Fig. 1 SETU logo

Software Development Year 4 Project

Report

Project Title: SiteSafeAI

By: Tomas Smitas
Student No: C00276177
Date: 2025
Supervisor: Dr Oisin Cawley

Fig. 2 SiteSafe Logo

Table of Contents

Introduction

This report presents a comprehensive overview of the development process for the SiteSafeAI web application, carried out as part of my Final Year Project (FYP). It reflects not only the technical work involved but also the personal and professional growth experienced throughout the project. The aim of this document is to demonstrate the skills I have developed, the challenges I encountered, and the lessons I have learned along the way.

Throughout the report, I will provide detailed descriptions of the individual modules within the system, the data structures used, including the database schema, and the tools and technologies that powered the application, such as Flask, React, MySQL, OpenCV, and YOLOv11. Where design choices differed from my initial plans, those changes will be documented along with the reasoning behind them and any additional research that supported those decisions.

This report will also review the major challenges faced during development and how I resolved them, showcasing the problem-solving strategies employed and highlighting areas where mistakes were made and how they became learning opportunities. Some requirements were met successfully, while others evolved or were partially implemented, these will be discussed honestly and in detail.

In addition, I will outline the testing approaches used to assess the reliability and performance of the software and reflect on the learning outcomes of this project. Finally, I will explore what I would do differently if I were to start the project again, identifying ways to improve the development workflow, design planning, or implementation strategies.

The goal of this report is not only to showcase the finished product, but also to demonstrate my ability to navigate a large software development process, gaining real-world experience, learning from setbacks, and emerging as a more capable software developer.

Application Overview

SiteSafeAI is an AI-powered safety monitoring web application designed to enhance compliance with personal protective equipment (PPE) policies, specifically helmet usage, on construction sites. The system processes real-time video feeds from surveillance cameras and applies object detection to identify individuals and determine whether they are wearing helmets. Its goal is to reduce safety violations by providing automated, real-time feedback and alert mechanisms for site supervisors.

The application is built using a modern full-stack architecture. The backend is developed with Flask, which handles video frame routing, model inference using the YOLOv11 object detection algorithm. The frontend, developed with React, presents a responsive, user-friendly interface for live video viewing, system configuration, and user/camera management. OpenCV is used for handling video streams, and MySQL serves as the database solution.

Key Features

1. Real-time Helmet Detection

At the core of SiteSafeAI is a helmet detection model trained using YOLOv11. The model is capable of identifying individuals in video frames and classifying them as either wearing or not wearing helmets. The system operates at approximately 2 inferences per second, ensuring real-time detection that is fast enough to monitor active environments without significant delay.

2. User Roles and Access Control

The system supports two types of users:

- *Admin Users:* Have full control over the platform, including access to all video streams, system settings, and user management. Admins can create, update, or delete other user accounts and modify system-wide configurations such as notification preferences.
- *Normal Users:* Can view video streams and access detection data relevant to their assigned company or site but cannot modify other user accounts or system settings.

This role-based access control ensures that only authorised personnel can make sensitive changes, helping to maintain data integrity and system security.

3. Configurable Email Notification System

SiteSafeAI includes an automated email alert system designed to notify users of safety violations. Admin users can enable or disable email notifications per company and customise the recipient email addresses directly through the interface. This flexibility allows organisations to adapt the notification system to their internal communication workflows. When enabled, the system sends an email alert whenever a person without a helmet is detected on camera.

4. Company-Based Multi-Tenancy

Each user and camera is associated with a specific company, allowing the system to support multiple companies operating independently under a shared platform. This structure enables scalability, where additional companies can be added with their own users, cameras, and settings without interfering with others.

5. Camera Management

Admin users can register and configure cameras by providing metadata such as camera name and location. This helps organise large ranges of cameras and ensures that detections are linked to the correct site areas for efficient monitoring.

6. Adjustable Confidence Threshold

Each company has a confidence threshold setting that determines the model's sensitivity to helmet detection. This setting can be adjusted in the admin dashboard to balance accuracy with false positive/negative rates depending on the site's needs.

7. Clean and Functional Interface

The React frontend is designed with usability in mind. It includes a live feed section for viewing real-time detections, a dashboard for managing settings and thresholds, and a user management interface for admins. The layout and visual components prioritise clarity and ease of use, even for non-technical users.

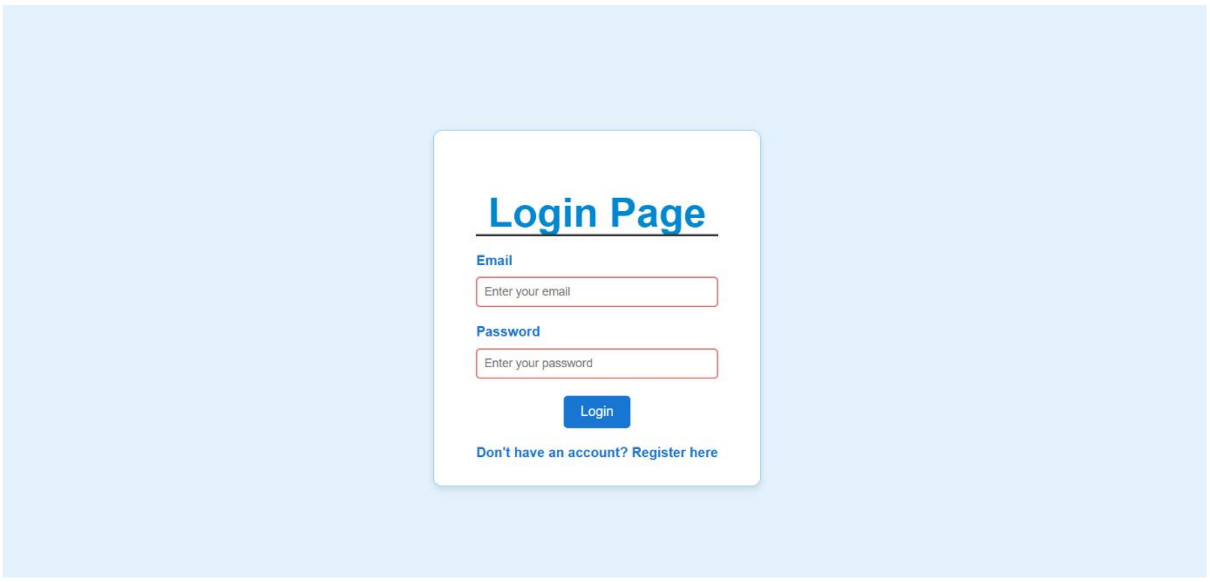
System Workflow Summary

1. Video feeds are captured by frontend and send to the backend.
2. Frames are passed to the YOLOv11 model for inference.
3. Detection results (helmet/no helmet/person) are overlaid on the frame.
4. Processed frames are sent back to the frontend for real-time display.
5. If a violation is detected and email alerts are enabled, a notification is sent to the configured email address.
6. All detections are logged and associated with the respective company, camera, and timestamp.

Process overview

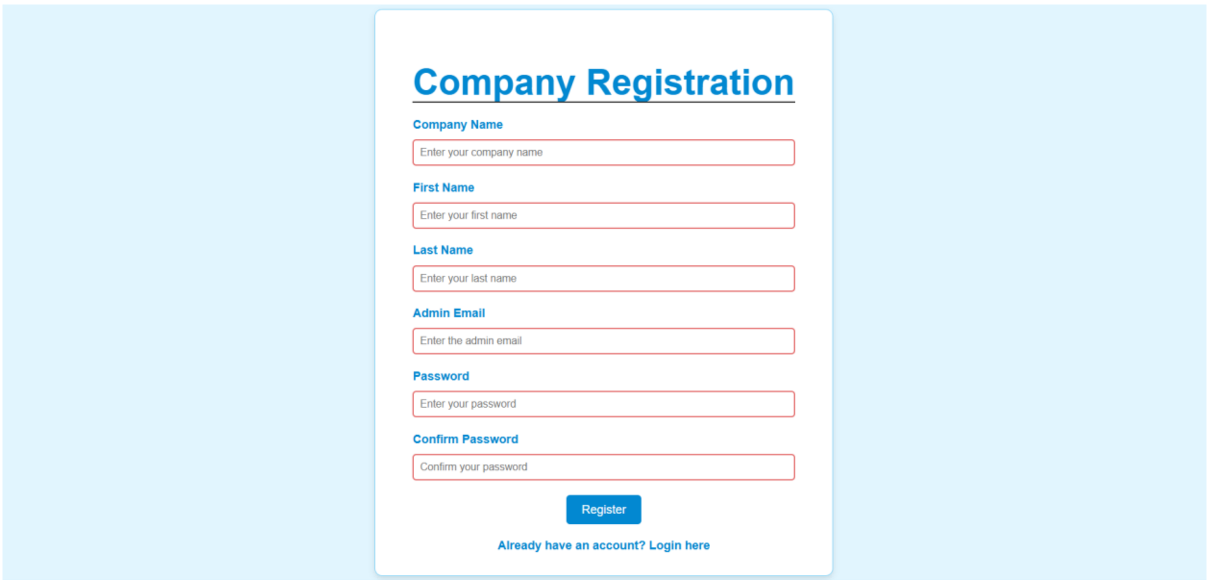
Login Page

Initially, when the user accesses the application, if they haven't logged in before, they are directed to the login screen. If they have a user account linked to a company, they can log in here. Alternatively, they can navigate to the registration screen to register a new company along with an initial admin user. Both normal users and admin users use the same screen to log in.



Company Registration

If the “Don’t have an account? Register here” button is pressed, the user is taken to a screen where they can create a new company and an initial admin account. From there, the user can log in as the admin and manage the accounts of everyone.

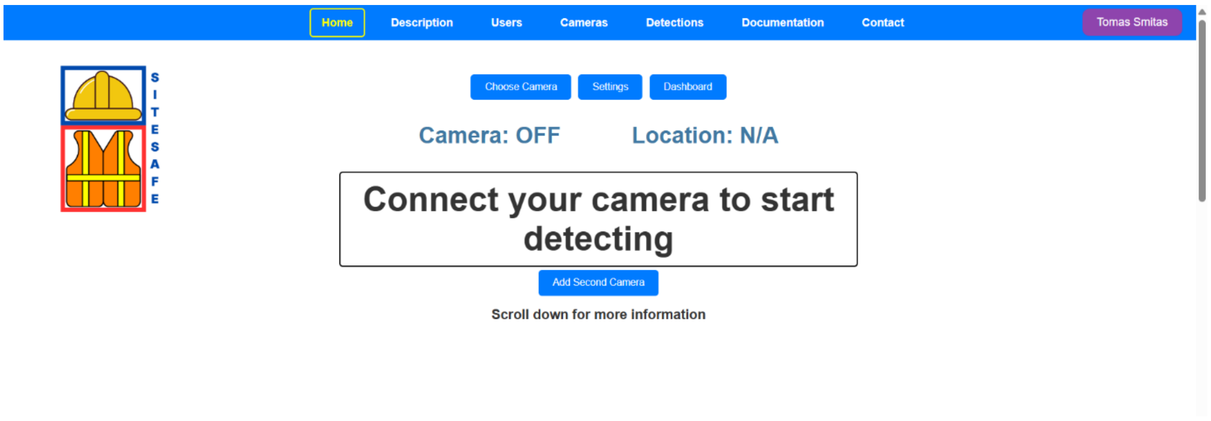


Main Page Description

Once the user logs in, they are brought to the main screen, which contains the majority of the options available to them. The main screen features a clean and simple design, eliminating the need for complicated user manuals and costly training for the company. All buttons are intuitive and clearly labelled, giving the user a clear understanding of their function.

At the top, there is a menu with all the essential options the user may need, such as managing users, cameras, and detections. There are also options to view the documentation, read a project description, and contact the creator. On the far right of the menu, the logged-in user's name is displayed. When clicked, it reveals an option to log out.

In the main section, the application's logo is displayed in the top left. In the center, users can see options like "Choose Camera," "Settings," "Dashboard," and "Add Second Camera," all of which will be explained in more detail.



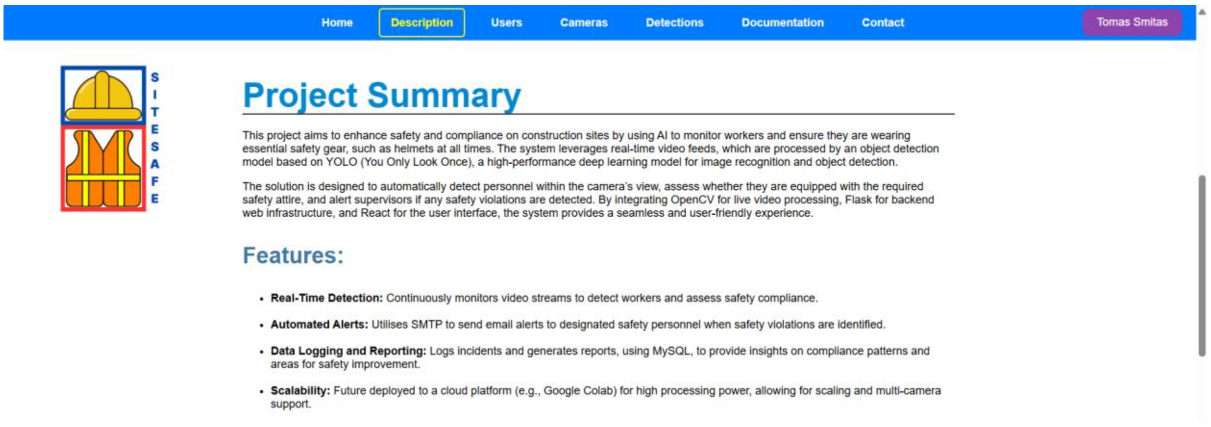
Camera: OFF Location: N/A

Connect your camera to start detecting

Scroll down for more information

Project Summary

The Description menu option brings the user to a simple overview of the description of the application and the technologies used to make the application work.



Project Summary

This project aims to enhance safety and compliance on construction sites by using AI to monitor workers and ensure they are wearing essential safety gear, such as helmets at all times. The system leverages real-time video feeds, which are processed by an object detection model based on YOLO (You Only Look Once), a high-performance deep learning model for image recognition and object detection.

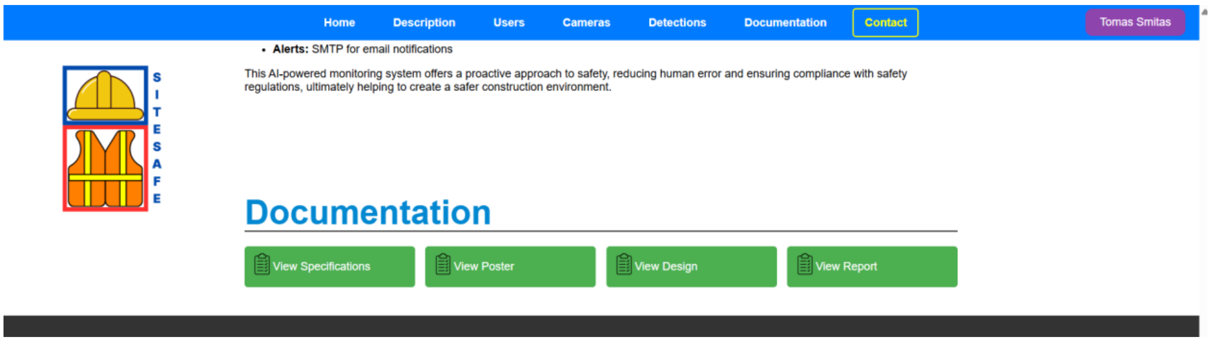
The solution is designed to automatically detect personnel within the camera's view, assess whether they are equipped with the required safety attire, and alert supervisors if any safety violations are detected. By integrating OpenCV for live video processing, Flask for backend web infrastructure, and React for the user interface, the system provides a seamless and user-friendly experience.

Features:

- **Real-Time Detection:** Continuously monitors video streams to detect workers and assess safety compliance.
- **Automated Alerts:** Utilises SMTP to send email alerts to designated safety personnel when safety violations are identified.
- **Data Logging and Reporting:** Logs incidents and generates reports, using MySQL, to provide insights on compliance patterns and areas for safety improvement.
- **Scalability:** Future deployed to a cloud platform (e.g., Google Colab) for high processing power, allowing for scaling and multi-camera support.

Documentation

The documentation section is used to give the users access to the pdf documents for the Specifications, Poster, Design and Report documents.



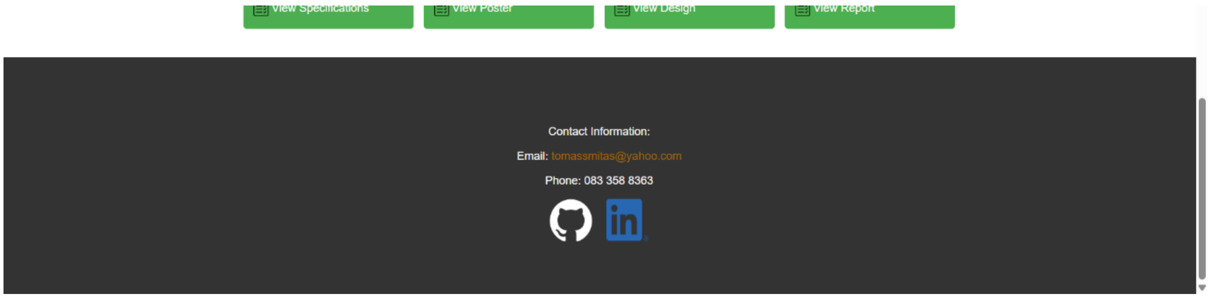
- **Alerts:** SMTP for email notifications

This AI-powered monitoring system offers a proactive approach to safety, reducing human error and ensuring compliance with safety regulations, ultimately helping to create a safer construction environment.

Documentation

Contact

The contact section is used to be able to contact the creator for any support associated with the application. There are multiple ways to make contact like email, phone, github and linkedIn. The users may contact the creator for any queries.



Contact Information:

Email: tomassmitas@yahoo.com

Phone: 083 358 8363

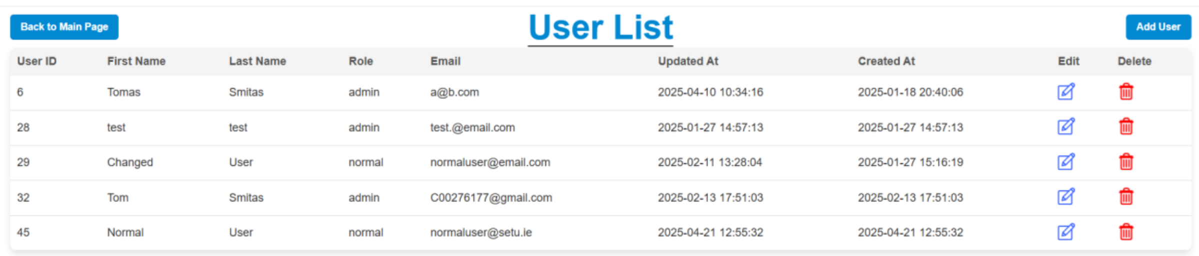
User List

The next major section of the application is the Users section, which is accessible only to admin users. Normal users do not have access to this page.

Upon entering the Users page, the admin is presented with a table displaying all users associated with their company, including their own account. From here, the admin can manage all users within the company. They can view detailed information for each user, including other admin users.

Each row in the table has two buttons on the right-hand side: an Edit button (in blue), which allows the admin to update user details, and a Delete button (in red), which removes the corresponding user.

At the top of the page, there are two additional buttons. The first, located on the left, allows the admin to return to the main page. The second, on the right, is used to add a new user, either a normal user or another admin.



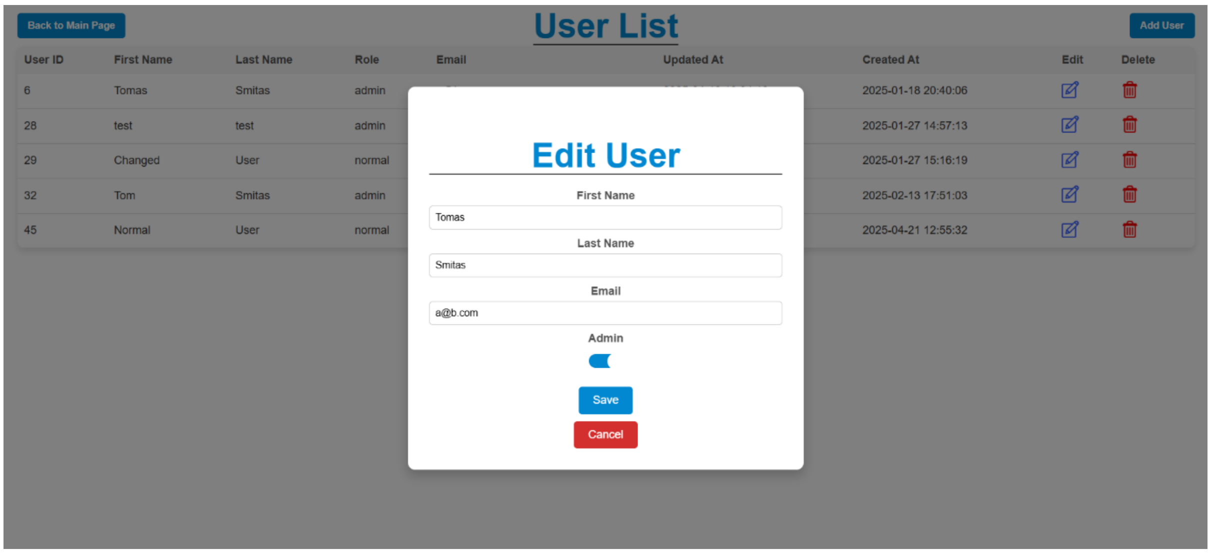
User List

User ID	First Name	Last Name	Role	Email	Updated At	Created At	Edit	Delete
6	Tomas	Smitas	admin	a@b.com	2025-04-10 10:34:16	2025-01-18 20:40:06	✎	🗑
28	test	test	admin	test.@email.com	2025-01-27 14:57:13	2025-01-27 14:57:13	✎	🗑
29	Changed	User	normal	normaluser@email.com	2025-02-11 13:28:04	2025-01-27 15:16:19	✎	🗑
32	Tom	Smitas	admin	C00276177@gmail.com	2025-02-13 17:51:03	2025-02-13 17:51:03	✎	🗑
45	Normal	User	normal	normaluser@setu.ie	2025-04-21 12:55:32	2025-04-21 12:55:32	✎	🗑

Edit User

When the Edit button is clicked on a user row, a user-friendly modal window appears. If the Cancel button is clicked or if the user clicks outside the modal, it will close without saving any changes, returning the admin to the user list.

Within the modal, the admin can update the user's First Name, Last Name, Email, and Access Status. Changes are only saved when the Save button is clicked. Upon saving, the updated information is stored in the database, and the admin is redirected back to the user list page.



User ID	First Name	Last Name	Role	Email	Updated At	Created At	Edit	Delete
6	Tomas	Smitas	admin			2025-01-18 20:40:06		
28	test	test	admin			2025-01-27 14:57:13		
29	Changed	User	normal			2025-01-27 15:16:19		
32	Tom	Smitas	admin			2025-02-13 17:51:03		
45	Normal	User	normal			2025-04-21 12:55:32		

Edit User

First Name: Tomas

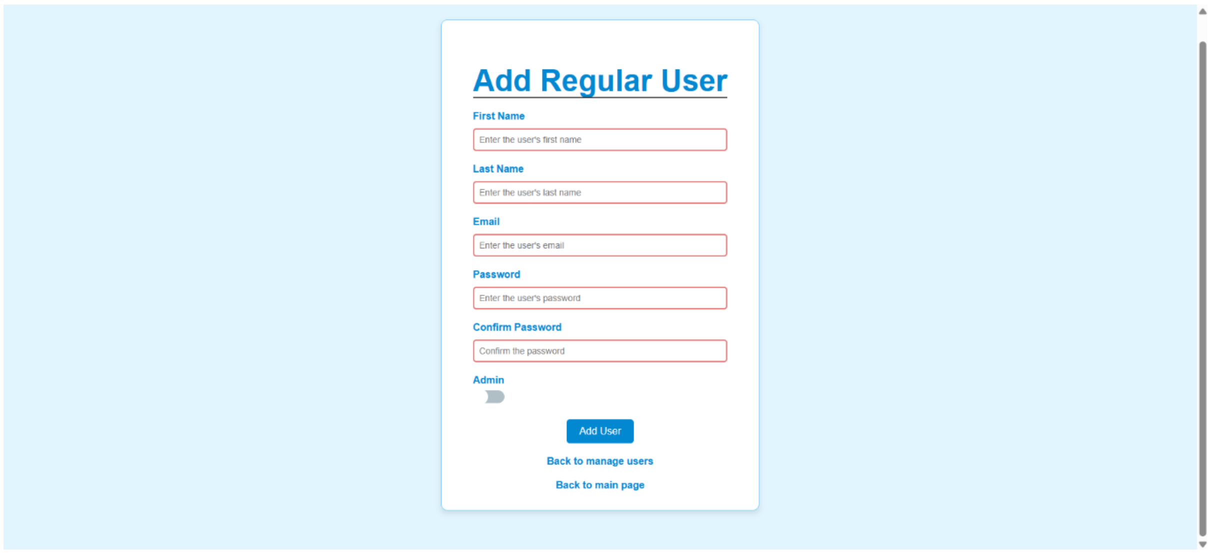
Last Name: Smitas

Email: a@b.com

Admin

Add User

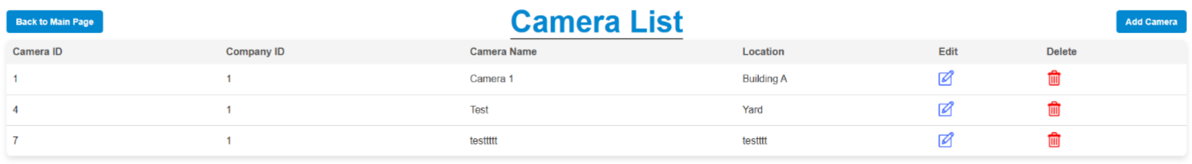
When the Add User button is clicked, the admin is taken to the Add User page. After entering all the required details, a new user is created and added to the database under the current company. The page also provides two navigation options: one to return to the Main Screen, and another to go back to the User List.



Camera List

The next major section is the Camera section. Here, the admin user can create, view, update, and delete cameras. These cameras are later displayed in a list when connecting to a camera, allowing the user to select and connect to one from the table.

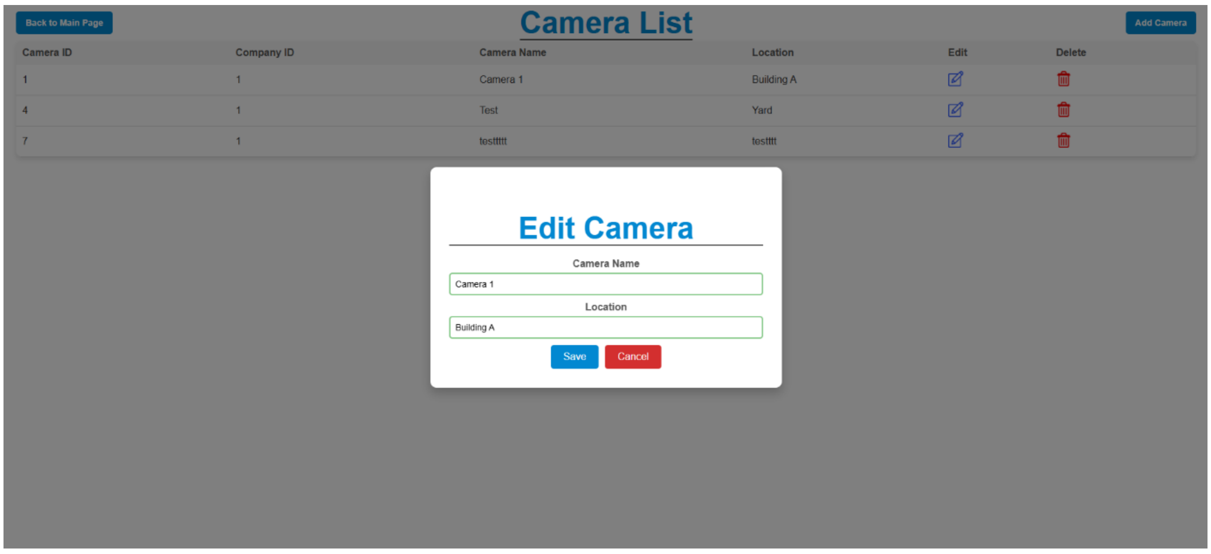
Similar to the Users screen, there are two buttons at the top: the left button navigates back to the Main Page, and the right button is used to add a new camera. Each camera entry in the table also includes Edit and Delete buttons on the right-hand side, which correspond to the actions for that specific camera.



Camera ID	Company ID	Camera Name	Location	Edit	Delete
1	1	Camera 1	Building A		
4	1	Test	Yard		
7	1	testtttt	testttt		

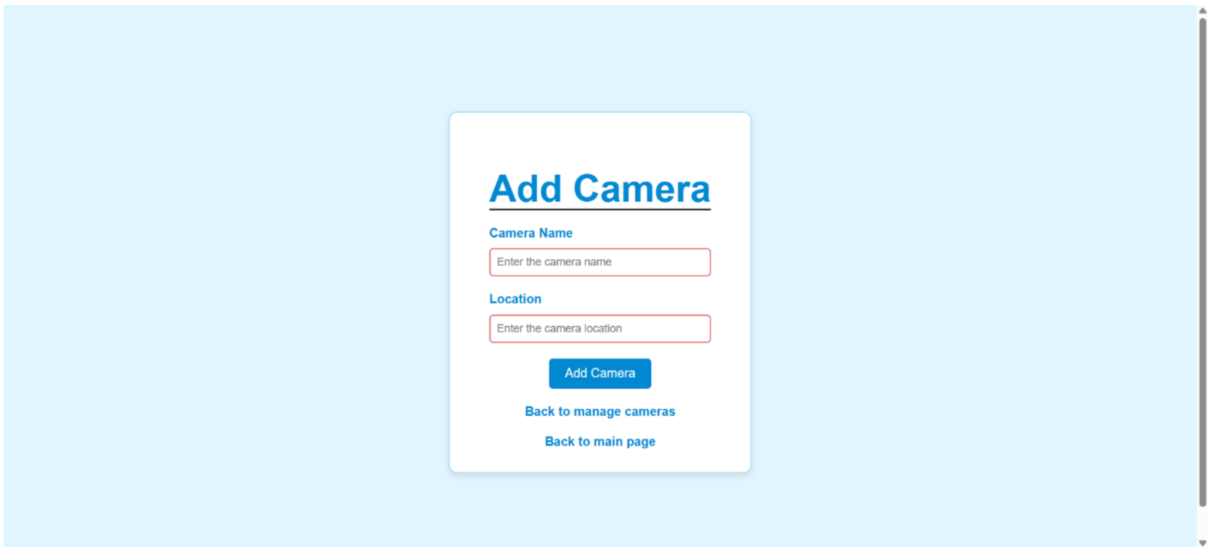
Edit Camera

The edit button allows the user to edit the following fields. Camera name and location that the camera is placed.



Add Camera

The add camera page only requires the user to enter in two fields. The camera name and the location.

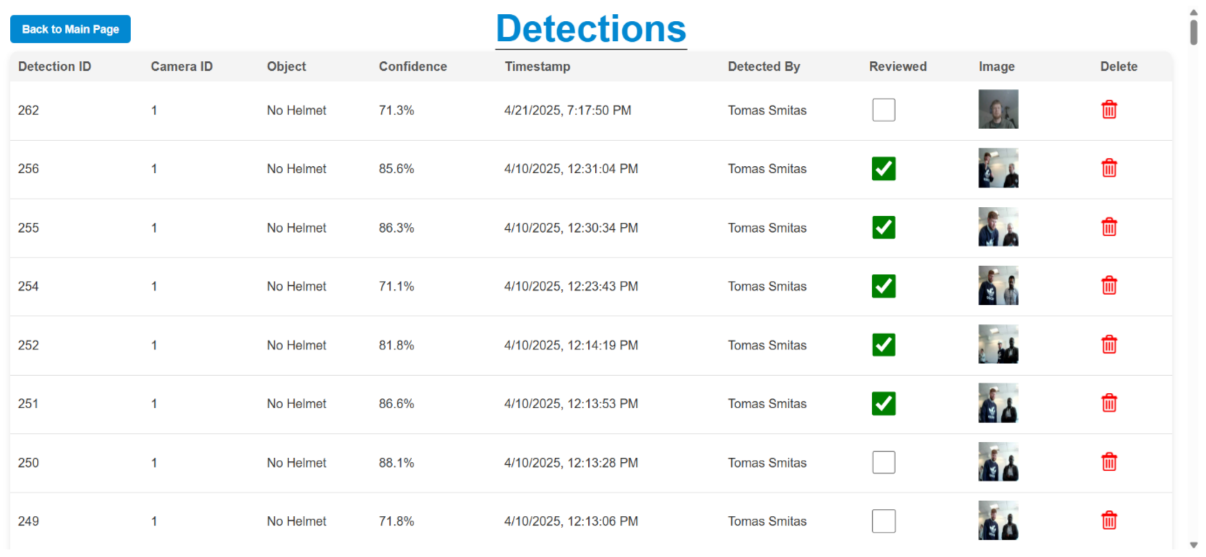


Detections

The next major section is the Detection page. This section differs slightly from the Users and Camera sections. There is no option to manually add a detection, detections are generated automatically by the system whenever a no-helmet incident is detected.

Each row in the detection table includes some unique features. On the far right, there is a Delete button, allowing the user to remove any false detections. Just before the delete button is the Image column. Clicking on an image enlarges it to help with review. The Review column contains a checkbox that the user can tick or untick to indicate whether the detection has been reviewed.

This review process can be tailored to each company's workflow. For example, one company might require a visual check of the image before marking it as reviewed, while another might require reviewing the automatic email notification as part of the process.

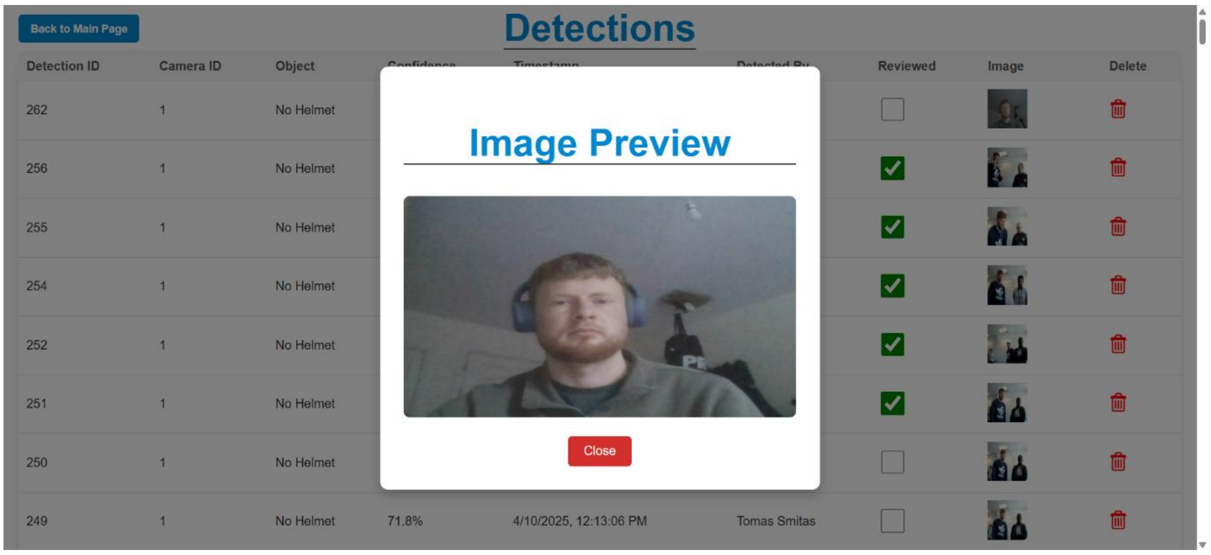


Detections

Detection ID	Camera ID	Object	Confidence	Timestamp	Detected By	Reviewed	Image	Delete
262	1	No Helmet	71.3%	4/21/2025, 7:17:50 PM	Tomas Smitas	☐		🗑
256	1	No Helmet	85.6%	4/10/2025, 12:31:04 PM	Tomas Smitas	✓		🗑
255	1	No Helmet	86.3%	4/10/2025, 12:30:34 PM	Tomas Smitas	✓		🗑
254	1	No Helmet	71.1%	4/10/2025, 12:23:43 PM	Tomas Smitas	✓		🗑
252	1	No Helmet	81.8%	4/10/2025, 12:14:19 PM	Tomas Smitas	✓		🗑
251	1	No Helmet	86.6%	4/10/2025, 12:13:53 PM	Tomas Smitas	✓		🗑
250	1	No Helmet	88.1%	4/10/2025, 12:13:28 PM	Tomas Smitas	☐		🗑
249	1	No Helmet	71.8%	4/10/2025, 12:13:06 PM	Tomas Smitas	☐		🗑

Image Preview

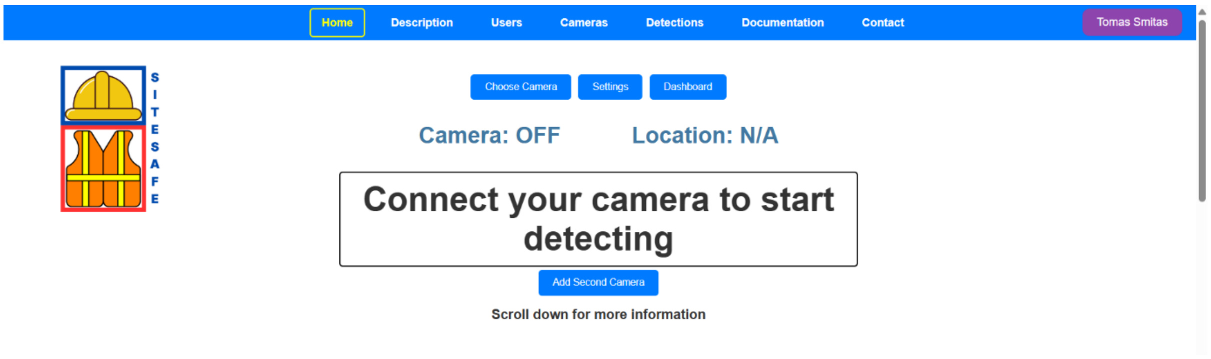
The enlarged image view provides a clearer, zoomed-in version of the detection image. This allows the reviewer to more easily identify the infraction and the person or people involved.



Detection ID	Camera ID	Object	Confidence	Timestamp	Detected By	Reviewed	Image	Delete
262	1	No Helmet				☐		
256	1	No Helmet				✓		
255	1	No Helmet				✓		
254	1	No Helmet				✓		
252	1	No Helmet				✓		
251	1	No Helmet				✓		
250	1	No Helmet				☐		
249	1	No Helmet	71.8%	4/10/2025, 12:13:06 PM	Tomas Smitas	☐		

Main Page Functionality

Returning to the Main Functionality screen, the primary features and options will now be explored and explained in more detail.



Camera: OFF Location: N/A

Connect your camera to start detecting

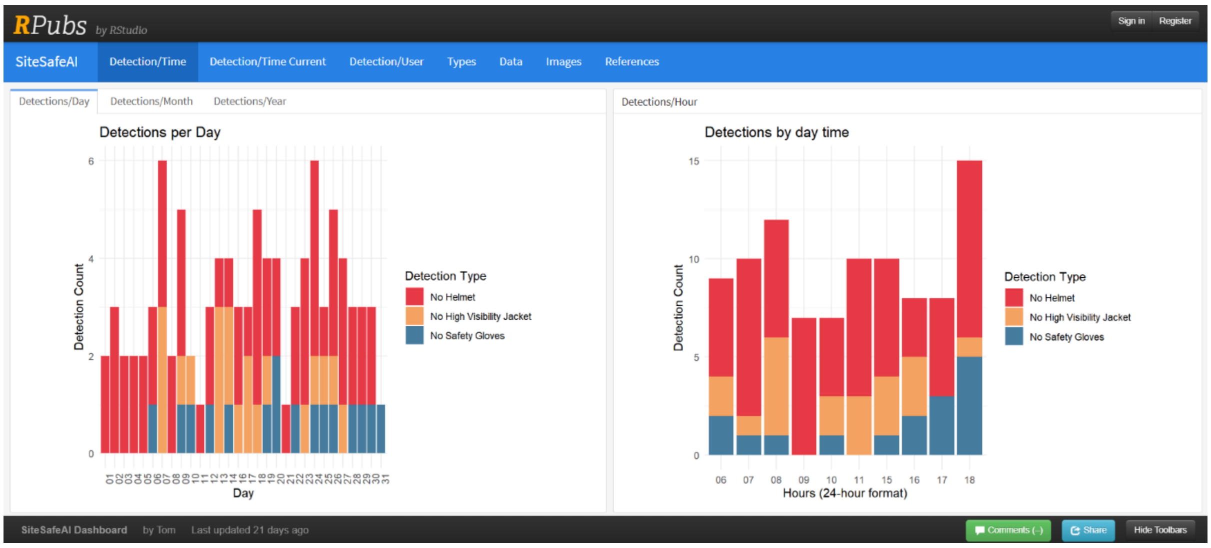
Scroll down for more information

Dashboard

Among the main options, the Dashboard button is located on the right. When clicked, it opens an example dashboard designed to showcase features planned for a future update. This dashboard is built using realistic, generated data.

The dashboard displays various graphs and tabs, including comparisons such as detections per day, month, and year, as well as detections per user ID. Additional features include views of both raw and processed data, and the last detected image for each camera.

The example dashboard also highlights planned future capabilities, such as high-visibility jacket detection and safety glove detection, representing possible extensions of the safety system.

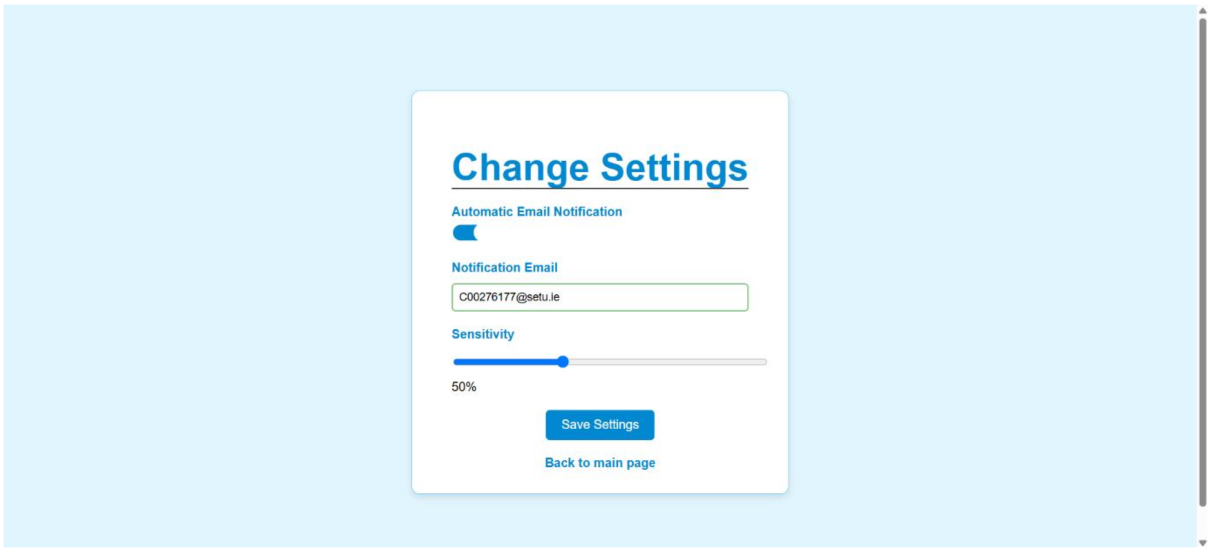


Change Settings

Moving on to the Settings button, this option provides important controls to adjust the automatic email service and detection sensitivity. The Settings button is only available to the admin user.

The first toggle, when turned on, reveals the "Notification Email" field, enabling the automatic email service. The admin can then specify the recipient of the safety emails, typically a safety officer or manager. If the toggle is switched off, the "Notification Email" section is hidden, and the automatic email service is disabled for that company.

Below the email field, the sensitivity slider allows the admin to fine-tune the detection sensitivity. The slider's range is from 30% to 90%, with 50% being the optimal setting based on testing. Values outside this range resulted in significantly diminished of performance.

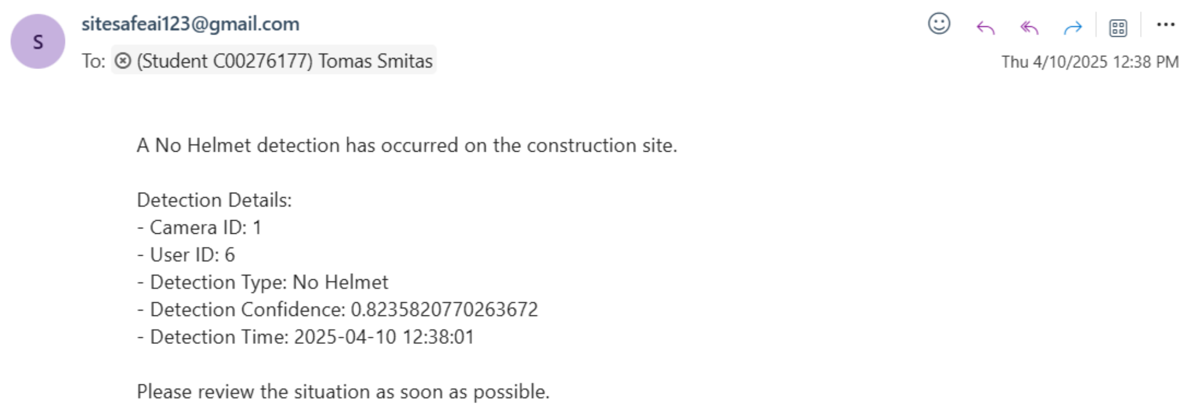


Notification Email: C00276177@setu.ie

Sensitivity: 50%

Email

Once a detection is recorded, it is added to the detection list, and an automatic email is triggered. The email settings are configured by the admin through the settings options. The automatic email contains essential details about the detection and notifies the person responsible for reviewing it, alerting them that a new detection has been added to the table. Below are the details included in the email.



sitesafeai123@gmail.com
To: (Student C00276177) Tomas Smitas
Thu 4/10/2025 12:38 PM

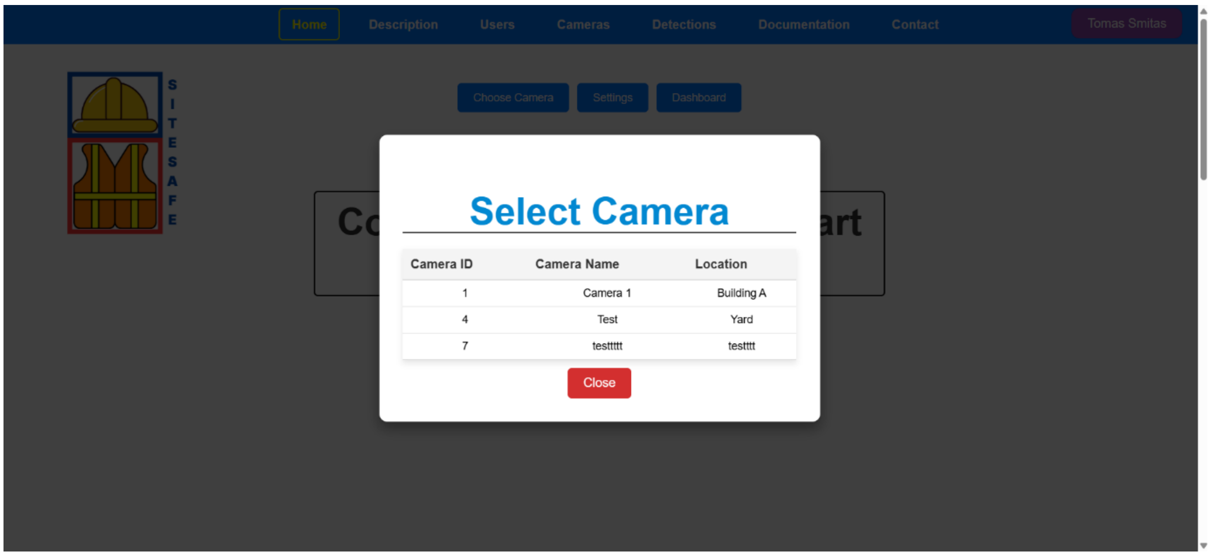
A No Helmet detection has occurred on the construction site.

Detection Details:
- Camera ID: 1
- User ID: 6
- Detection Type: No Helmet
- Detection Confidence: 0.8235820770263672
- Detection Time: 2025-04-10 12:38:01

Please review the situation as soon as possible.

Select Camera

Now, the main feature, the “Choose Camera” button. When clicked, it displays a table of available cameras. The user can then select the camera they wish to connect to by clicking on the corresponding row, and the camera will be connected.



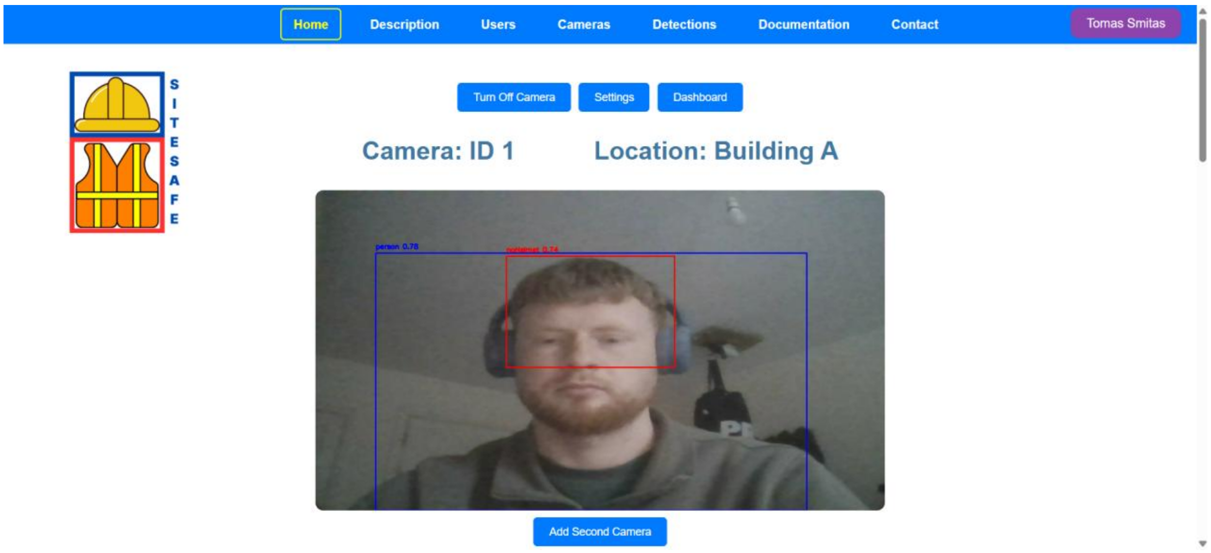
Select Camera

Camera ID	Camera Name	Location
1	Camera 1	Building A
4	Test	Yard
7	testtttt	testttt

Main Page 1 Camera

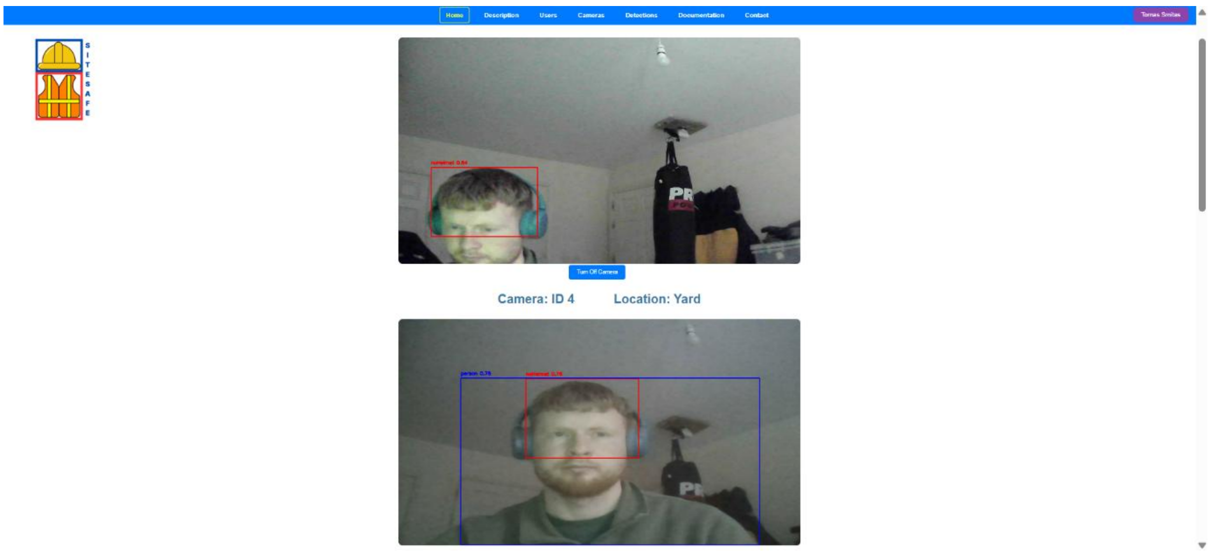
Once a row is clicked, the camera automatically connects, and detection begins. When a no-helmet detection occurs, the system records a detection based on the last 8 out of 10 frames.

Below the image detection, there's an “Add Second Camera” button, allowing the user to connect an additional camera.



Main Page 2 Cameras

When the Add Second Camera button is clicked, a table of available cameras is displayed again, allowing the user to select another camera to connect. With both cameras connected, they can now run inference and record detections of infractions simultaneously.



Email Service

The email notification system was successfully implemented as part of the SiteSafeAI project to support real-time alerting for safety violations. Once an incident is detected, such as a worker not wearing a helmet, the system automatically sends an email to the relevant personnel, such as the site manager or designated safety officer.

The email is dynamically generated using detection data pulled from the database, including key details such as the detection timestamp and information about the camera that recorded the violation. Although the final version of the email does not currently include the detection image due to technical constraints, it still provides all critical context needed for prompt action.

SMTP (Simple Mail Transfer Protocol) was used to manage the delivery of these emails. The Flask backend generates and sends the email as soon as a detection is confirmed by the algorithm. During development, the system was tested rigorously to ensure that emails matched the corresponding database entries for accuracy and reliability.

One challenge encountered was network restrictions on the college Wi-Fi, which blocked outgoing emails. To overcome this, a mobile hotspot was used during demos and testing phases, ensuring that the service could still function in a live environment.

This feature strengthens the project's goal of real-time safety monitoring by providing immediate and automated communication to relevant personnel, enabling swift responses to unsafe behaviours on construction sites.

Spam Protection

To ensure the alert system remains effective and user-friendly, a spam protection mechanism was successfully implemented as part of the final SiteSafeAI application. This system prevents users from being overwhelmed by redundant or excessive notifications triggered by repeated detections of the same type of violation within a short period.

A cooldown period mechanism was introduced to address this issue. Once an alert is triggered for a specific violation, such as a worker not wearing a helmet, the system initiates a cooldown period of 30 seconds. During this interval, no additional email alerts are sent for the same violation from the same camera, even if the detection persists or recurs. This significantly reduces notification noise and ensures only the most relevant and actionable alerts are sent.

In addition, the system was designed with the potential for future enhancements, such as aggregated reporting. While not implemented in this version, the system structure supports batching multiple similar detections into a single summarised alert, reducing clutter in the recipient’s inbox and improving the efficiency of the alerting process.

This spam protection feature played an important role in refining the alert system, improving overall usability, and ensuring that personnel such as safety officers or site managers receive notifications that are timely, relevant, and easy to act upon, without being overwhelmed.

Database

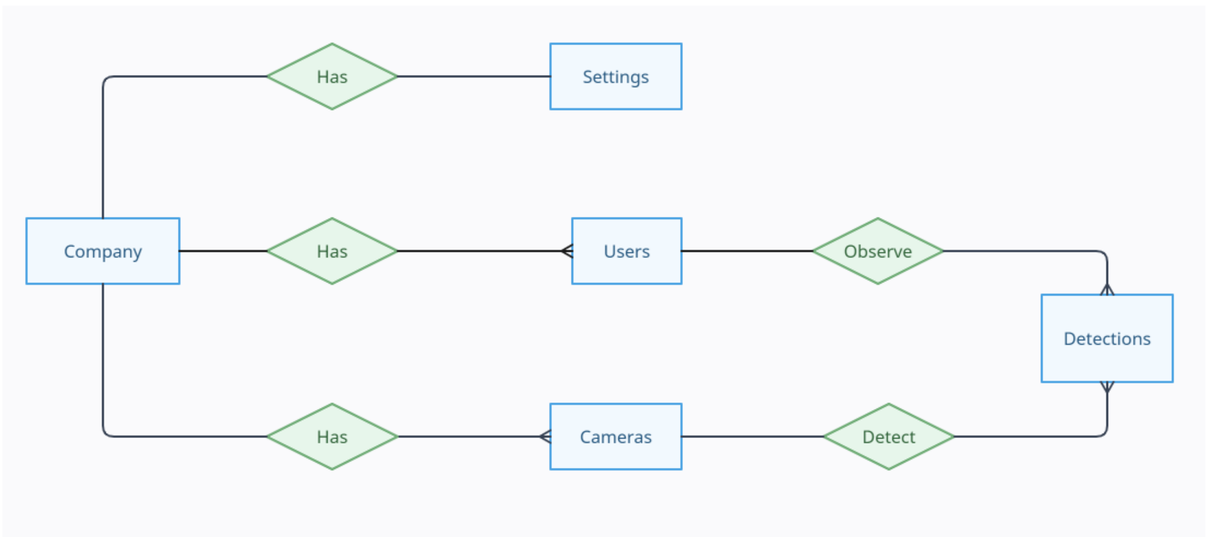
The database is a critical component of the Construction Safety Monitoring System. It will store key information related to video stream monitoring, object detection results, and user interactions with the system.

MySQL was selected as the database solution for the Construction Safety Monitoring System due to its robust performance, reliability, and widespread use in the industry.

MySQL offers high scalability, ease of integration with Python and other web technologies, and strong support for complex queries, which are essential for storing and analysing real-time detection data. Additionally, MySQL's support for Atomicity, Consistency, Isolation, and Durability (ACID) compliance ensures data consistency and integrity, making it an ideal choice for managing critical safety-related information in the system.

The database will facilitate the tracking of safety violations, generate detailed dashboards, and support real-time alerts. Below is the Entity Relationship Diagram (ERD) that reveals the database layout and key directions.

Entity Relationship Diagram (ERD)



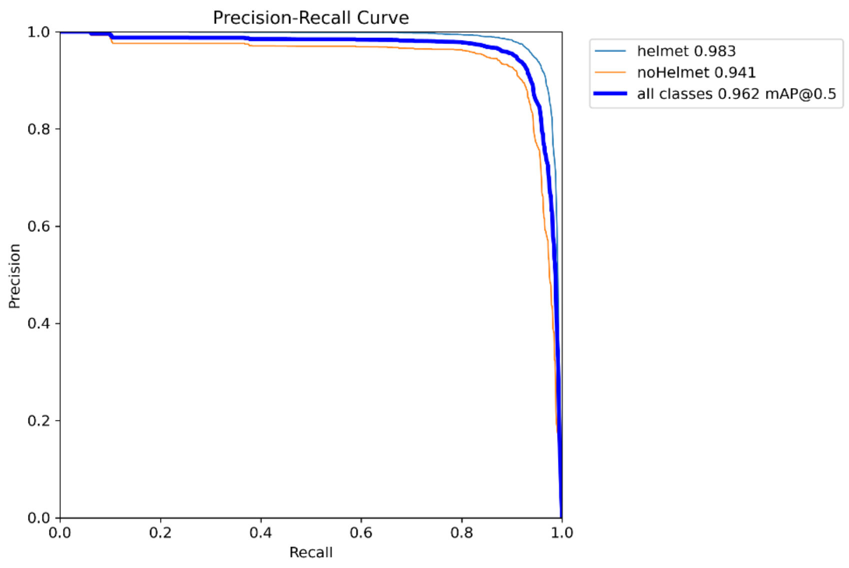
Evaluation of detection model

The evaluation of the detection model for helmets was focused on key performance metrics to ensure the system meets safety monitoring standards. Metrics such as confusion matrix, precision, recall, and the F1-score were used to assess the balance between the model's accuracy and its ability to minimise false positives and false negatives. A confusion matrix provides a detailed breakdown of true positives, false positives, true negatives, and false negatives, offering insights into areas of improvement for the model.

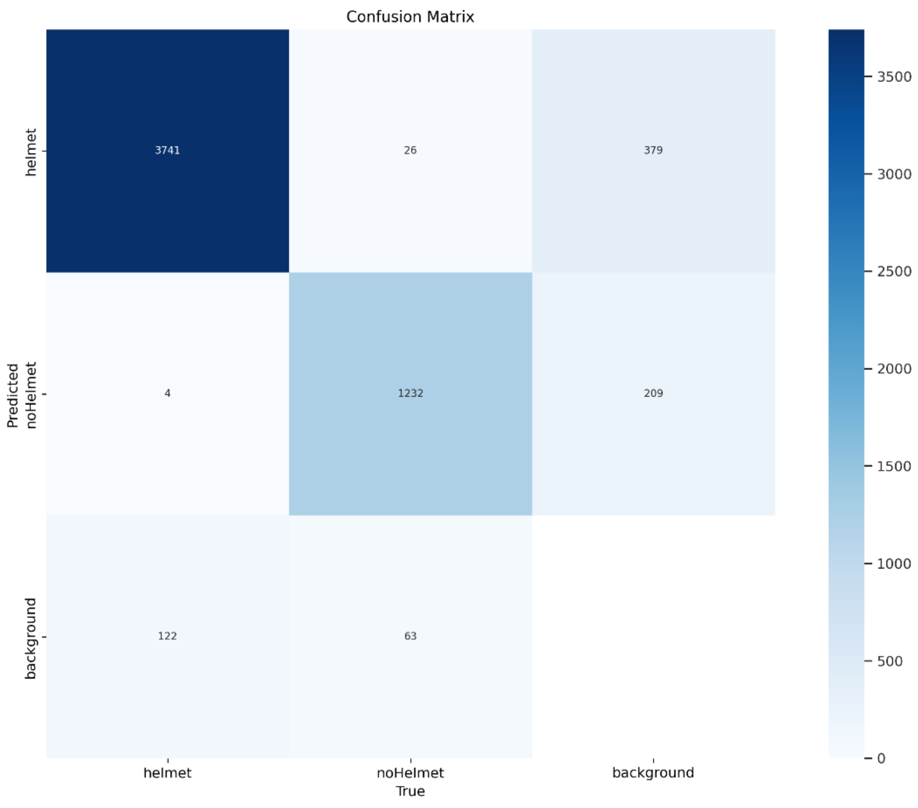
In addition to these metrics, the Precision-Recall (PR) curve was utilised to analyse the trade-off between precision and recall across different confidence thresholds. This evaluation helps optimise the model's detection threshold for specific scenarios, such as ensuring that all workers wearing PPE are identified while minimising false alarms. High recall was prioritised for detecting helmets, as missed detections pose significant safety risks.

This evaluation is conducted on a diverse test dataset to account for variations in lighting, camera angles, and occlusions, ensuring the robustness of the detection system in real-world construction site environments. By iteratively refining the models based on these evaluation metrics, the system was able to achieve high reliability and efficiency in monitoring safety compliance. See results below.

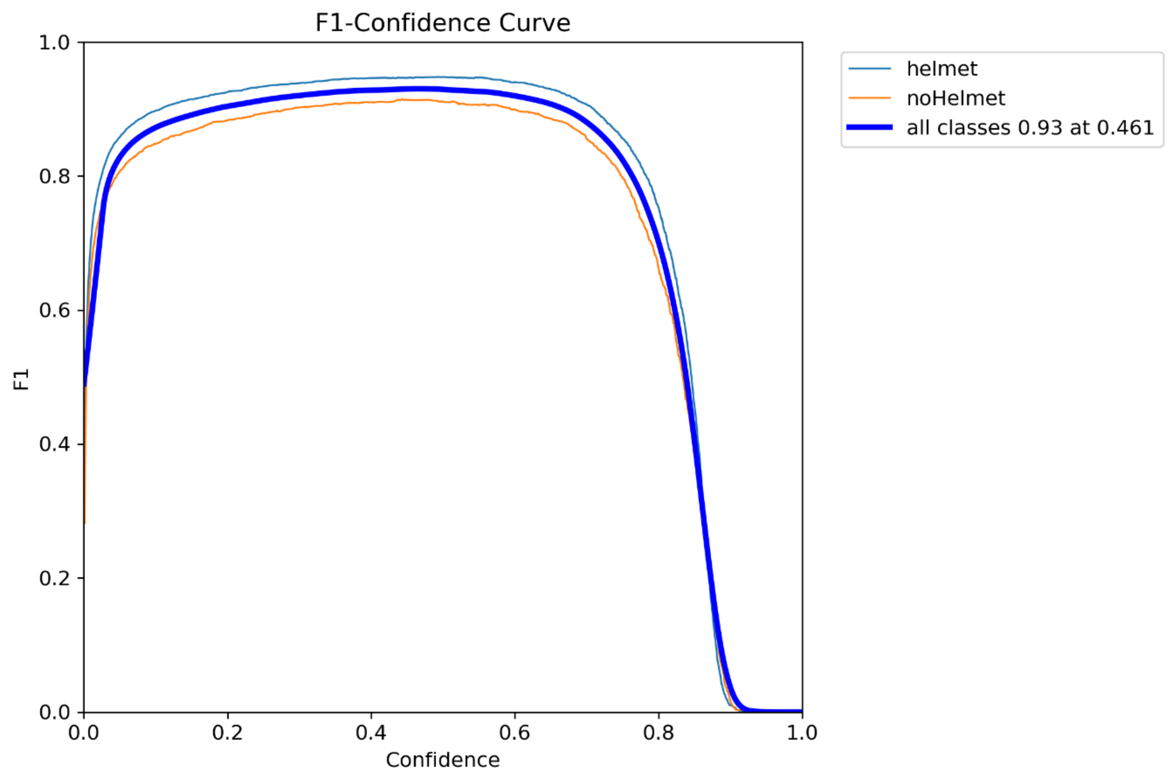
Precision-Recall (PR) curve



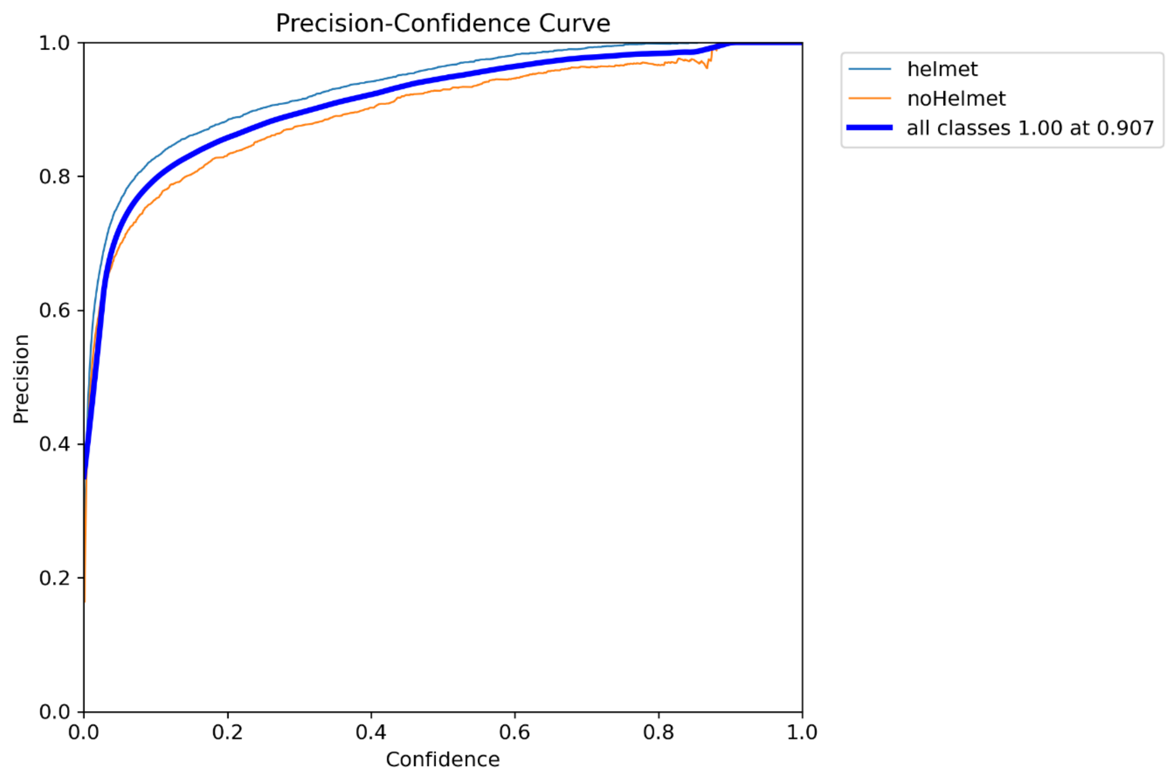
Confusion Matrix



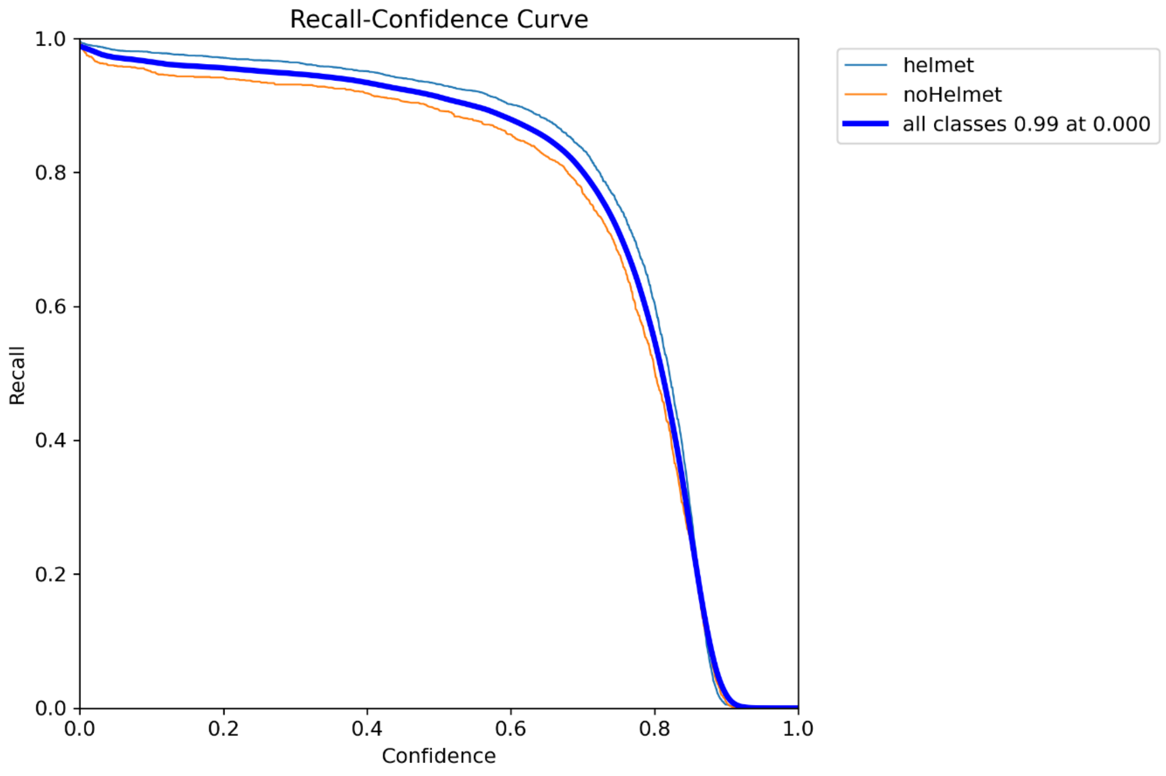
F1 – Confidence Curve



Precision – Confidence Curve



Recall – Confidence Curve



Technologies

The selection of technologies for the SiteSafeAI project was driven by the need to support real-time video monitoring, ensure high detection accuracy, and enable seamless integration between system components. Each technology was chosen after careful evaluation and was tailored to meet the specific requirements of the project. Below is an overview of how each was utilised:

- **Python** served as the core programming language due to its extensive ecosystem for AI and computer vision. It was used to implement the object detection logic, handle real-time processing of video feeds, and manage backend operations through Flask.
- **YOLOv11** (You Only Look Once) was employed as the primary object detection model for identifying safety gear such as helmets and high-visibility jackets. Its single-pass architecture allowed for fast and accurate detection, which is essential for real-time applications on construction sites.
- **Ultralytics** provided a streamlined interface for deploying and managing YOLO models. It significantly simplified the training, testing, and integration process, allowing rapid development and fine-tuning of detection workflows within the system.
- **OpenCV** was used to handle video capture from connected cameras and perform frame-by-frame annotation. It enabled the overlay of bounding boxes, labels, and confidence scores directly onto video feeds, offering visual clarity for each detection made by the system.
- **Flask** acted as the backend server, facilitating communication between the detection engine and the frontend interface. It handled API endpoints, processed incoming detection data, and served results to the user interface in real time.
- **React** was chosen for the frontend due to its ability to create dynamic and responsive user interfaces. It allowed users to interact with the system, monitor live camera feeds, configure settings, and review safety detections with ease.
- **SMTP** (Simple Mail Transfer Protocol) was integrated to send automated email alerts whenever a safety violation was detected. This ensured that site supervisors or safety officers received real-time notifications for timely intervention.
- **MySQL** served as the system's database, storing structured data such as companies, detections, cameras, users, and settings. This enabled efficient querying for generating reports and tracking compliance trends over time.

This cohesive combination of technologies allowed SiteSafeAI to deliver a robust, efficient, and scalable solution tailored for enhancing safety monitoring on construction sites.

Challenges

Developing SiteSafeAI was a highly rewarding experience, but it also came with several challenges, especially as this was my first time creating a full-stack application from scratch. From building and connecting the frontend, backend, and database, to integrating real-time services like automatic email alerts, the project pushed my skills across multiple domains. Below are the key challenges I encountered and how I addressed them.

1. **Building a Full-Stack Application from Scratch**
   This project marked my first experience in building a complete full-stack system on my own. Integrating the frontend (React) with a Flask backend, connecting everything to a MySQL database, and incorporating services like automated emails into a cohesive application required learning multiple technologies simultaneously. Coordinating all these moving parts to function seamlessly was a significant learning curve, especially in handling the flow of data and ensuring proper communication between components.

2. **Learning and Implementing Flask**
   Using Flask for the first time posed a challenge as I had to understand how to set up routing, connect with the database, and expose APIs for the frontend to consume. Unlike frameworks I had previously encountered, Flask required careful planning of request handling and state management. Debugging issues and structuring the backend efficiently was time-consuming, but ultimately it gave me a much deeper understanding of backend development.

3. **Training the Detection Model for Accuracy**
   One of the most technically demanding aspects of the project was training a reliable helmet detection model. Achieving high accuracy required me to train the model over 20 times, starting with about 200 images and the final version was trained on about 600 images, each time using different variations of image datasets and hyperparameters. Managing data preprocessing, labelling images, and balancing the dataset for optimal performance was a labour-intensive task. Despite these difficulties, the iteration process allowed me to significantly improve model precision and recall.

4. **Real-Time Video Streaming with Sockets**
   Streaming video from the frontend to the backend and returning the processed frames in near real-time was particularly complex. I implemented web sockets

to establish continuous communication between the browser and the Flask server. Each video frame needed to be encoded, transmitted, decoded, processed, then re-encoded and sent back, which introduced both performance and technical challenges. Understanding how to manage these transformations efficiently without lag was a critical hurdle.

5. **Creating a Reliable Detection Algorithm**
   Rather than triggering a detection based on a single frame, I implemented a robust queue-based algorithm to ensure that detections were only recorded when the system was confident. The logic was designed to only trigger a detection when 8 out of the last 10 frames contained a "no helmet" result. Creating and fine-tuning this algorithm required careful thought to balance accuracy with reliability, and involved using data structures like queues to track recent detection results efficiently.

Overall, overcoming these challenges was an essential part of the learning process and significantly deepened my understanding of full-stack development, real-time systems, and machine learning model integration. Each obstacle ultimately became an opportunity to grow and improve the SiteSafeAI system.

Learning Outcomes

Working on the SiteSafeAI project provided a transformative learning experience that went far beyond just writing code. It gave me the opportunity to build a full-stack application from scratch, tackle real-world challenges, and explore technologies I hadn't used before. The skills and insights I gained from this project have been some of the most valuable in my learning journey so far.

1. **Full-Stack Development Experience**
   This was my first time independently building a full-stack application, and the experience was both challenging and rewarding. I learned how to manage and coordinate a system that included a frontend, backend, database, and additional services like automated email alerts. Understanding how each layer communicates and fits into the broader system architecture helped me become more confident as a developer and gave me a clear picture of how large-scale applications are built in the real world.

2. **Technical Skill Growth**
   Through this project, I significantly improved my proficiency in Python, particularly in areas such as API development with Flask, real-time data handling, and model integration. I also gained practical experience using React for the frontend, OpenCV for video processing, and MySQL for database operations. Learning to integrate these different technologies and make them work together smoothly has been a major milestone in my development.

3. **Discovery of Personal Interests**
   One of the most unexpected but valuable outcomes was discovering my passion for object detection and computer vision. As I worked on training and fine-tuning the detection model, I found myself genuinely enjoying the process of improving accuracy and interpreting model performance. This sparked a strong interest in continuing to explore AI, particularly in the area of computer vision, and has influenced my future learning goals and career direction.

4. **Problem-Solving and Independence**
   This project also taught me how to approach complex problems independently. From debugging real-time socket communication to implementing a custom detection algorithm, I learned to break down issues, research solutions, and test implementations systematically. The ability to stay persistent and adapt to new challenges has made me more confident and resourceful as a developer.

In summary, SiteSafeAI helped me grow both technically and personally. It strengthened my full-stack development skills, deepened my interest in AI and object detection, and showed me how capable I can be when taking on large projects. This experience has laid a strong foundation for my future work in software development and machine learning.

Future Development

While the current version of SiteSafeAI delivers a strong foundation for real-time safety monitoring, there are several areas where the application could be further enhanced and developed to improve functionality, scalability, and user experience. These improvements would help the system better meet the dynamic needs of construction site monitoring and safety compliance.

1. **Support for Unlimited Camera Connections**
   Currently, the application supports simultaneous connection of up to two cameras. A key goal for future development is to allow users to connect an unlimited number of cameras. This would make the system suitable for larger construction sites where more extensive monitoring is required.
2. **Adaptive User Interface for Multi-Camera Display**
   The user interface in its current form is optimised for one or two camera feeds. To support additional camera connections, the UI will need to be redesigned to dynamically adapt based on the number of active cameras. This includes creating a layout that allows easy navigation and monitoring of multiple feeds while maintaining a clean and functional design.
3. **Expanded Detection Capabilities**
   While helmet detection is currently the focus, the model could be extended to include detection of high-visibility jackets and safety boots. These additional detection options could be made toggleable in the UI, allowing users to select which types of safety violations they want the system to monitor at any given time. This would offer greater flexibility and customisation for different site requirements.
4. **Improved Email Notification System with Image Attachments**
   The automatic email service currently sends notifications based on violations but does not include image evidence. In future versions, the system could attach the detected image with annotated bounding boxes directly in the email. This would streamline the review process by allowing safety officers to quickly verify incidents without logging into the dashboard, saving time and increasing responsiveness.

These future enhancements would significantly increase the scalability, usability, and effectiveness of SiteSafeAI, moving it closer to becoming a comprehensive safety monitoring solution for construction sites of all sizes.

Project Requirement Fulfilment

The SiteSafeAI project successfully implemented a wide range of planned features, fulfilling most of the core requirements and objectives. While there were a few limitations and areas left for future development, the overall outcome of the project met expectations and provided a solid foundation for further improvements.

Achieved Requirements

- Login and registration system
- User roles with admin and normal user access control
- Clean and functional frontend using React
- Full user management functionality
- Camera management interface and logic
- Detection management and record-keeping
- Detection image review system for verification
- No helmet detection using a trained object detection model
- Custom detection algorithm using a queue-based logic
- Logging of all detections with timestamps and details
- Support for connecting multiple cameras
- Automatic email service notifying safety officers upon detection

Not Achieved / Limitations

- Multiple camera support is currently limited to two cameras
- Automatic email service does not yet include image attachments
- UI layout and design needs further adaptation to comfortably support multiple camera feeds

This breakdown highlights the strong progress made during development, while also identifying specific areas for future refinement and scalability.

What I Would Do Differently If Starting Again

Reflecting on the development process of SiteSafeAI, one of the key things I would approach differently is how I initially prioritised the object detection tasks. At the beginning of the project, I was very focused on trying to get both helmet and high-visibility jacket detection working at the same time. While the intention was to make the system as complete as possible from the start, this split focus ended up slowing down my overall progress.

Managing and training detection models for multiple classes introduced extra complexity, especially considering this was my first time working on a full-stack application with object detection. It wasn't until later in the development that I decided to shift my full attention toward helmet detection, which was the core functionality and safety requirement of the project. Once I made this change, progress significantly accelerated and the system became more reliable and consistent.

Looking back, I realise that I would have made much better use of my time and resources by focusing solely on helmet detection at the start. This would have allowed me to develop a strong, stable core system first, and then gradually build upon it with additional features like high-visibility detection as future enhancements. This experience has taught me the importance of clear prioritisation and tackling one major component at a time in complex software projects.

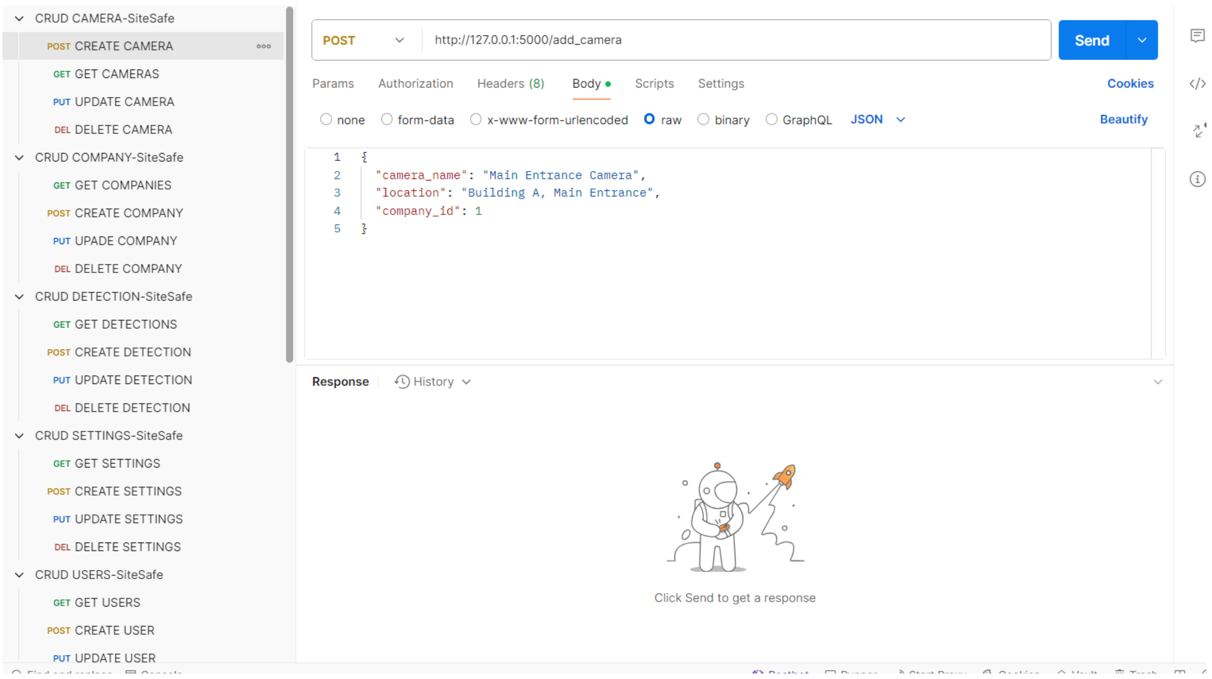
Testing

Rigorous testing was an integral part of the development process for SiteSafeAI to ensure that all system components like the backend, frontend, and services functioned reliably and consistently.

API Testing with Postman

During development, Postman was used extensively to test each API endpoint. This allowed for the verification of request/response structures and ensured that data was being sent and received correctly. After each API call, the corresponding entries in the database were checked to confirm that the expected changes, such as user detail updates or detection logs had been applied correctly.

Below shows the extensive and rigorous testing that was completed through Postman. Each connection was tested and checked in the database to make sure the correct changes had taken place. These are just a portion of the calls tested seen on the left had side of the image.



CRUD CAMERA-SiteSafe
- POST CREATE CAMERA
- GET GET CAMERAS
- PUT UPDATE CAMERA
- DEL DELETE CAMERA

CRUD COMPANY-SiteSafe
- GET GET COMPANIES
- POST CREATE COMPANY
- PUT UPADE COMPANY
- DEL DELETE COMPANY

CRUD DETECTION-SiteSafe
- GET GET DETECTIONS
- POST CREATE DETECTION
- PUT UPDATE DETECTION
- DEL DELETE DETECTION

CRUD SETTINGS-SiteSafe
- GET GET SETTINGS
- POST CREATE SETTINGS
- PUT UPDATE SETTINGS
- DEL DELETE SETTINGS

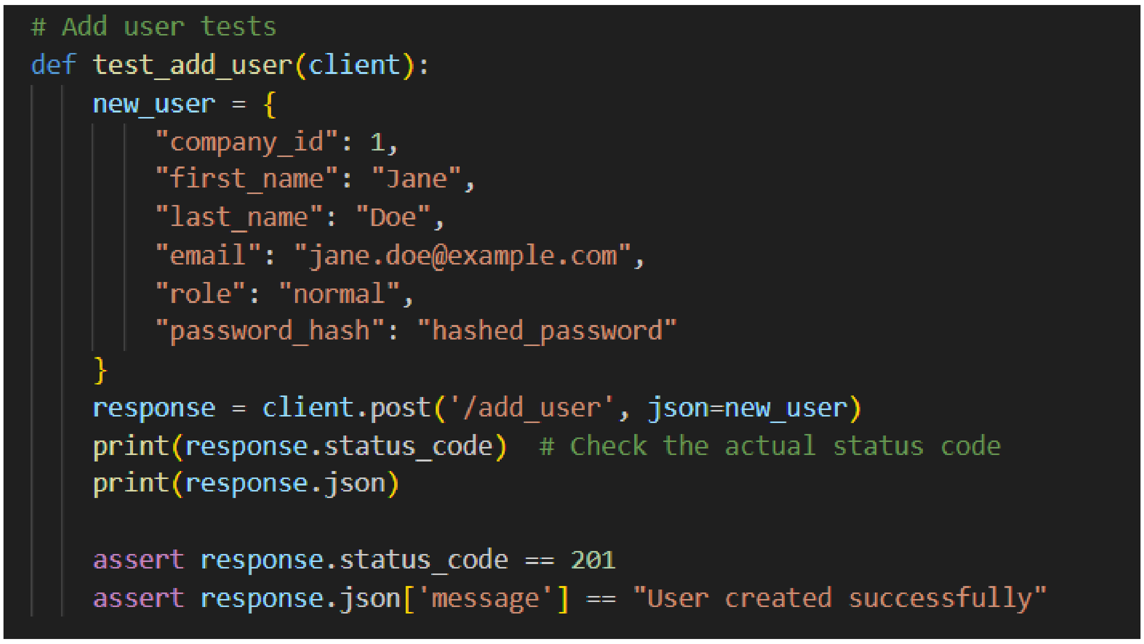
CRUD USERS-SiteSafe
- GET GET USERS
- POST CREATE USER
- PUT UPDATE USER

POST http://127.0.0.1:5000/add_camera

```
{
  "camera_name": "Main Entrance Camera",
  "location": "Building A, Main Entrance",
  "company_id": 1
}
```

Backend Unit and Regression Testing

Many test cases were written directly into the Flask backend using automated test suites. These tests targeted the API endpoints and ensured their functionality remained stable throughout the development lifecycle. As new APIs were added, these automated tests enabled effective regression testing, providing confidence that previously developed features continued to work as intended even as new functionality was introduced. Below we can see an example of a test to add a user.



```python
# Add user tests
def test_add_user(client):
    new_user = {
        "company_id": 1,
        "first_name": "Jane",
        "last_name": "Doe",
        "email": "jane.doe@example.com",
        "role": "normal",
        "password_hash": "hashed_password"
    }
    response = client.post('/add_user', json=new_user)
    print(response.status_code)  # Check the actual status code
    print(response.json)

    assert response.status_code == 201
    assert response.json['message'] == "User created successfully"
```

Frontend-Backend Integration Testing

Once the API endpoints were tested in isolation, they were integrated into the React frontend. Further testing was conducted through the UI to ensure that the API calls made from the frontend correctly modified the database. For example, when editing a user’s details through the interface, the tests confirmed that the changes were reflected and saved properly in the backend database.

Detection Algorithm Validation

Special focus was given to testing the detection logic, as it forms the core of the system. Through experimentation and iteration, the detection algorithm was tuned to log a detection only when 8 out of the last 10 frames showed a "no helmet" violation. This queue-based approach was tested under various scenarios to ensure it reliably avoided false positives and recorded only legitimate infractions.

Emailing Service Testing

The email notification system was also thoroughly tested to ensure that all detection details sent matched the data logged in the database. During testing, an issue was identified where the college Wi-Fi network blocked the email service, preventing successful delivery. To work around this, a mobile hotspot was used during demonstrations and for final testing to guarantee reliable functionality.

Overall, these multiple layers of testing like unit, regression, integration, and service-level ensured that SiteSafeAI was a dependable and production-ready application.

Conclusion

The development of SiteSafeAI has been a deeply rewarding and educational journey, pushing me to expand my knowledge and skills across multiple areas of software development. As my first full-stack project built entirely from scratch, it provided an invaluable opportunity to integrate a range of technologies, spanning from a Flask backend and a React frontend, to MySQL databases and additional services like automated email notifications.

Throughout the project, I encountered numerous challenges, from learning how to structure a scalable application using Flask to implementing real-time video processing and training a custom object detection model. Each obstacle became a learning moment, helping me develop more robust problem-solving skills and a deeper understanding of how modern software systems work in practice. I also discovered a strong interest in computer vision and object detection, which I now plan to explore further in my future studies and career.

Although some features remain to be improved or expanded, such as support for additional cameras and better UI scalability, the core goals of the project have been successfully achieved. Most importantly, this project has made me a more confident and capable software developer, and I now feel much better prepared to take on large, complex software development challenges in a professional setting.

References

Figures

Fig 1. SETU Logo: SETU Logo

Fig 2. SiteSafe Logo Creation Tool: https://www.canva.com/design/play?type=TAB7AVEOUWQ&category=tACZCvjI6mE&locale=en

General Information

1. Python: https://www.python.org
2. YOLO: https://yolo-docs.readthedocs.io/en/latest/
3. Ultralytics: https://ultralytics.com
4. OpenCV: https://opencv.org
5. Flask: https://flask.palletsprojects.com
6. React: https://reactjs.org
7. SMTP: https://tools.ietf.org/html/rfc5321
8. MySQL: https://www.mysql.com