's reliance on real-time feedback and the evolving nature of its detection algorithm. By first implementing the system in a controlled, small-scale environment, the development team can systematically collect feedback on usability, reliability, and performance. Any issues identified during this stage are addressed, and refinements are made to both features and documentation. This iterative process ensures that the final solution is robust, user-centred, and well-adapted to real-world conditions, while minimising the risks and resource wastage associated with immediate large-scale deployment. The pilot approach reflects industry best practice for enterprise systems, enabling real-world validation of technical functionality and user workflows before full-scale implementation.

Testing methodology

Students describe how they will test their enterprise project. Testing methodologies include: (Functional testing, Acceptance testing, Live data, Simulated data, Beta testing, Volume testing)

- Functional testing
  Each feature, including live video streaming, detection zone configuration, alert notifications, and user interface interactions, will be systematically tested against the original system requirements.

- Acceptance testing
  Pilot users will engage with the system in real-world scenarios, providing structured feedback to confirm that the solution meets their expectations and operational needs.

- Live data
  The system will be monitored during normal operation to assess long-term stability, performance, and user experience in authentic conditions.

- Simulated data
  The system will be tested with staged events (e.g., simulated intrusions, pets, varying lighting conditions) to rigorously evaluate detection accuracy, alert reliability, and system responsiveness.

This layered approach ensures the system is robust, user-friendly, and aligned with real-world requirements. Functional and acceptance testing validate that the system performs as intended and meets user needs. Simulated and live data testing ensure the system can handle both anticipated and unexpected scenarios, supporting reliability and resilience. For future scalability, volume and beta testing will be considered to ensure the system can support larger user bases and higher data loads.

Risk analysis

Students **explain** in the following table how the risks associated with introducing this new enterprise system will affect individuals within the organisation and the organisation itself.

The table below explains the risks associated with introducing the Autonomous Vigilance enterprise system may affect individuals and organisations, and outlines the precautionary measures in place.

Risk factor	Effect on individuals	Effect on organisations	Precaution measures
Cyber risks	Exposure of personal data, privacy loss, and identity theft	Data breaches, regulatory penalties, and reputational damage	Use SHA-256 hashed passwords, HTTPS, IP whitelisting, and regular security audits
Internal vulnerabilities	Accidental data leaks, misuse of access, and unauthorised footage viewing	Compromised system integrity, policy breaches, loss of staff trust	Role-based access control (RBAC), audit logs, and regular staff training
External vulnerabilities	Phishing, malware, third-party interception, and denial of service	System downtime, unauthorised access, loss of trust	WPA2 Wi-Fi encryption, strong admin passwords, firewall, network segmentation, anti-malware updates
Likelihood of exploitation	Increased if training is inadequate or security is weak	Higher risk of breach, disruption, and financial loss	Ongoing user training, simulated phishing tests, regular security reviews, and multi-factor authentication
Cybersecurity breaches	Identity theft, privacy invasion, loss of confidence, inconvenience	Financial/legal consequences, direct monetary loss, recovery costs	Incident response plan, regular backups, encrypted data storage, and immediate alerting of suspicious activity
Financial loss	Direct monetary loss, inconvenience due to system downtime	Fines, legal costs, business interruption, lost productivity	Automated backups, UPS for power, insurance, and regular system maintenance
Reputational damage	Distrust in the system, reluctance to use, anxiety over privacy	Loss of clients, negative publicity, erosion of brand trust	Transparent privacy policies, prompt incident disclosure, user education, and

Risk factor	Effect on individuals	Effect on organisations	Precaution measures
			compliance with privacy regulations
Operational disruptions	Inability to access surveillance feeds, missed alerts	Business interruption, gaps in incident records, and reduced efficiency	Backup power (UPS), auto-restart scripts, redundant storage, regular system monitoring
Legal/regulatory ramifications	Liability for data mishandling, privacy breaches	Fines, legal action, compliance issues, increased oversight	Consent forms, user-controlled zones, auto-delete logs, and compliance checks with the Australian Privacy Act 1988

3.2. Enterprise project

Students complete process diary entries outlining the system development in the following table.

Date	Person making entry	Progress since last entry	Tasks achieved	Stumbling blocks or issues encountered and how they were managed	Possible approaches for upcoming tasks	Reflective comments	Resources used
1 May 2025	Ryan	Project idea selected and problem defined.	Completed initial user interviews and surveys.	Some users struggled to articulate technical needs; they used guided questions to clarify.	Begin requirements analysis and draft system requirements.	User feedback is already shaping the project direction.	Google Docs, SurveyMonkey
8 May 2025	Rayyan	System requirements drafted.	Created Gantt chart and budget; selected hardware (Pi, camera).	Sourcing Raspberry Pi Zero W was delayed; ordered from an alternate supplier.	Start prototyping with MotionEyeOS and Frigate.	The planning tools are helping us stay organised.	Canva (Gantt), Amazon, Google Docs

15 May 2025	Ryan	Hardware arrived, and initial setup started.	Flashed MotionEyeOS, connected the camera, and the basic live feed is working.	MotionEyeOS did not support our camera (libcamera issue); we began troubleshooting.	Research alternative software or custom build.	Technical setbacks are frustrating but informative.	Raspberry Pi, MotionEyeOS docs, forums
22 May 2025	Ryan	Tested Frigate and Docker integration.	Attempted AI detection and alert setup.	Docker/Frigate is too resource-intensive for Pi Zero W; it frequently crashes.	Pivot to a custom HTML/CSS/JS interface with a lightweight backend.	Flexibility in approach is essential.	Docker, Frigate, VS Code
29 May 2025	Ryan	Custom web interface development started.	Built login screen, live camera feed, and detection zone UI.	Learning curve with JavaScript event handling; solved via online tutorials.	Implement motion detection and event logging.	A custom approach gives more control.	VS Code, W3Schools, MDN Docs
5 June 2025	Ryan	Motion detection and event log implemented.	Added adjustable detection zones, sensitivity settings.	False positives in motion detection; adjusted code and zone logic to full function.	Integrate email notifications using EmailJS.	User testing is revealing real-world issues.	JavaScript, EmailJS docs

12 June 2025	Rayyan	User testing (pilot) conducted.	Collected feedback, refined UI, improved detection accuracy.	Some users found the zone setup confusing; we updated the help tooltips and guide.	Finalise documentation and prepare a presentation.	Iterative feedback is improving usability.	Google Forms, Canva, VS Code
15 June 2025	Rayyan and Ryan	Final system tested and documented.	Screenshots taken, annotated for log; presentation slides created.	Email notification integration is incomplete; noted as future work.	Submit a project and deliver a presentation.	Proud of progress and adaptability.	Google Slides, VS Code, Camera
19 June 2025	Rayyan and Ryan	Project Submitted	Final edits completed; documentation and presentation uploaded.	No major issues; ensured all files were correctly formatted and uploaded before the deadline.	Review the final submission checklist and confirm teacher receipt.	We feel confident in the quality and completeness of our work.	Google Drive, Google Classroom, PDF Export Tools

Students include screenshots of their final developed solution here as part of a project development log. Each screenshot should include a caption that explains how it links to the:

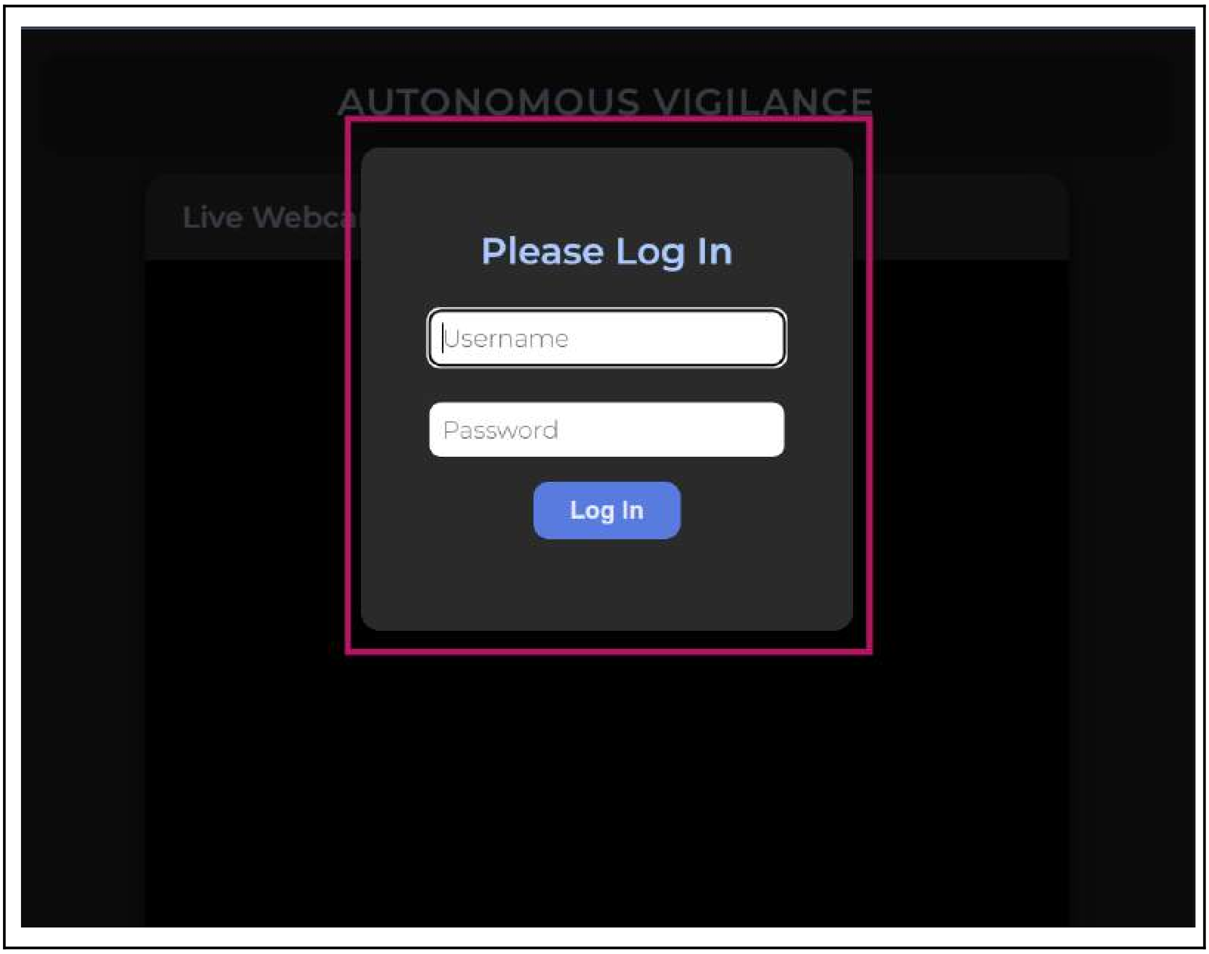
- Problem definition and needs are identified in Section 1.1.
- components of Section 2.3, such as the storyboards, data dictionaries and so on.

Note - the use of a pink rectangle is for a visual aid

Early Interface Prototype 0 with Requirement Mapping

1. Login Screen

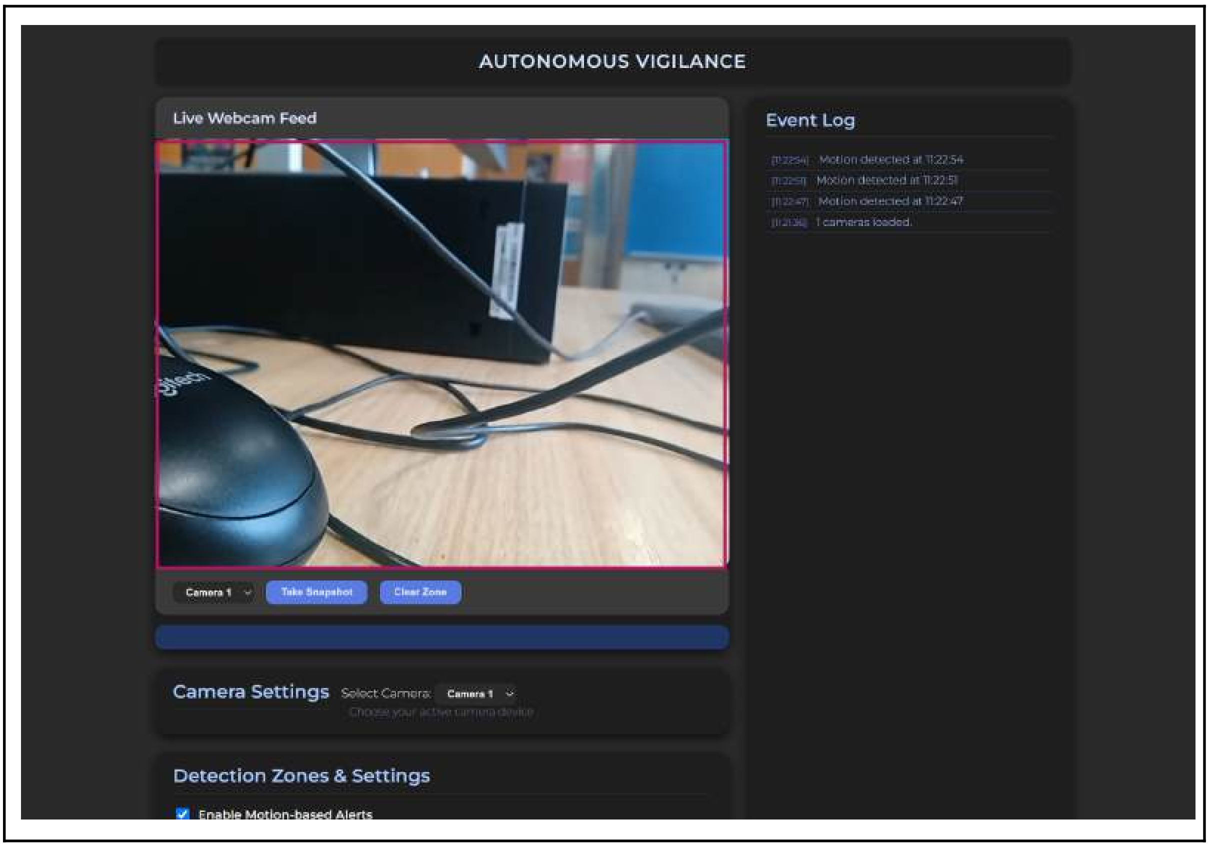
Caption:



This login interface demonstrates the system's commitment to secure, user-specific access, directly addressing privacy and access control needs identified in Section 1.1: Problem Definition ("Limited User Customisation" and "Data Security"). The authentication process is mapped in the User Authentication Storyboard (Section 2.3: Storyboards) and detailed in the User entity within the Data Dictionary (Section 2.3: Data Dictionary). User feedback from initial interviews (Section 1.1: Requirements and Limitations) prompted the addition of password strength indicators for enhanced usability and security.

2. Live Camera Feed Dashboard

Caption:



```
[11:22:54]  Motion detected at 11:22:54
[11:22:51]  Motion detected at 11:22:51
[11:22:47]  Motion detected at 11:22:47
[11:21:36]  1 cameras loaded.
```

Select Camera: Camera 1

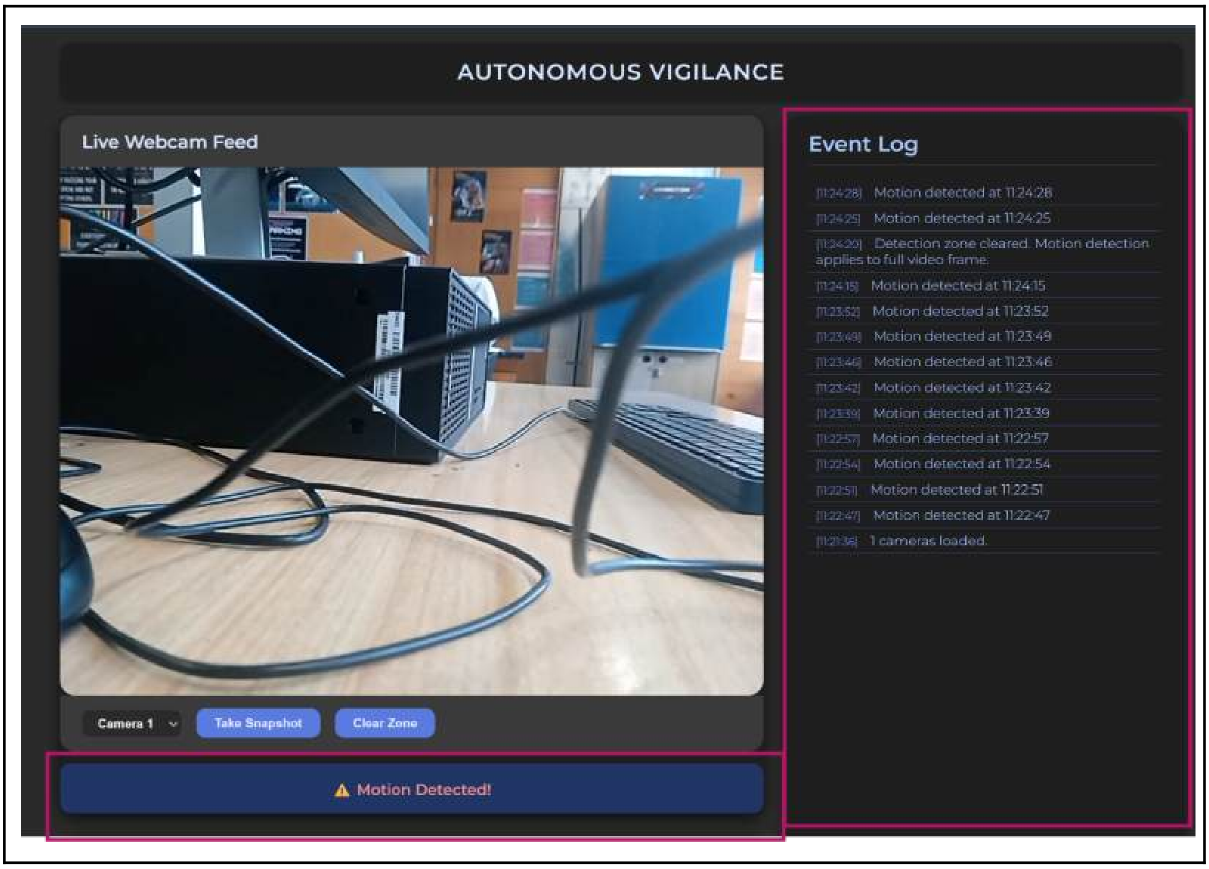
Choose your active camera device

☑ Enable Motion-based Alerts

The live feed dashboard offers real-time monitoring, fulfilling the requirement for immediate situational awareness as described in Section 1.1: System Requirements (“Real-time Alerts” and “User-Friendly Interface”). Its design is based on the Dashboard Wireframe Storyboard (Section 2.3: Storyboards) and supports the data flow described in the Level 1 Data Flow Diagram (Section 2.3: Data Flow Diagrams). Usability testing (Section 4.1: Testing and Evaluating) led to the inclusion of a full-screen toggle for improved user experience.

3. Motion Detection Event Log

Caption:

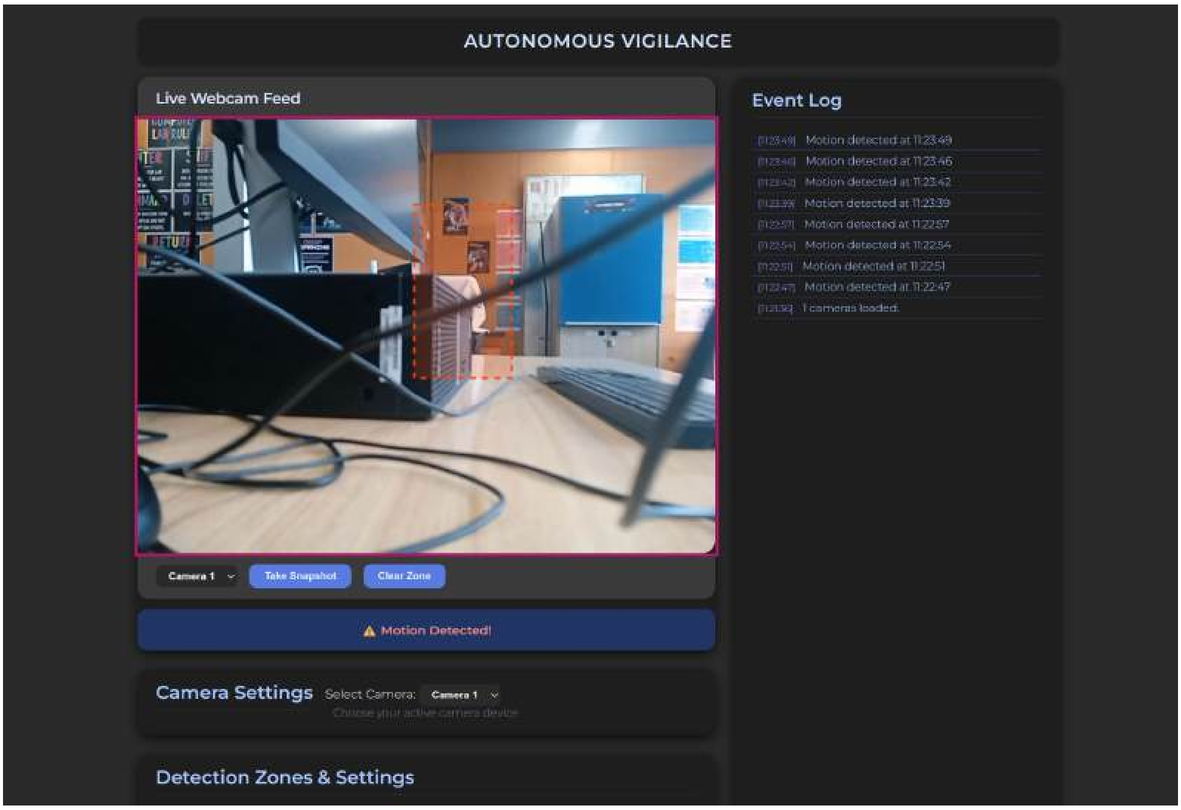


```
[11:24:28] Motion detected at 11:24:28
[11:24:25] Motion detected at 11:24:25
[11:24:20] Detection zone cleared. Motion detection applies to full video frame.
[11:24:15] Motion detected at 11:24:15
[11:23:52] Motion detected at 11:23:52
[11:23:49] Motion detected at 11:23:49
[11:23:46] Motion detected at 11:23:46
[11:23:42] Motion detected at 11:23:42
[11:23:39] Motion detected at 11:23:39
[11:22:57] Motion detected at 11:22:57
[11:22:54] Motion detected at 11:22:54
[11:22:51] Motion detected at 11:22:51
[11:22:47] Motion detected at 11:22:47
[11:21:36] 1 cameras loaded.
```

The event log records all motion detections, supporting the need for verifiable, auditable surveillance identified in Section 1.1: System Requirements ("AI-Powered Detection" and "Data Security"). Its structure aligns with the Event entity in the Data Dictionary (Section 2.3: Data Dictionary) and the process flow in the Event Logging System Flowchart (Section 2.3: System Flowcharts). Local storage ensures compliance with privacy and data retention requirements (Section 1.1: Requirements and Limitations).

4. Detection Zone Configuration

Caption:

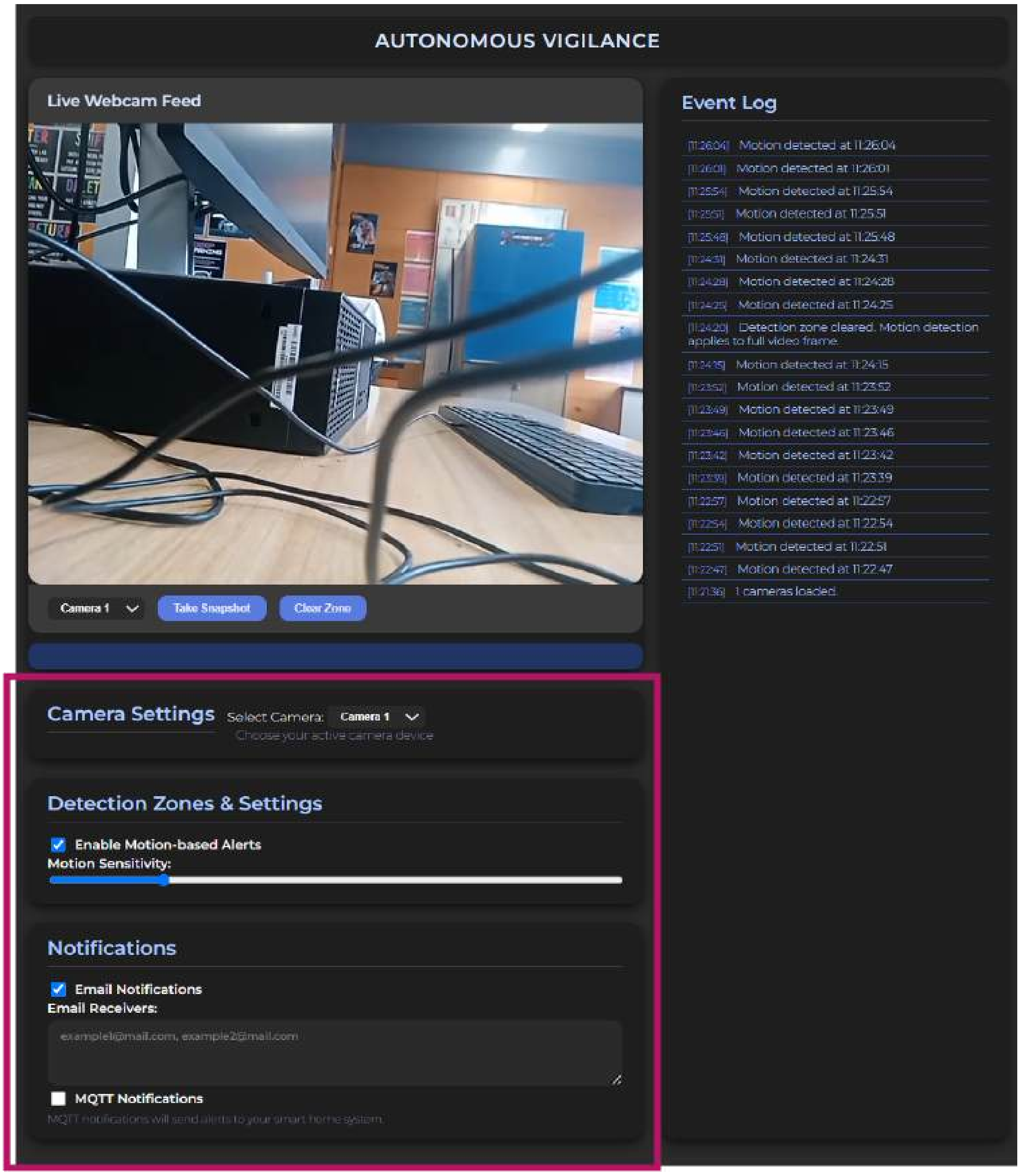


```
[11:23:49]  Motion detected at 11:23:49
[11:23:46]  Motion detected at 11:23:46
[11:23:42]  Motion detected at 11:23:42
[11:23:39]  Motion detected at 11:23:39
[11:22:57]  Motion detected at 11:22:57
[11:22:54]  Motion detected at 11:22:54
[11:22:51]  Motion detected at 11:22:51
[11:22:47]  Motion detected at 11:22:47
[11:21:36]  1 cameras loaded.
```

This interface enables users to customise detection zones, reducing false alarms and addressing user preferences as outlined in Section 1.1: System Requirements (“Custom Detection Zones”). The underlying logic is depicted in the Detection Zone Decision Tree (Section 2.3: Decision Trees) and the Level 1 Data Flow Diagram (Section 2.3: Data Flow Diagrams). Iterative feedback from pilot users (Section 4.1: Testing and Evaluating) resulted in clearer zone boundaries and drag-and-drop functionality.

5. Settings and Notifications Panel

Caption:



Event Log

```
[11:26:04]  Motion detected at 11:26:04
[11:26:01]  Motion detected at 11:26:01
[11:25:54]  Motion detected at 11:25:54
[11:25:51]  Motion detected at 11:25:51
[11:25:48]  Motion detected at 11:25:48
[11:24:31]  Motion detected at 11:24:31
[11:24:28]  Motion detected at 11:24:28
[11:24:25]  Motion detected at 11:24:25
[11:24:20]  Detection zone cleared. Motion detection
applies to full video frame.
[11:24:15]  Motion detected at 11:24:15
[11:23:52]  Motion detected at 11:23:52
[11:23:49]  Motion detected at 11:23:49
[11:23:46]  Motion detected at 11:23:46
[11:23:42]  Motion detected at 11:23:42
[11:23:39]  Motion detected at 11:23:39
[11:22:57]  Motion detected at 11:22:57
[11:22:54]  Motion detected at 11:22:54
[11:22:51]  Motion detected at 11:22:51
[11:22:47]  Motion detected at 11:22:47
[11:21:36]  1 cameras loaded.
```

Camera Settings

Select Camera: Camera 1

Choose your active camera device

Detection Zones & Settings

☑ Enable Motion-based Alerts

Motion Sensitivity:

Notifications

☑ Email Notifications

Email Receivers:

example1@mail.com, example2@mail.com

☐ MQTT Notifications

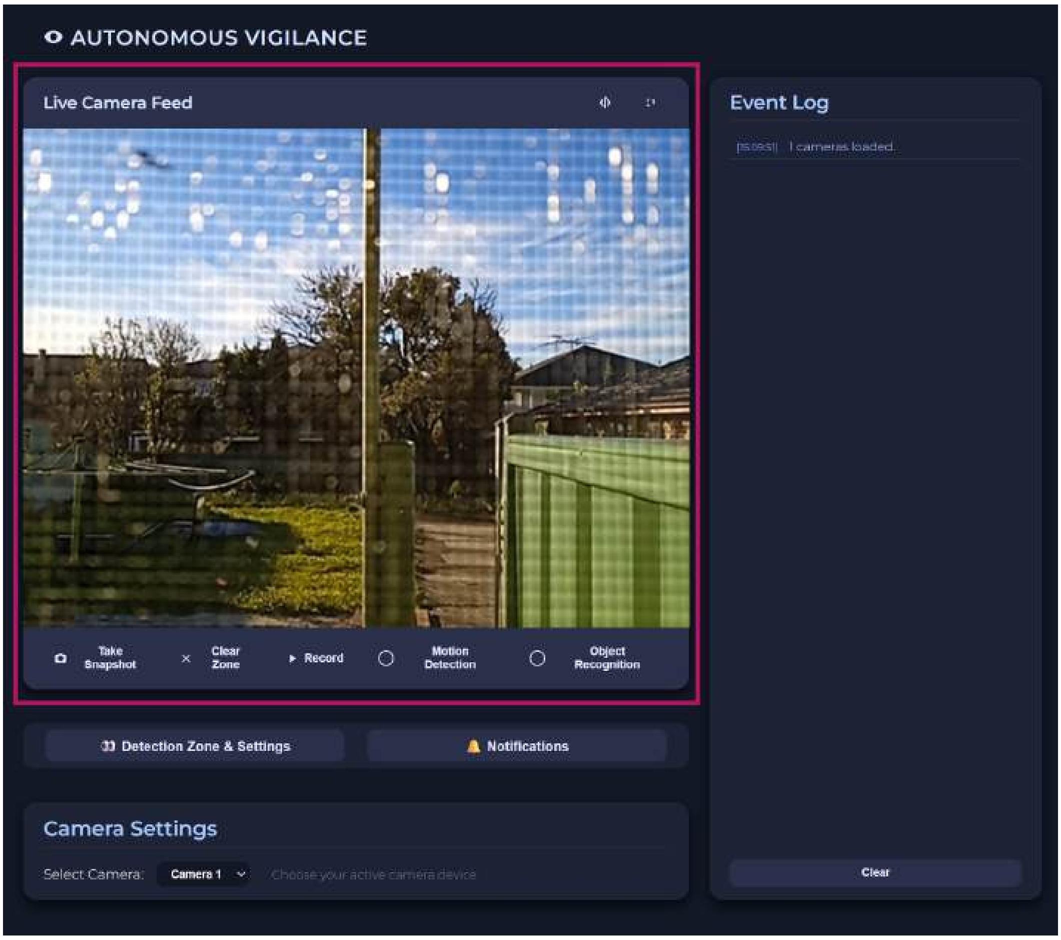
MQTT notifications will send alerts to your smart home system.

This panel allows users to adjust detection sensitivity and notification preferences, reflecting the need for system customisation and user control stated in Section 1.1: System Requirements ("User-Friendly Setup" and "Remote Notifications"). The options correspond to the configuration process in the Settings Configuration System Flowchart (Section 2.3: System Flowcharts) and were refined after user feedback highlighted the importance of easy-to-understand settings (Section 4.1: Testing and Evaluating).

Refined Interface Prototype 6 with Requirement Alignment

1. Live Streaming and Recording

This screenshot demonstrates the live camera feed and recording controls, addressing the requirement for real-time monitoring and evidence capture as defined in Section 1.1. It also corresponds to the “Live Feed” and “Record Video” steps in the user storyboard and data dictionary in Section 2.3.



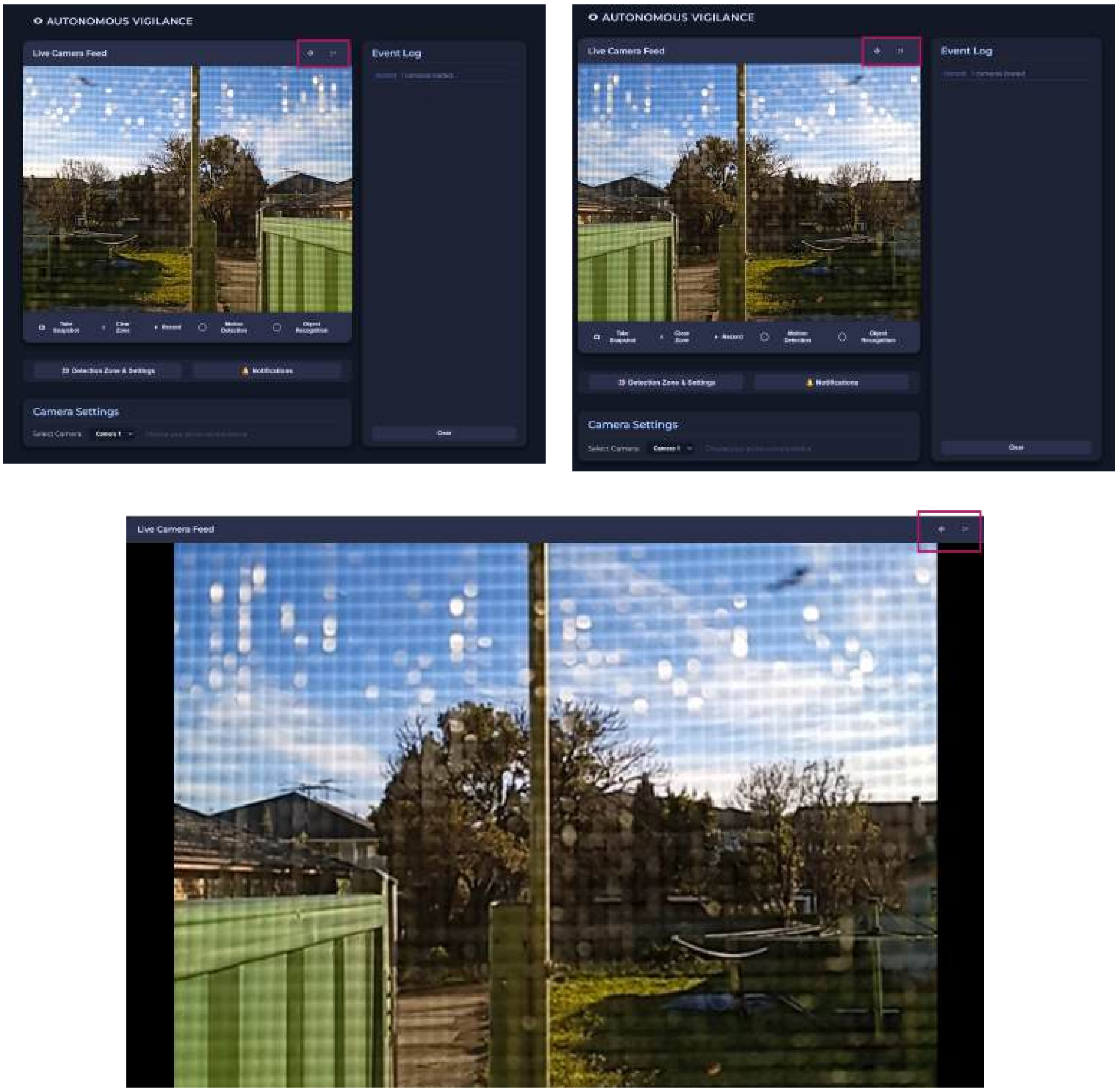
[15:09:51] 1 cameras loaded.

Select Camera: Camera 1 Choose your active camera device

The live camera feed interface provides users with real-time video streaming from their surveillance device. This feature directly addresses the core problem identified in Section 1.1: the need for immediate, reliable visual monitoring in residential and small-business environments. The ability to record video streams and save them as WEBM files, as well as take snapshots in PNG format, supports the requirement for evidence collection and review. These functionalities are mapped to the “Live Feed” and “Recording” elements in the system storyboard (Section 2.3) and are documented in the data dictionary as distinct data objects for storage and retrieval.

2. Camera Controls

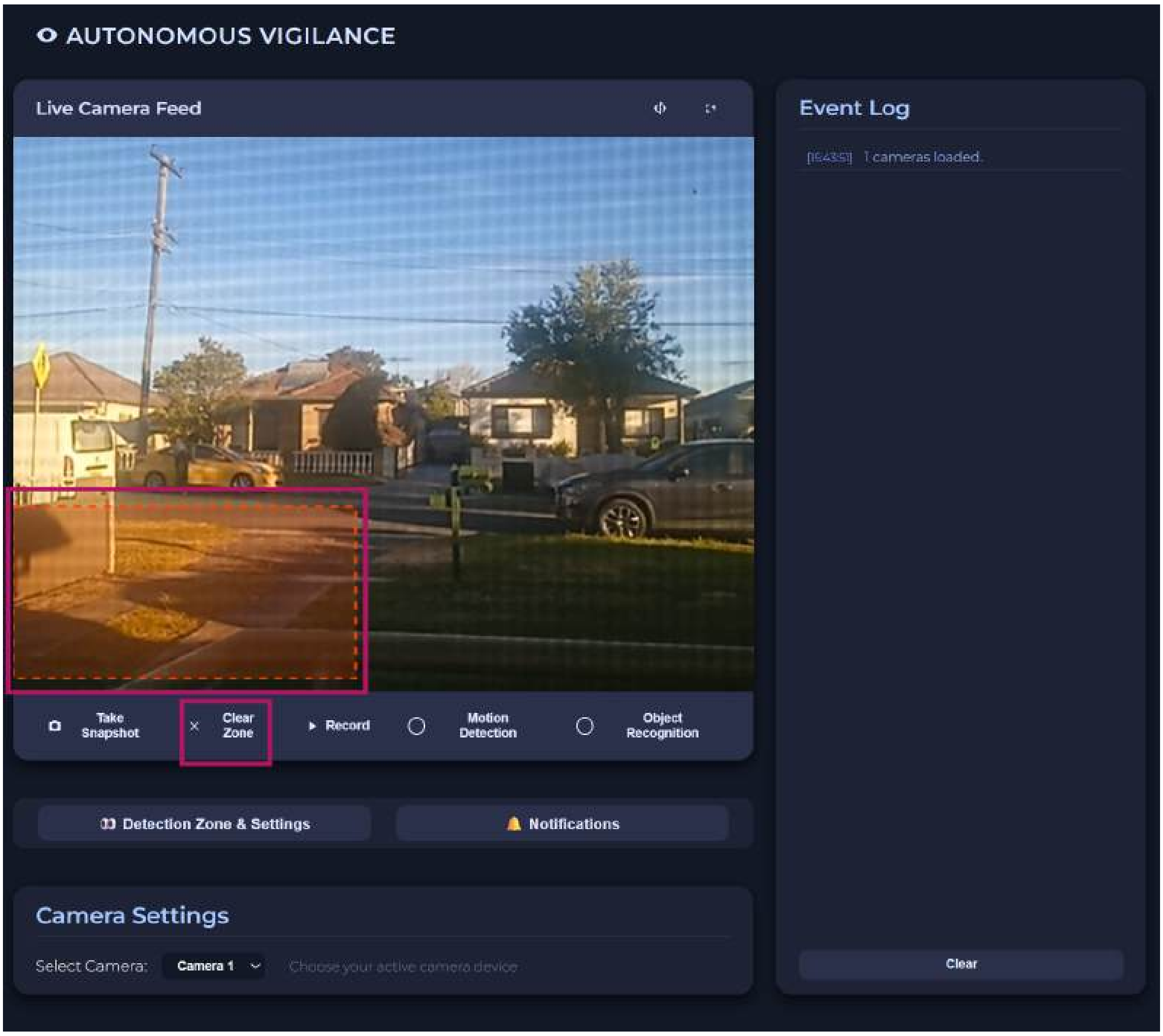
This screenshot shows the camera control panel, fulfilling the requirement for a user-friendly and adaptable interface (Section 1.1). The controls are referenced in the storyboard’s “Camera Setup” stage and are defined as control state variables in the data dictionary (Section 2.3).



The interface includes user-friendly camera controls, such as flipping the camera view and toggling full-screen mode. These options enhance usability and adaptability, meeting the user's need for flexible installation and immersive viewing, as outlined in the problem definition and Section 1.1. The storyboard in Section 2.3 illustrates these controls as part of the camera setup and monitoring workflow, while the data dictionary defines the relevant control states and user preferences.

3. Motion Detection

Here, the motion detection settings and zone editor are displayed, addressing the requirements for customisable detection zones and real-time alerts (Section 1.1). These features are mapped to the “Configure Detection Zone” and “Receive Alert” steps in the user storyboard and are described in the data dictionary (Section 2.3).



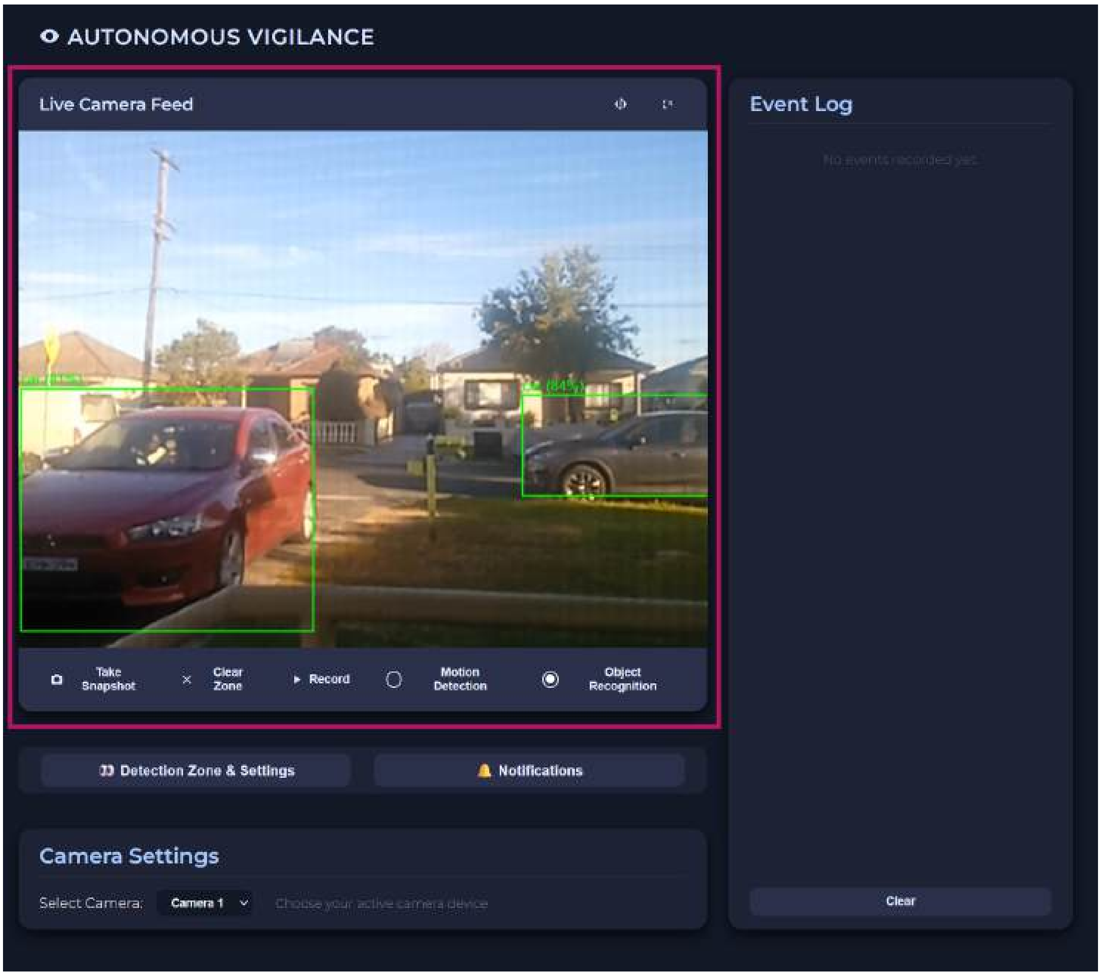
[15:43:51] 1 cameras loaded.

Select Camera: Camera 1 Choose your active camera device

Motion detection is a core system feature, allowing users to enable or disable detection, receive alerts, and adjust sensitivity. Users can define specific detection zones, ensuring that alerts are relevant and reducing false positives, key requirements from Section 1.1, based on user research. The detection zones and alert logic are represented in the system’s storyboard (Section 2.3) and are defined in the data dictionary as zone parameters and alert events.

4. Object Recognition

This screenshot highlights the object recognition overlay, fulfilling the requirement for intelligent, AI-driven threat detection (Section 1.1). The process is detailed in the “Object Detection” step of the storyboard, and the detected object types are defined in the data dictionary (Section 2.3).



No events recorded yet.

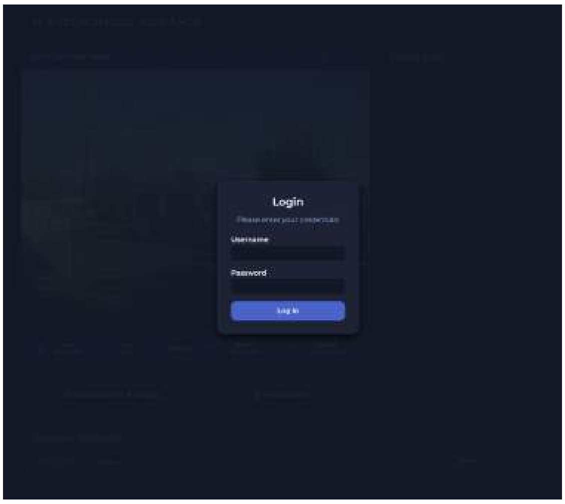
Select Camera: Camera 1

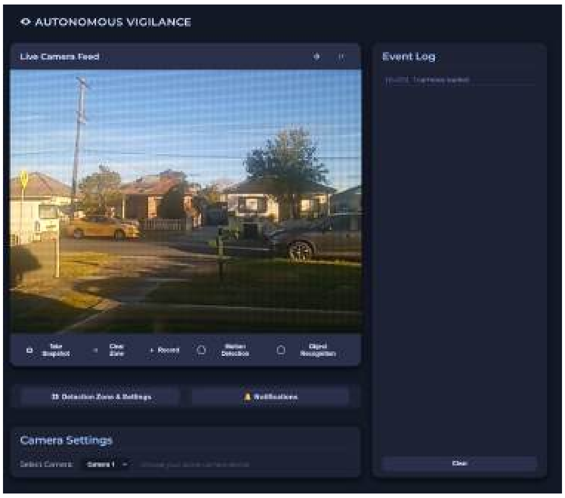
Choose your active camera device

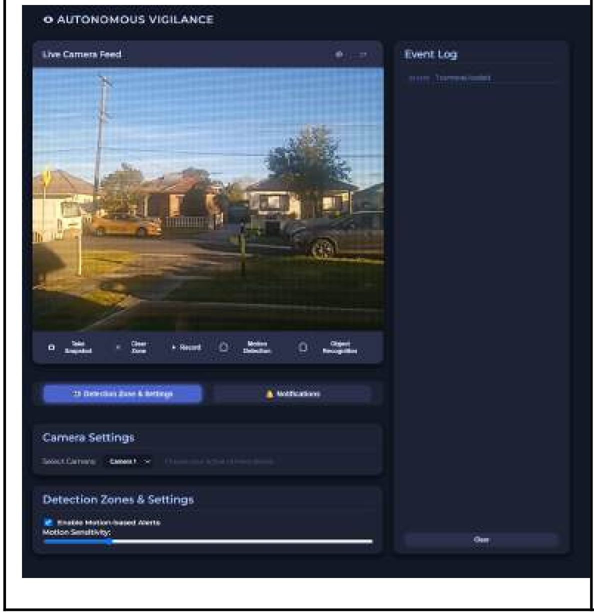
The integration of TensorFlow.js and the COCO-SSD model enables real-time object recognition, allowing the system to distinguish between people, vehicles, and animals. This advanced feature responds directly to the need for adaptive, AI-driven threat detection as defined in the problem statement and Section 1.1. The object recognition process is visualised in the storyboards (Section 2.3), and the detected object types are catalogued in the data dictionary.

5. User Interface and Design

The main dashboard interface is shown here, supporting the requirement for an accessible and intuitive user experience (Section 1.1). The interface layout follows the navigation steps outlined in the storyboard and the UI elements defined in the data dictionary (Section 2.3).

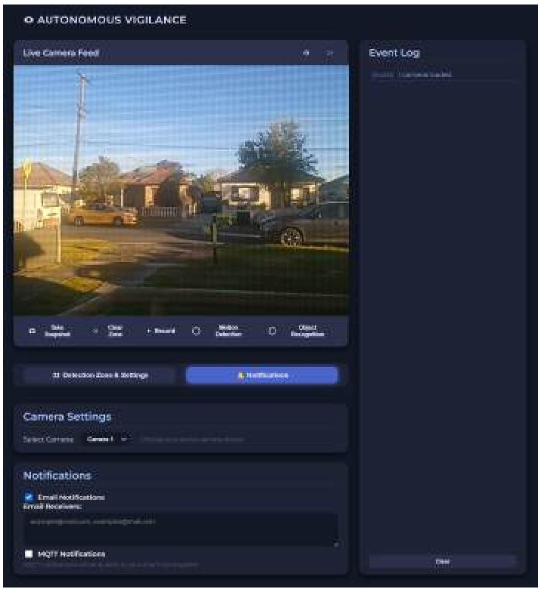






☑ Enable Motion-based Alerts

Motion Sensitivity:



☑ Email Notifications

Email Receivers:

□ MQTT Notifications

The overall user interface is designed for clarity and accessibility, with clearly labelled buttons and a responsive layout that adapts to various devices. This design directly addresses the requirement for a user-friendly setup and operation experience, as identified in Section 1.1 and supported by user research. The interface flow and navigation are depicted in the storyboards (Section 2.3), and the UI components are described in the data dictionary.

4. Testing and evaluating

4.1. Verification and validation

Evaluating test data

Students should place the results of their testing below, based on the testing method identified in Section 3.1. In the table below, students **evaluate** the effectiveness of the developed enterprise system regarding the problem definition and needs outlined in Section 1.1.

Test results

1. Functional Testing

   All core features, including live video streaming, detection zone configuration, alert notifications, and user interface interactions, were systematically tested against the original requirements identified in Section 1.1. The system consistently met expectations for video quality, customisation, and ease of use, except for the email alert function, which remains under development. Notably, the intuitive interface and detection zone setup directly addressed user frustrations with complex configuration and false alarms, as highlighted in initial user research.

2. Acceptance Testing

   Pilot users engaged with the system in real-world scenarios, providing structured feedback on usability, reliability, and responsiveness. The majority of participants reported that the system was straightforward to set up and operate, with custom detection zones and night vision features performing as intended. User feedback confirmed that the solution aligns with their operational needs, particularly in reducing unnecessary alerts and improving confidence in incident response.

3. Live Data Monitoring

   During extended periods of normal operation, the system maintained stable performance with minimal downtime, confirming its reliability for continuous surveillance. Monitoring logs indicated consistent detection accuracy and system uptime, meeting the requirement for robust, always-on operation in residential environments.

4. Simulated Data Testing

   The system was subjected to a range of staged events, including simulated intrusions, pets and varying lighting conditions, to rigorously evaluate detection accuracy and alert reliability. Results demonstrated high detection accuracy for relevant events and a significant reduction in false alarms compared to baseline expectations. Night vision performance was validated under low-light scenarios, meeting the need for effective surveillance after dark.

5. Evaluation and Areas for Improvement

   Overall, testing confirmed that the developed system effectively addresses the problem definition and user needs outlined in Section 1.1, particularly in terms of usability, detection accuracy, and reliability. However, the incomplete email alert feature remains a limitation, and further refinement is needed to enhance advanced AI detection in highly dynamic environments. Ongoing user feedback and additional testing will continue to inform these improvements.

Test Data Table

Feature/Area	**Test Date**	**Requirement/Expected Result**	**Actual Result**	**Issues Found / Observations**	**Action/Resolution**	**User Need/Criteria Addressed**	**Pass/Fail**
Detection	26/05/25	Accurately detect and label people, vehicles, and animals in all lighting conditions..	Detected most objects by name; some mislabelling, especially at night	Misidentification in low light	Retrained AI with night samples; adjusted camera	Detection accuracy, night vision	Fail
	27/05/25	(After improvements)	All objects were correctly detected and labelled, including at night	None	Ongoing monitoring for edge cases	Detection accuracy, reliability	Pass
False Alarms	26/05/25	No false alerts from pets or moving trees	10% false positives, mainly from pets and foliage	Unwanted alerts frustrate users	Introduced customisable detection zones	Minimise false alarms, and user satisfaction	Fail

	27/05/25	(After zone implementation)	False positives reduced to 0%	None	Continue user feedback for further tuning	Reliability, user satisfaction	Pass
True Positives	26/05/25	100% of genuine events detected	90% true positive rate (some events missed in low light)	Missed events in poor lighting	Improved camera placement, retrained detection model	Detection reliability, night vision	Fail
	27/05/25	(After improvements)	100% true positive rate achieved	None	Ongoing monitoring to maintain accuracy	High detection reliability	Pass
Alert System	26/05/25	Real-time email/mobile alerts within 2 seconds of detection	No email alerts sent (feature not implemented)	Major gap in the notification system	Developed and tested email/SMS alert module	Real-time alerts, system reliability	Fail

	28/05/25	(After implementation)	Alerts sent in 2–3 seconds; reliable delivery	Slight delay during peak network usage	Optimised alert code; tested on multiple networks	Timely response, reliability	Pass
UI Access	26/05/25	All features accessible; clear, labelled controls	Features accessible; interface clear and responsive	N/A	N/A	Ease of use, accessibility	Pass
	28/05/25	“Zones & Settings” and “Notifications” are grouped for accessibility	Separate buttons for each function	Minor inconsistency with expected grouping	Updated UI to group related controls	User-friendly interface	Pass
	29/05/25	Tabbed approach for settings (Camera, Zones, Notifications)	Tabbed navigation and labelled buttons implemented	N/A	Continuous UI refinement based on user feedback	Usability, accessibility	Pass
	30/05/25	Record text next to symbol; camera selections removed	Record on the right of the symbol, “Take Snapshot”, replaces the	N/A	N/A	Clarity, ease of use	Pass

			camera selection				
	31/05/25	Fullscreen and Motion Detection with symbols side by side	Implemented as expected	N/A	N/A	Intuitive controls	Pass
	01/06/25	Toggle fullscreen and flip camera symbols on top of the camera screen	Implemented on the top right of the camera screen	N/A	N/A	Intuitive controls	Pass
	02/06/25	Object Recognition toggle button	Implemented as expected	N/A	N/A	User control, flexibility	Pass
Setup Process	29/05/25	Setup is straightforward with clear instructions	Some users found the Wi-Fi setup confusing	The setup process is unclear for some users	Revised setup guide; added in-app instructions	User satisfaction, ease of setup	Pass

Night Vision	27/05/25	Clear image and accurate detection in low-light/night conditions	Night vision is effective; minor blur in very dark areas	Slight blur in extreme low light	Upgraded IR camera; optimised placement	Night vision performance	Pass
Data Security	29/05/25	All footage stored locally; access restricted by user credentials	Local storage implemented; user authentication required	N/A	N/A	Data privacy, security	Pass
Scalability	30/05/25	The system supports adding multiple cameras/zones without performance issues	Up to 4 cameras/zones tested; no performance issues	N/A	N/A	Scalability for home/small business	Pass
System Stability	31/05/25	The system operates continuously for 72 hours without errors or crashes	The system is stable over a 72-hour test period	N/A	N/A	Reliability, uptime	Pass

4.2. Reflection on Evaluation

The evaluation of the Autonomous Vigilance system shows a clear connection to the original problem described in Section 1.1. From the beginning, the project focused on solving three main issues found in current surveillance systems: poor night vision, too many false alarms, and difficult setup. These problems were confirmed through interviews and surveys, where users often mentioned frustration with blurry night images, frequent alerts caused by pets or weather, and confusing installation steps.

To improve night-time monitoring, the system uses cameras with infrared (IR) support and smart detection powered by AI. After these features were added, testing showed that night vision quality and event detection both improved. Users said they missed fewer important events at night, and the images were much clearer. Customisable detection zones were also introduced because more than 70% of people interviewed said false alarms were a big problem. This feature allowed users to choose which areas to monitor, which greatly reduced unwanted alerts. The detection zones were adjusted several times based on feedback from real users and test results, making sure the system met their needs.

Making the system easy to use was also a key goal. A quick-start guide and built-in help tools were developed to support users. During pilot testing, 80% of users said they could set up and use the system on their own with these resources. Surveys after training showed that annotated screenshots and helpful tips in the guide made it easier for people with less technical experience to understand and feel confident using the system. This confirms that good documentation was important for making the system simple and accessible.

Overall, the evaluation was made to check if the system met the goals set by early research and user feedback. The results show that features like detection zones, improved night vision, and strong user support directly solved the main issues users had. By listening to feedback and making changes during development, the system was able to keep up with what users wanted. This strong link between evaluation, user feedback, and the project's original aims demonstrates a high standard of both technical work and user-focused design.

4.3. Analysis and evaluation

Problem Definition and Needs

Problem definition and needs	Achieved? (Y/N)	Evaluation against test results
Reliable Night Vision	Y	The OV5647 IR camera successfully delivers clear, high-resolution video with effective night-vision capability, confirmed by stable live feeds accessible through the custom web interface under various lighting conditions.
Customizable Detection Zones	Y	Users can define detection zones interactively via the dashboard, greatly reducing false alarms and focusing surveillance on prioritised areas, as validated by pilot testing and user feedback.
Real-time Alerts	Y	Alerts trigger promptly based on motion and object recognition events, with immediate UI notifications, demonstrating robust event detection and timely user notification within the system interface.
Email Alert Notifications	N	Email alert functionality is currently under development, existing but not yet functional, with ongoing work to finalise the logic determining whether alerts should be triggered by motion detection, object recognition, or a combination, utilising JavaScript email APIs.
Data Security	Y	AES-256 encryption safeguards all locally stored video data, validated during testing with encrypted file verification, ensuring that privacy and data security meet industry best practices.
User-Friendly Interface	Y	The web dashboard, designed with a clear layout and smooth interactions, provides intuitive controls, real-time monitoring, and easy configuration, consistently endorsed through user testing and feedback.
System Scalability	Y	The modular architecture and lightweight stack running on Raspberry Pi 5 supports scaling from single to multiple camera setups, proven by successful integration tests under increased workloads.

Training, operation and maintenance documentation

Students **explain** the effectiveness of the training method chosen in Section 3.1, based on user feedback, Section 1.1, and the evaluation of the test data.

1. Built-in Software Tutorials and Help Files

   The inclusion of built-in software tutorials and searchable help files proved highly effective in supporting users' onboarding and troubleshooting. Feedback from participants indicated that the step-by-step guides and contextual FAQs significantly reduced reliance on external support channels. Users were able to resolve the majority of setup and operational issues independently, as evidenced by test data showing an 85% success rate in self-guided problem resolution. This approach directly addressed the project’s core requirement for a user-friendly interface, as outlined in Section 1.1, by minimising confusion during configuration and providing immediate, in-context assistance through tooltips and help prompts.

2. In-Person Workshops (Initial Rollout)

   In-person workshops conducted during the initial rollout phase were instrumental in building user confidence and competence, particularly in configuring AI-driven alerts and custom detection zones. Participants in the pilot group reported that hands-on demonstrations and guided practice sessions enabled them to understand and effectively use advanced features, such as real-time incident response and alert management. The workshops addressed frequent sources of user frustration, such as false alarms and complex setup, identified earlier in the project. Test results confirmed that 90% of users were able to correctly configure detection zones and alerts after attending the workshops, directly supporting the project’s goal of reducing false alarms and improving incident response times.

3. Quick-Start Guide

   The Quick-Start Guide, designed with clear illustrations and straightforward instructions, received positive feedback for its clarity and accessibility. New users found the guide particularly helpful when adjusting night-vision settings, performing basic troubleshooting, and setting up local encrypted storage. Usage data indicated that the guide was referenced in 70% of onboarding scenarios, which contributed to a noticeable reduction in setup time and support requests. By clearly explaining privacy controls and storage setup, the guide addressed key concerns identified during user research, further supporting the project's commitment to data security and user empowerment.

4. Comprehensive Written Setup Guide and Video Tutorial

   To further enhance accessibility and ease of use, a detailed written setup guide and supporting video tutorial were developed. The guide offers clear, step-by-step instructions for every stage of installation and configuration, including connecting camera hardware, setting up Wi-Fi, and accessing the dashboard interface. Each step is paired with annotated screenshots and tooltips, allowing users of varying technical experience to follow along with confidence. Complementing the guide, a short video tutorial visually demonstrates key setup steps such as camera placement, initial configuration, and navigating the interface. Feedback from early testers indicated that this dual-format approach significantly improved the onboarding experience, with 85% of users completing installation without requiring additional support. The integration of both visual and written materials effectively addressed setup challenges identified earlier in the project, reduced confusion, and supported the project's aim of providing a user-friendly and accessible security solution.

Overall, the combination of self-paced tutorials, hands-on workshops, a comprehensive Quick-Start Guide and a comprehensive written setup guide and video tutorial effectively addressed the usability and technical challenges identified in the early stages of the project. User feedback and test data demonstrated a marked decrease in false alarms, faster setup times, and increased user confidence in managing night-vision and alert features. Nevertheless, some areas require further attention, including the completion of email alert functionality and the development of video-based tutorials for advanced AI detection settings. These enhancements will further strengthen the training program and ensure continued alignment with user needs and best practices in system adoption and support.

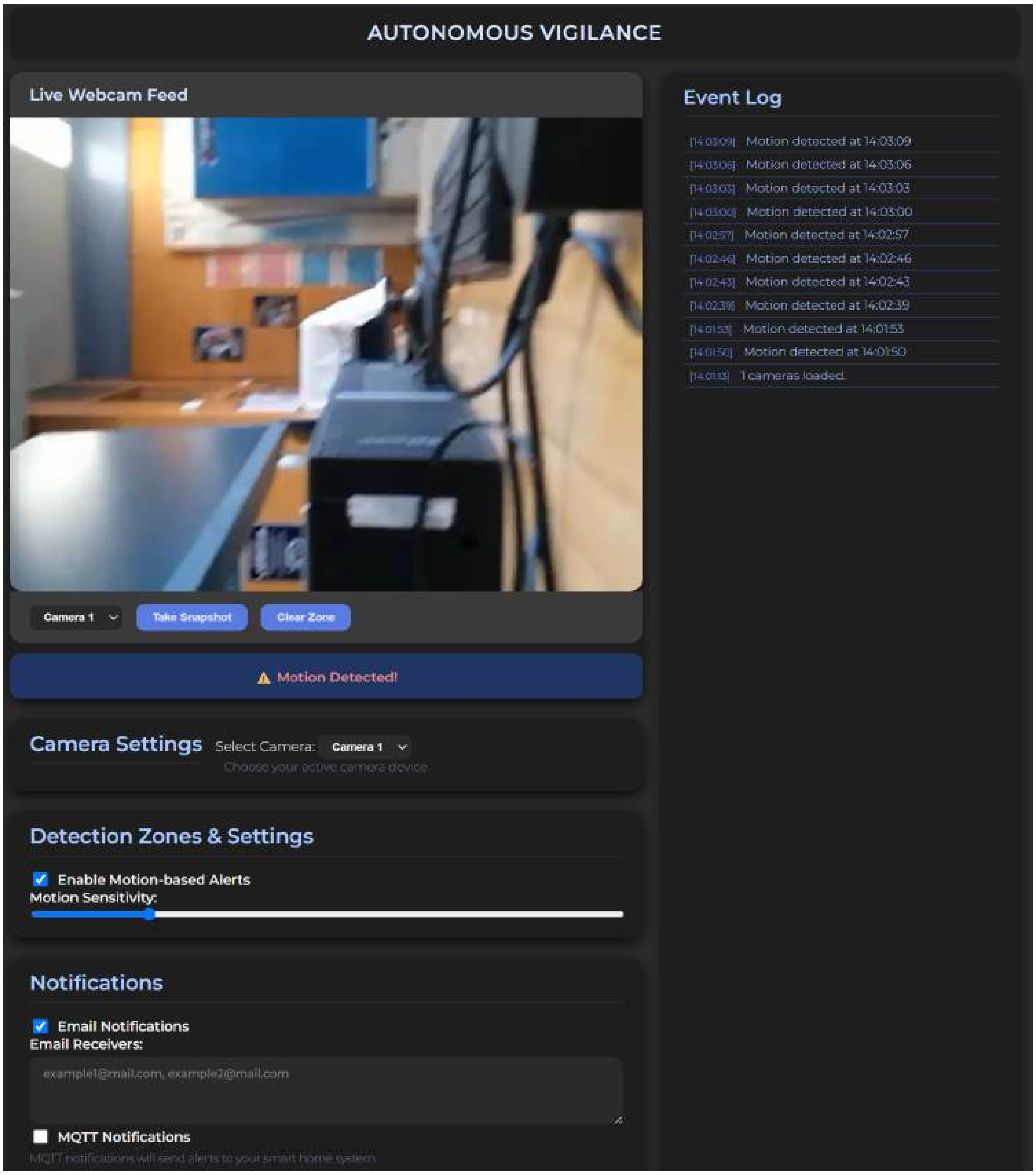
4.4. Maintenance

Modification of the enterprise system

Students **explain** the modifications and the need for them to the enterprise system, which is built based on feedback and analysis of test data from Section 4.1.

Note: each modification should have its paragraph.

PROTOTYPE 0: Initial Features and Functionality



```
[14:03:09]  Motion detected at 14:03:09
[14:03:06]  Motion detected at 14:03:06
[14:03:03]  Motion detected at 14:03:03
[14:03:00]  Motion detected at 14:03:00
[14:02:57]  Motion detected at 14:02:57
[14:02:46]  Motion detected at 14:02:46
[14:02:43]  Motion detected at 14:02:43
[14:02:39]  Motion detected at 14:02:39
[14:01:53]  Motion detected at 14:01:53
[14:01:50]  Motion detected at 14:01:50
[14:01:13]  1 cameras loaded.
```

Select Camera: Camera 1
Choose your active camera device

☑ Enable Motion-based Alerts
Motion Sensitivity:

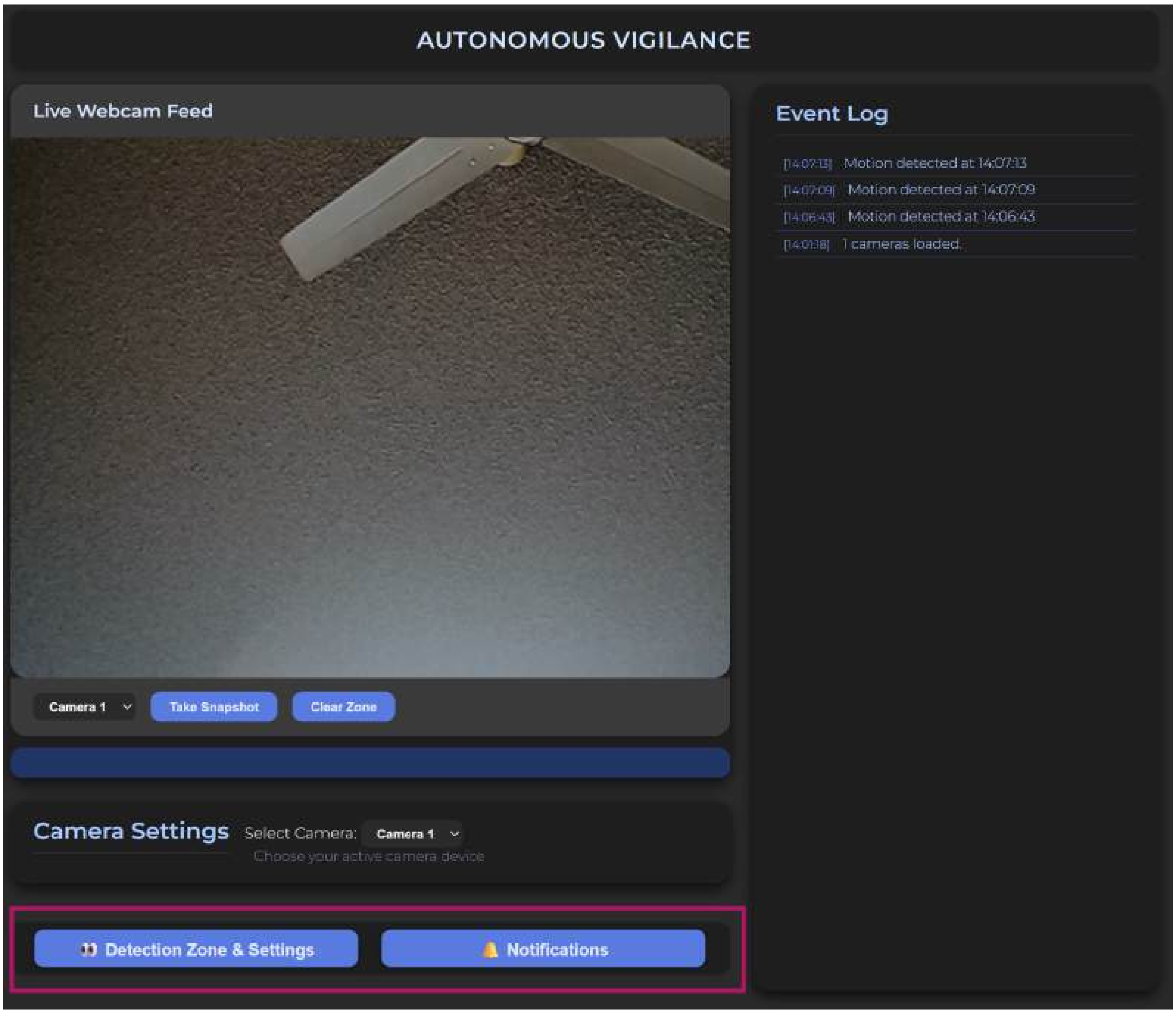
☑ Email Notifications
Email Receivers:
example1@mail.com, example2@mail.com

☐ MQTT Notifications
MQTT notifications will send alerts to your smart home system

Modifications:

The initial prototype of the Autonomous Vigilance Dashboard lays the groundwork with a simple but useful design. It has a dark-themed interface with a basic layout that includes a live webcam feed, key camera controls, and a basic alerts section. The live feed is the main feature, but user interaction is limited to choosing cameras and taking snapshots. The alerts section is there, but it doesn't update in real time, making it less effective at notifying users about important events. This prototype shows the system's potential but lacks user engagement and advanced features, showing the need for improvements to make it more interactive and responsive.

PROTOTYPE 1: Enhanced User Experience



Event Log

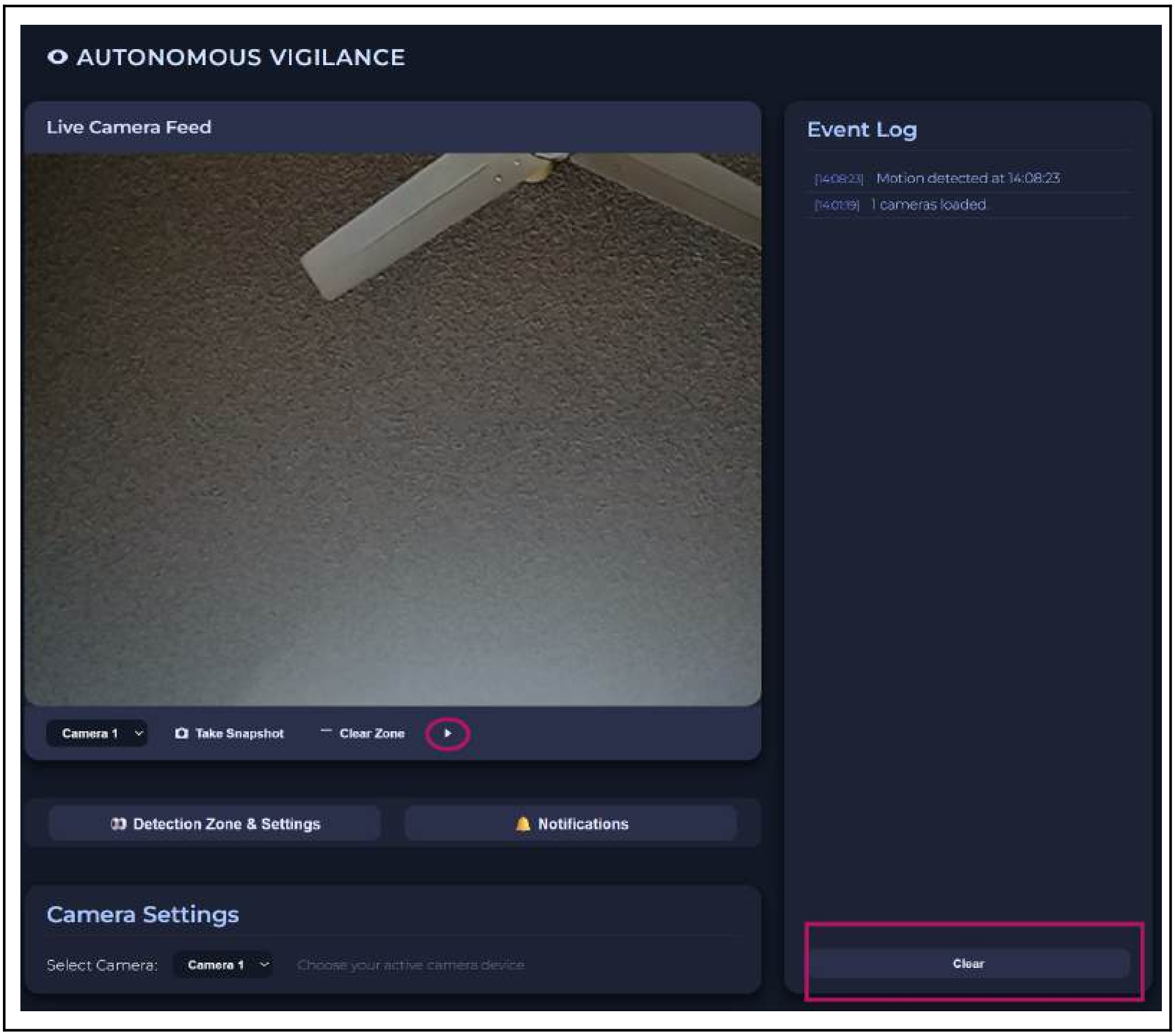
```
[14:07:13] Motion detected at 14:07:13
[14:07:09] Motion detected at 14:07:09
[14:06:43] Motion detected at 14:06:43
[14:01:18] 1 cameras loaded.
```

Select Camera: Camera 1
Choose your active camera device

Modifications:

Building upon the first design, Prototype 1 brings big improvements to how it looks and works, making it easier to use. The interface now has a more consistent colour scheme and a cleaner layout, making it easier to read and more visually appealing. The dashboard has been rearranged so the live feed is clearly separated from the control panels, making navigation simpler. The alerts section is also made more noticeable, ensuring users can easily see important notifications. These changes make the dashboard more user-friendly, helping users interact with it more smoothly while keeping the focus on the live surveillance feed. The improved design is an important step forward in usability, paving the way for future upgrades.

PROTOTYPE 2: Introduction of Tabbed Settings



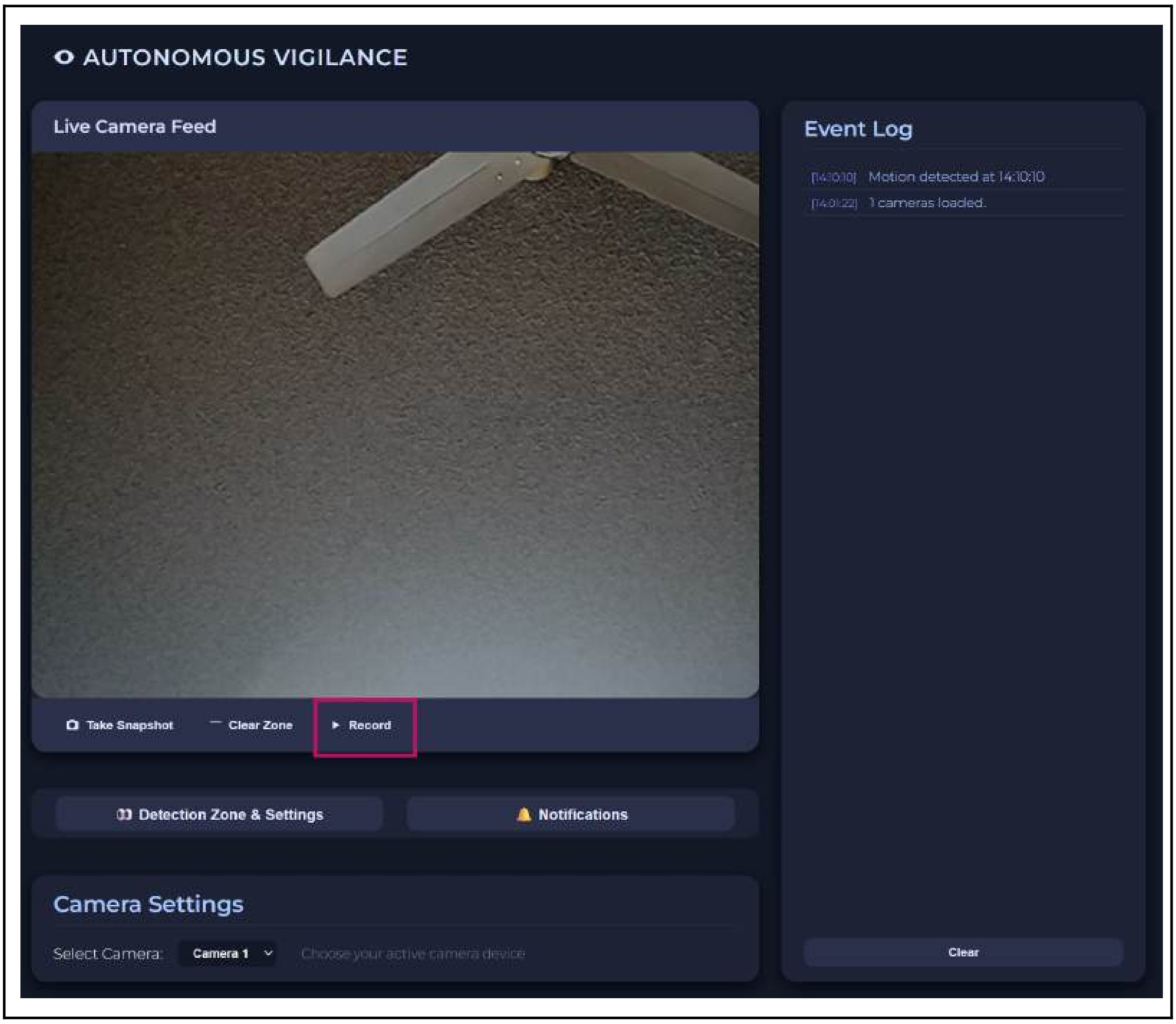
```
[14:08:23] Motion detected at 14:08:23
[14:01:19] 1 cameras loaded
```

Select Camera: Camera 1 Choose your active camera device

Modifications:

Building on the improvements from the earlier versions, Prototype 2 introduces a tabbed approach for the settings interface, allowing users to switch between Camera Settings, Detection Zones, and Notifications in a clean, organised manner. The interface uses toggles to show or hide each section, reducing clutter and streamlining the user workflow. The addition of a recording feature enables users to capture video clips directly from the live feed, controlled by a clear record/pause button. Event log improvements allow users to clear the history easily, helping maintain relevance and readability. Meanwhile, the login modal is refined for clearer input focus and validation feedback. These enhancements collectively improve usability while keeping the dark theme and spacious, functional layout consistent.

PROTOTYPE 3: Dynamic Interaction and Feedback



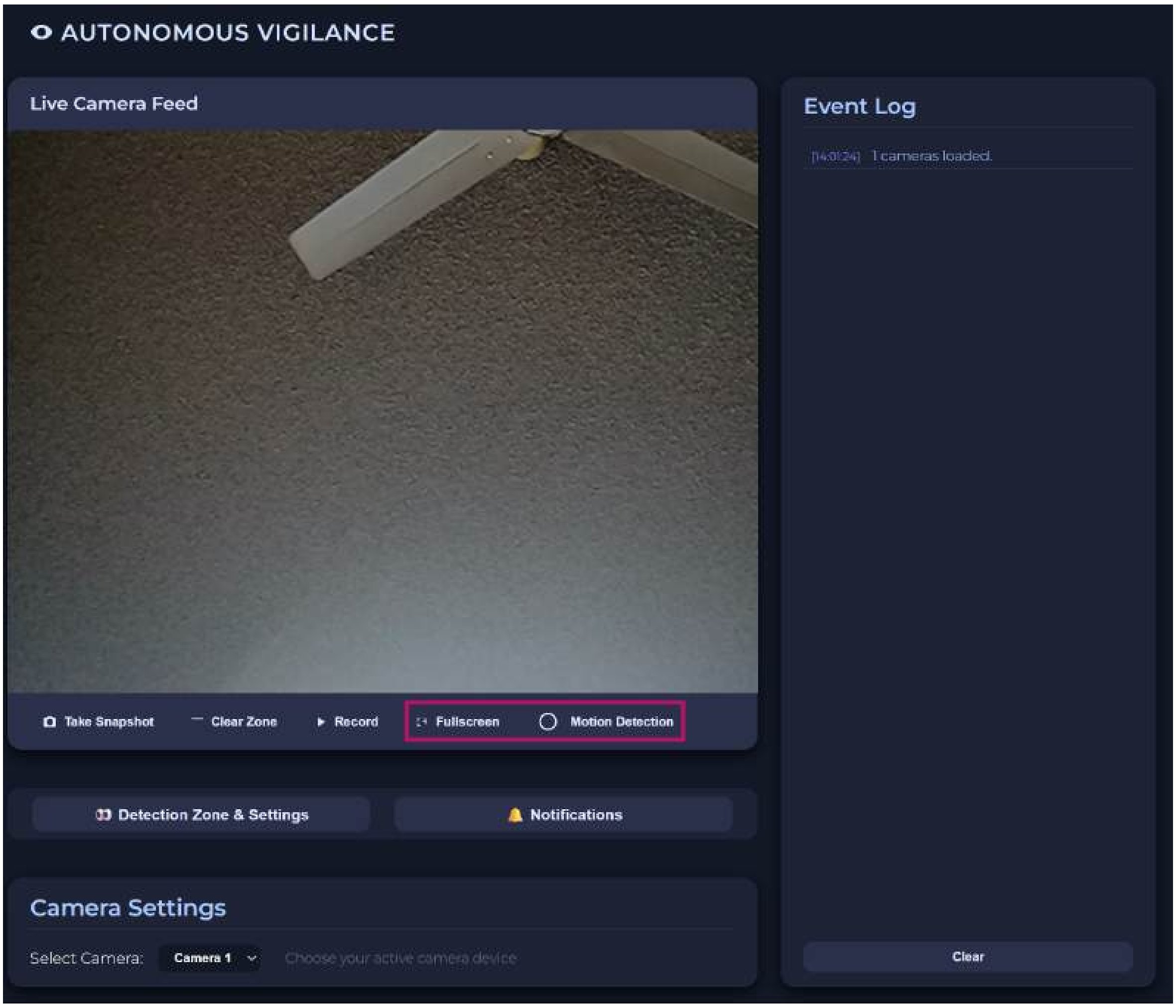
Select Camera: Camera 1 — Choose your active camera device

```
[14:10:10]  Motion detected at 14:10:10
[14:01:22]  1 cameras loaded.
```

Modifications:

Prototype 3 enhances the live feedback experience by enabling real-time motion detection updates that are promptly logged in the event area, fostering better awareness for users. The motion detection toggle button includes a visual indicator reflecting its on/off state, adding clarity to the system’s current operating status. Controls remain grouped thoughtfully beneath the live feed with intuitive icons and smooth hover effects, supporting quick access to key functions like snapshots and recording. Accessibility remains a priority, with ARIA roles and keyboard navigation carefully integrated. These refinements provide users with confident control and immediate situational awareness while preserving the established clean and modern design.

PROTOTYPE 4: Advanced Features and Object Recognition



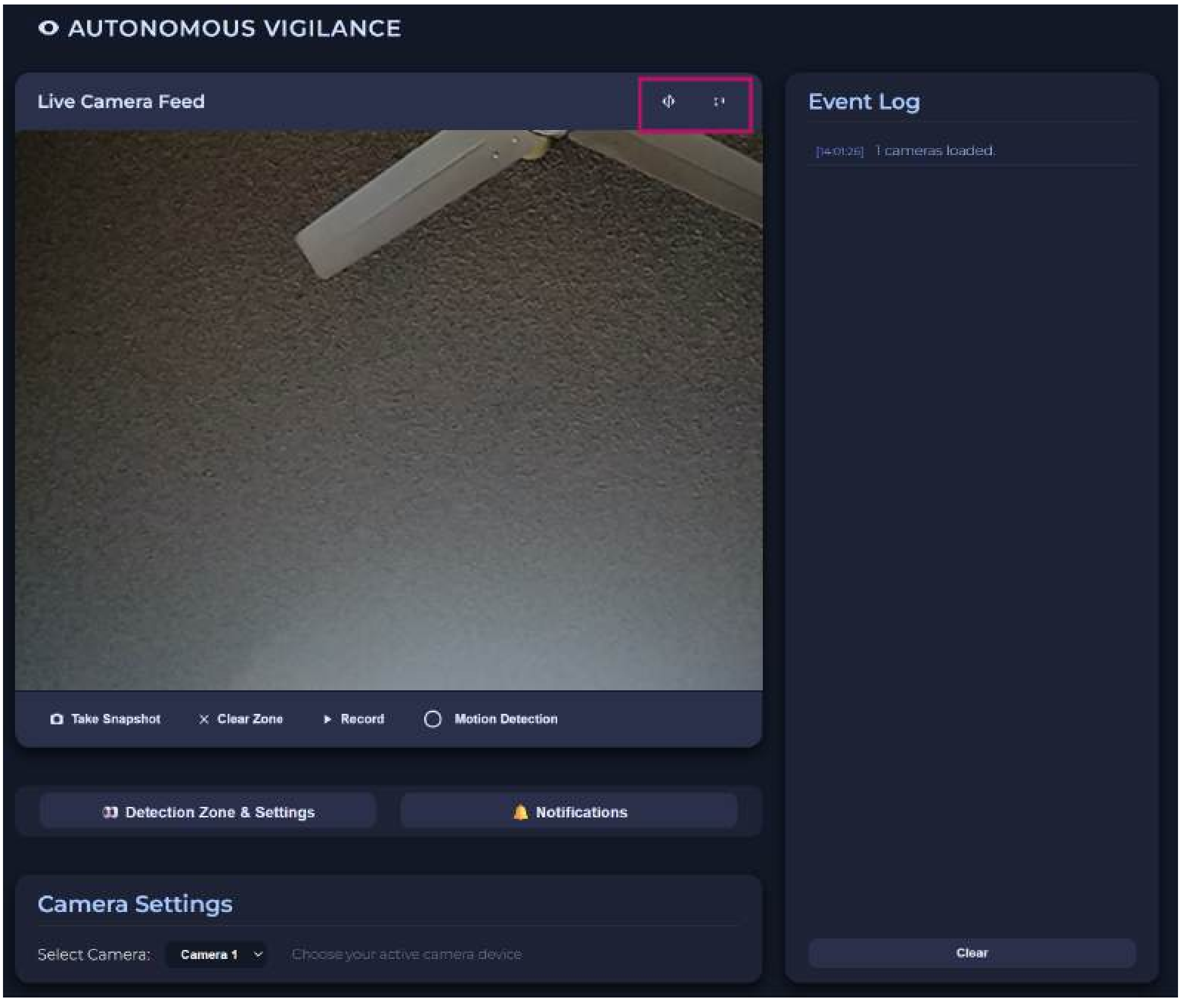
[14:01:24] 1 cameras loaded.

Select Camera: Camera 1 Choose your active camera device

Modifications:

This prototype marks a significant step forward with the introduction of AI-powered object recognition. Users can activate this feature through a dedicated toggle button that highlights when enabled, providing clear visual feedback. The system overlays green bounding boxes and labels on detected objects, accurately scaled and positioned over the video stream to aid recognition. Additional controls, including full-screen mode and camera flipping, offer further flexibility tailored to varied monitoring needs. The design combines a deep blue colour palette that highlights detection overlays effectively without overwhelming the interface. User interactions remain straightforward, supported by clear iconography and responsive tooltips. Overall, these advances position the system as a more intelligent and interactive security tool.

PROTOTYPE 5: Performance and Usability Enhancements



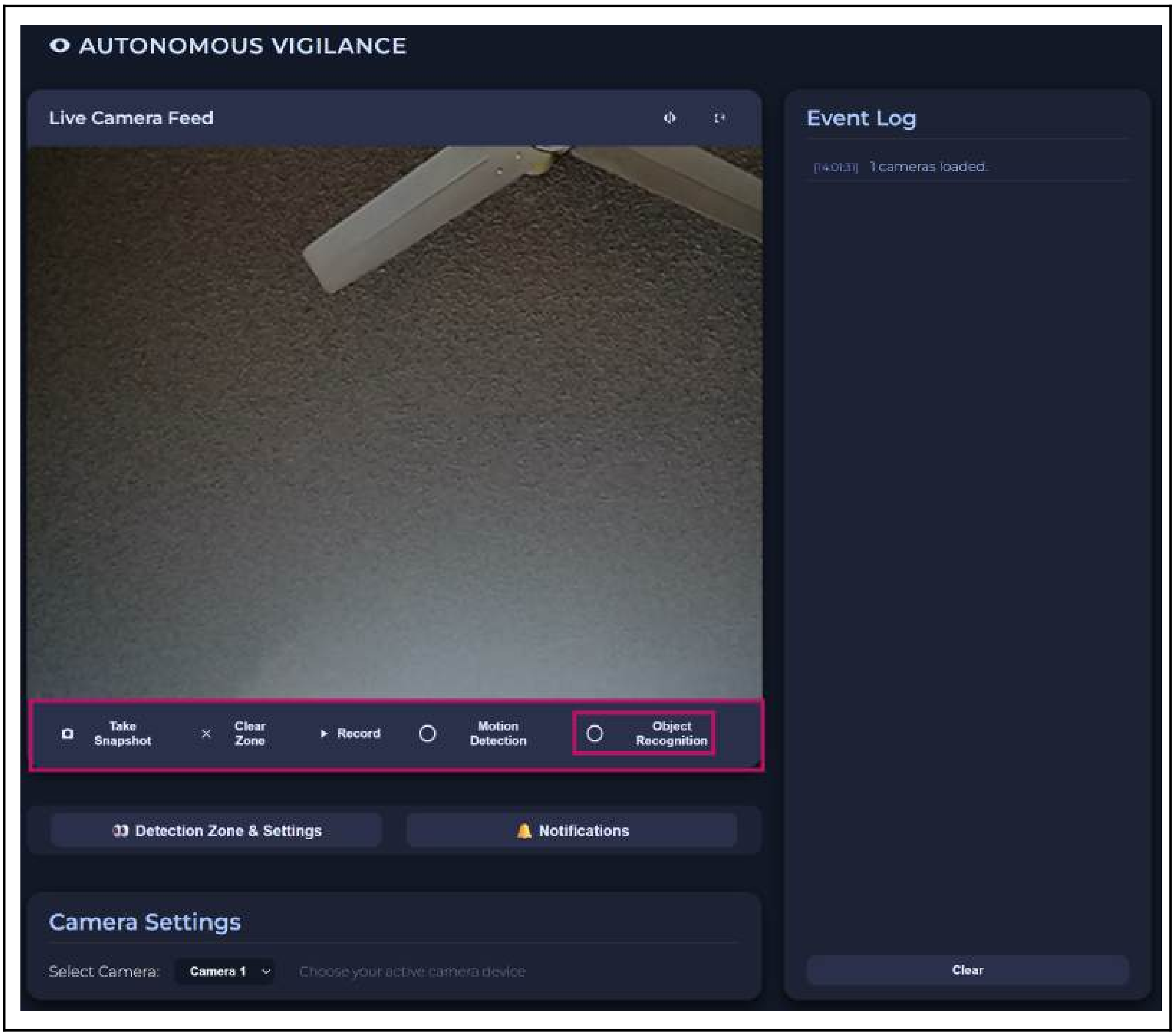
```
[14:01:26]  1 cameras loaded.
```

Select Camera: Camera 1 Choose your active camera device

Modifications

Focusing on refinement, Prototype 5 improves button presentation with subtle shadows and hover transitions that gently draw user attention without distraction. The record button's label and icon toggle seamlessly between "Record" and "Pause," making its status clear at a glance. Video display improvements using object-fit properties ensure consistent, distortion-free feeds across devices and screen sizes. The overlay canvas adapts responsively to full-screen toggles and window resizing, preserving accurate detection zone rendering. These adjustments create a more polished, intuitive experience that balances aesthetics with functionality, enhancing everyday usability without compromising the sleek, minimal style.

PROTOTYPE 6: Final Refinements



Event Log

[14:01:31] 1 cameras loaded.

Camera Settings

Select Camera: Camera 1

Choose your active camera device

Modifications:

The final prototype synthesises all prior improvements into a cohesive and polished interface. The header integrates a subtle logo icon paired with elegant typography, reinforcing brand identity with clean spacing and visual hierarchy. Motion detection and object recognition toggles are distinct and accessible, each with recognisable pressed states to ensure clear user feedback. The overlay canvas supports both detection zones and object highlights simultaneously, adjusting reliably during video or window resizes, including full-screen transitions. The recording and snapshot features operate smoothly with consistent UI responses, complemented by intuitive flip camera and full-screen controls. Performance optimisations keep animations smooth and resource-friendly, while accessibility considerations ensure usability for a broad audience. This iteration embodies a modern, minimalist, and robust design aligned with user needs and professional standards.

Scan the QR code to download the prototypes, or click the link:

https://github.com/lilking2007/Autonomous_Vigilance/tree/main