Kalpa Publications in Computing
Volume 22, 2025, Pages 117–128
Proceedings of The Sixth International Conference on Civil and Building Engineering Informatics

Creation of a Web-Based Tool for the Visual Inspection of Building Equipment

Angelina Aziz[1] and Markus König[1]
[1] Ruhr University Bochum, Bochum, Germany
angelina.aziz@ruhr-uni-bochum.de, koenig@inf.bi.rub.de

Abstract

Fire safety inspection is essential for protecting occupants from fire hazards. Traditional inspection methods often rely on subjective assessments, often leading to potential errors and inadequate maintenance. This study introduces a visual fire safety inspection tool that integrates self-trained Machine Learning (ML) services to enhance the accuracy and efficiency of documenting Fire Safety Equipment (FSE) using images. The ML services were incorporated into a web-based application built with the React framework, featuring a backend developed using FastAPI and MongoDB for efficient processing and scalability. The tool achieved a high mean average precision (mAP) over 80% on testing datasets. It offers a robust environment for fire safety experts to validate and compare models, also providing insights into the impact of active learning algorithms. Despite the tool's high accuracy, challenges such as slow loading times and application freezing were identified, with proposed solutions focusing on optimizing backend and frontend processes. The integration of ML services demonstrates significant potential for improving fire safety inspections, with future work aimed at refining models, expanding real-time monitoring capabilities, and ensuring compatibility with Building Information Modeling (BIM) systems and conventional smartphones.

1 Introduction

Traditionally, fire safety equipment (FSE), such as fire extinguishers, smoke detectors, and safety signs, are inspected manually. These inspections often rely on subjective judgment and manual data entry, increasing the likelihood of human error and incomplete documentation. As buildings grow in complexity, the need for more accurate, efficient, and automated facility management and maintenance becomes essential to ensure safety compliance and reduce risks (Sanzana et al., 2022; Yousefli et al., 2020).

The facility management sector has increasingly adopted Computer-Aided Facility Management (CAFM) software to manage support activities, often complemented by Computerized Maintenance

Management System (CMMS) software for more detailed monitoring and maintenance scheduling (Rudl, 2021). However, over the years, researchers have explored various specific strategies to improve facility management processes, particularly during the Operation and Maintenance (O&M) phase of buildings. Previous literature concentrated on developing mobile and web-based applications to facilitate this process. For instance, (Alhaj et al., 2022) proposed an occupant-centric facility maintenance management system, demonstrating the value of mobile applications for streamlining building maintenance tasks. Similarly, (Wen et al., 2017) introduced a web-based hybrid BIM cost-estimating system for fire safety engineering, which includes a "Fire Safety Equipment Property Database" module, underscoring the importance of digital tools for fire safety inspections. Yousefli et al. (2020) developed an automated Multi-Agent Facility Management system (MAFMS) that leverages a web-based application and multi-agent technology to integrate components such as workflow prioritization and resource scheduling. This system effectively automates facility management processes, demonstrating significant improvements in time-sensitive environments like hospitals. Using BIM and robotics, Chen et al. (2023) of this study focus on a knowledge-driven approach through a formalized model called the integrated Scene-Task-Agent (iSTA). This model synergizes information from BIM and robotics to enable autonomous facility inspections. However, while these systems offer significant improvements in overall facility management, they often fall short when it comes to automating specific fire safety inspection subjects, which still require manual data collection and input.

The integration of machine learning (ML) into the fire safety inspection processes offers an opportunity to enhance both the accuracy and efficiency of these inspections. By automating tasks such as detecting fire safety equipment in images, identifying safety signs, and recognizing inspection tags, ML can streamline the inspection process. In particular, computer vision techniques have shown significant promise in the detection and documentation of FSE through image analysis (Aziz et al., 2022; Corneli et al., 2020; Heinbach & Aziz, 2023). However, there is a notable gap in the availability of image-based web tools that can quickly analyze FSE and evaluate captured images within the application, enabling rapid responses and direct data storage.

To address these limitations, this paper presents the concept of a web-based visual fire safety inspection tool that integrates ML services and demonstrates it through a proof-of-concept implementation. By automating the detection and documentation of FSE using images, this tool aims to decrease documentation efforts and streamline the inspection process. The tool leverages computer vision techniques to detect key FSE components directly from images. The web application is built using modern web technologies, with a frontend developed in React (Clark et al., 2022), and a backend powered by FastAPI (Ramírez, 2023) and MongoDB (Stephens, 2023) for data management. The integration of ML services into this application allows fire safety managers to automate routine evaluation tasks, compare model performance, and validate results.

2 Method

The development of the visual fire safety inspection tool was focused on creating a scalable and user-friendly web-based application that automates FSE documentation through the integration of ML services. As illustrated in Figure 1, the workflow starts with users, typically fire safety inspectors, uploading images of FSE such as fire extinguishers, safety signs, and inspection tags through the tool's web interface. Once images are uploaded, the user requests the ML service to automatically detect and document key fire safety features within the images. This is handled using computer vision techniques that analyze the images.

Additionally, the tool incorporates an active learning component that plays a critical role in improving the performance of the ML models. As shown in Figure 1, the system can request annotation from the user on the most informative image data. These are the data that the model finds uncertain or challenging. Once annotated, the data is sent back into the system, where it is stored in the database for further use like retraining the ML models improving their accuracy and performance in detecting FSE.

The tool should manage data storage and facilitate seamless communication between the user and ML services. Inspection results generated by the ML system, including detected FSE and any accompanying documentation, are directly downloadable and archived in the database. This process of archiving results allows users to retrieve and review inspection reports at any time, so that the tool supports long-term data management.

In this workflow, the combination of automated detection, user feedback through active learning, and periodic model retraining creates a tool that not only reduces manual documentation efforts but also adapts and improves overtime, meeting the evolving needs of fire safety inspections.

The next subchapters outline the key components of the tool's creation, from setup and interface design to the integration of ML services and backend architecture.

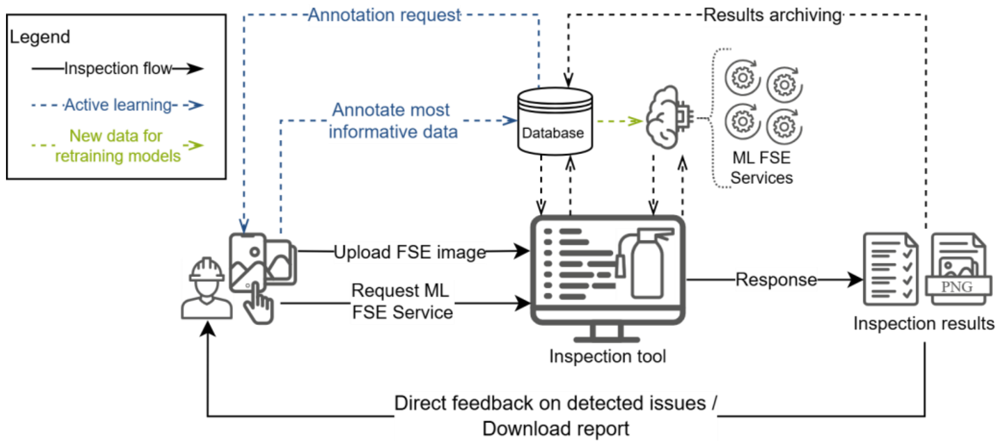


Figure 1: Workflow of the web-based visual fire safety inspection tool integrating ML services and active learning.

2.1 Web Application Setup

The web application was developed using a modular, component-based architecture to ensure flexibility and scalability for both the frontend and backend. As illustrated in Figure 2, the frontend was built using React (Clark et al., 2022), a widely used JavaScript library known for efficiently rendering dynamic user interfaces. React's Virtual DOM optimizes updates in response to user actions, making it an ideal framework for handling large datasets of images and annotations. This is particularly important in the context of fire safety inspections, where the application needs to manage a significant volume of images without sacrificing performance.

The backend was developed using FastAPI (Ramírez, 2023), a modern Python framework designed for high-performance asynchronous operations. FastAPI was chosen for its ease of use and its ability to handle concurrent requests, ensuring that multiple users can interact with the application seamlessly. As shown in Figure 2, FastAPI communicates with the MongoDB (Stephens, 2023) database to store, retrieve, and manage project data, user information, and image annotations. MongoDB was selected for its flexibility in handling dynamic, document-based data, which is generated during on-site inspections and can scale with increasing data volumes.

Additionally, FastAPI integrates with the CVAT (Computer Vision Annotation Tool) (Sekachev et al., 2020), which is used for the annotation of new FSE images. This allows images and their associated annotations to be sent to CVAT for processing, as depicted in Figure 2, and then returned to FastAPI for further handling and storage. The interaction between FastAPI and MongoDB ensures that annotated images are properly archived and available for later retrieval, while CVAT facilitates the efficient annotation of new data.

To ensure consistent deployment and operation across various environments, the entire application was containerized using Docker. Docker simplifies the deployment process by bundling all components, including the FastAPI backend, MongoDB database, and CVAT integration, into containers that can be deployed on any machine without complex setup procedures. This containerized architecture guarantees that the system performs reliably and efficiently, regardless of the deployment environment.

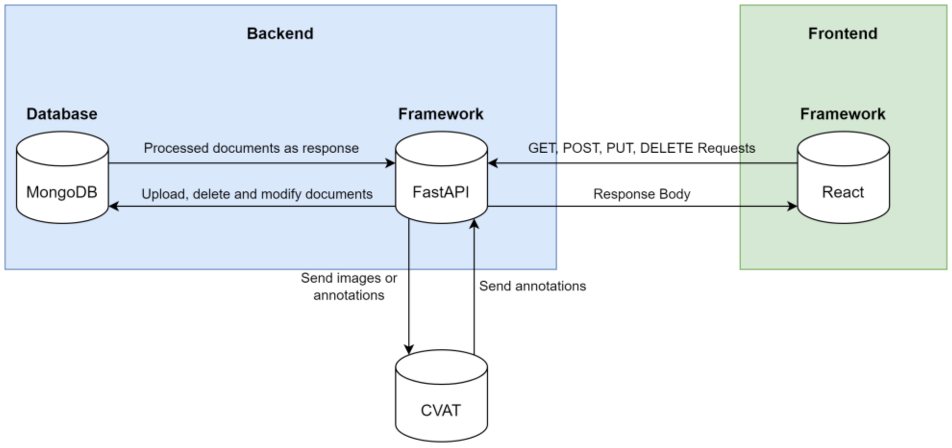

Backend
Frontend
Database
Framework
Framework
Processed documents as response
GET, POST, PUT, DELETE Requests
MongoDB
Upload, delete and modify documents
FastAPI
Response Body
React
Send images or annotations
Send annotations
CVAT

Figure 2: Workflow of the web-based visual fire safety inspection tool integrating ML services and annotation.

2.2 ML Services Integration

The core feature of the web application is its ability to automate the detection and documentation of FSE using ML models. Several custom-trained ML services were integrated into the application to facilitate this process. These services are built on object detection models such as YOLOv5 (Jocher et al., 2022) and YOLOv7 (Wang et al., 202366), selected for their high precision in object detection and their adaptability to the specific needs of fire safety inspections. YOLOv5 and YOLOv7 were chosen based on their proven performance in prior studies on fire safety equipment detection (Aziz et al., 2023; Aziz & König, 2024; Bayer & Aziz, 2022), offering a reliable balance of accuracy and speed (mAP > 80%). Newer models were not adopted to maintain consistency with these validated results and avoid additional retraining efforts.

As illustrated in Table 1, the ML services integrated into the tool cover a range of critical detection tasks, enabling the system to identify FSE, safety signs, inspection tags, and maintenance information. These services were developed based on prior ML-based studies (Aziz et al., 2023; Aziz & König, 2024; Bayer & Aziz, 2022) and are trained on custom datasets. One of the core services is *FSE Detection*, which identifies FSE such as fire extinguishers, fire blankets, smoke detectors and manual call points within images. This service is powered by a custom-trained YOLOv5 model, specifically optimized for real-time object detection. It achieves a mean Average Precision (mAP) of 80.1% at Intersection over Union (IoU) thresholds between 0.5 and 0.95, ensuring reliable identification of critical equipment in various environments. Another essential service is *Safety Sign Detection*, responsible for recognizing safety signage typically found near fire safety equipment.

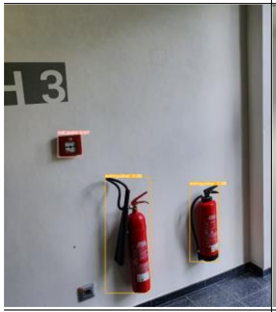
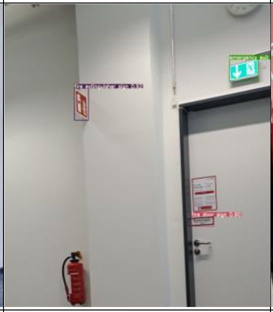
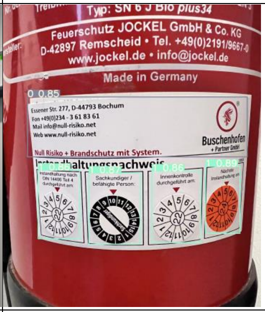
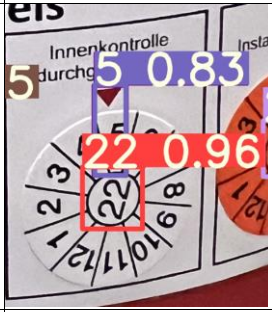
FSE Detection	Safety Sign Detection	Inspection Tag Detection	Maintenance Information Detection
			
Used model: YOLOv5	Used model: YOLOv7	Used model: YOLOv7	Used model: YOLOv7
mAP@0.5:0.95 = 80.1%	mAP@0.5 = 85.2%	mAP@0.5:0.95 = 85.5%	mAP@0.5:0.95 = 86.1%

Table 1: Performance metrics of ML FSE services integrated into the visual inspection tool, detailing the models used and their mAP at various IoU thresholds on test datasets.

This service uses a YOLOv7 model to detect important safety signs such as exit signs and fire extinguisher location markers. The model performs with an mAP of 85.2% at a 0.5 IoU threshold. *Inspection Tag Detection* is a service focused on detecting inspection tags and manufacturer tag affixed to fire extinguishers and other equipment using the YOLOv7. With an mAP of 85.5% at IoU thresholds between 0.5 and 0.95, this service is highly accurate.

Finally, *Maintenance Information Detection* is another key service integrated into the tool. It is designed to recognize and extract date information from inspection tags, particularly focusing on fields such as the month and year of the last or next inspection. Like the inspection tag service, this functionality is driven by a YOLOv7 model and achieves a high level of accuracy, with an mAP of 86.1% at IoU thresholds of 0.5 to 0.95. These ML services, integrated into the backend architecture, allow users to upload images of FSE and receive automated feedback on the equipment's condition and inspection status.

2.3 User Interface

The user interface of the web application was designed with a focus on usability, scalability, and performance, ensuring that users can effectively manage their images, annotations, and models within various projects. As outlined in Figure 3, the component hierarchy of the application reflects a modular architecture that allows for the smooth management of fire safety inspection tasks, while also accommodating active learning algorithms to minimize the need for manually labeled images.

The *App Component* serves as the root of the application, acting as the central entry point where essential states—such as user credentials, access tokens, and project data—are managed and passed down to other components. This architecture ensures that the system remains both scalable and responsive, making it capable of handling a large number of projects and associated datasets. After logging in via the *Login Component*, where users can either sign in or register for a new account, users are redirected to the *ProjectList Component*.

The *ProjectList Component* offers an overview of all current projects associated with the user's account, including functionalities to create, delete, or search for specific projects. Each project is represented by a *Project Component*, which displays important information, such as the project's image, title, description, and additional metadata. If a new project needs to be created, the *Popup*

Component is rendered, allowing the user to enter the necessary details, such as the project's name and description.

Once a project is selected, the user is redirected to the *ProjectSite Component*, which contains the primary functionalities of the application. Within the *ProjectSite* Component, each project is represented visually by a *ProjectSiteCard*, which serves as an interface summarizing the key details of the project. This includes the project's name, associated images, annotation progress, and model versions. The *ProjectSiteCard* provides users with quick access to essential information and actions, such as uploading new images, viewing statistics, or managing annotations

The active learning mechanism implemented in this study builds upon the approach described by Soultana and Aziz (2023) and is designed to enhance the performance of pretrained object detection models while reducing the annotation burden. The human-in-the-loop framework plays a central role in this process, enabling users to iteratively refine model performance by providing annotations for the most uncertain or challenging samples. Specifically, the algorithm employs class weighting to address class imbalance by assigning higher importance to underrepresented classes, ensuring their inclusion in the training set. Diversity sampling clusters images containing rare classes, improving the likelihood of detecting and training on these less frequent instances. Additionally, image quality assessment via Laplacian variance reduces the influence of low-quality images by prioritizing higher-quality samples for annotation. This active learning cycle allows for continuous improvement of pretrained models by leveraging user-provided feedback to retrain models with a focus on the most informative and representative data.

The *Statistics Component* enables users to evaluate and validate their models, providing visual comparisons between different versions of their object detection models. Users can easily compare fine-tuned models against the original ones using performance metrics such as mAP**,** a critical measure of the effectiveness of the active learning algorithms used during the annotation and training process. This functionality allows users to continuously refine their models.

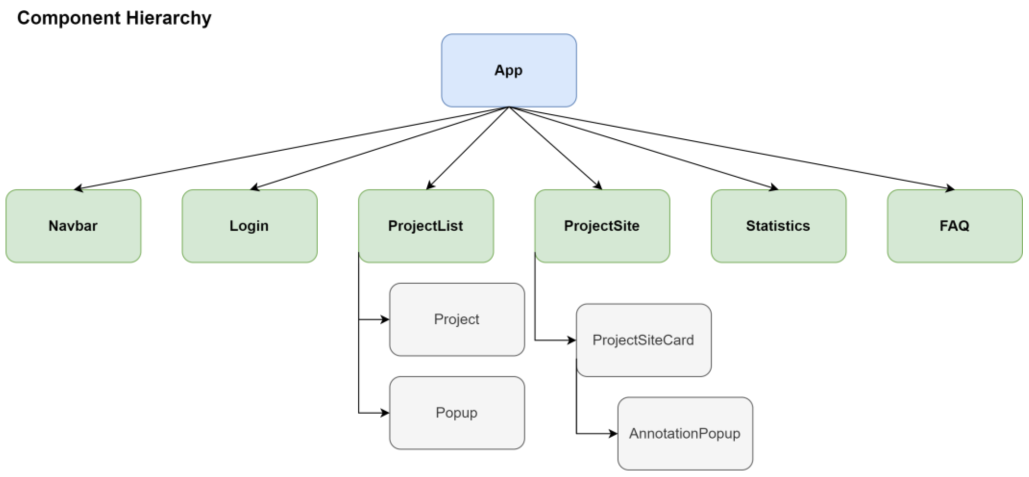


Figure 3: Performance metrics of ML FSE services integrated into the visual inspection tool, detailing the models used and their mAP at various IoU thresholds on test datasets.

2.4 Backend Architecture

FastAPI (Ramírez, 2023), a modern Python framework, was chosen for the backend due to its ability to support asynchronous programming and its compatibility with key Python libraries, such as Keras and TensorFlow, which are essential e.g., for implementing the active learning algorithms used in this study. FastAPI's simplicity and performance make it an ideal choice for handling large numbers of user requests while maintaining high throughput.

The communication between the frontend and backend is handled through a REST API, which facilitates various operations via GET, POST, PUT, and DELETE requests. To improve performance and handle concurrent requests efficiently, the backend leverages FastAPI's support for asynchronous programming. By using the async and await keywords, the backend can manage multiple user requests concurrently without blocking the main application. Figure 4 depicts an example showcasing how upcoming requests are processed. For long-running synchronous tasks, such as training a YOLO model or ranking images using active learning, background tasks are assigned to dedicated worker threads using Python's ThreadPoolExecutor. This ensures that resource-intensive tasks run in the background, allowing other user requests to be processed simultaneously. Users are kept informed of the status of these tasks through visual feedback, such as loading indicators.

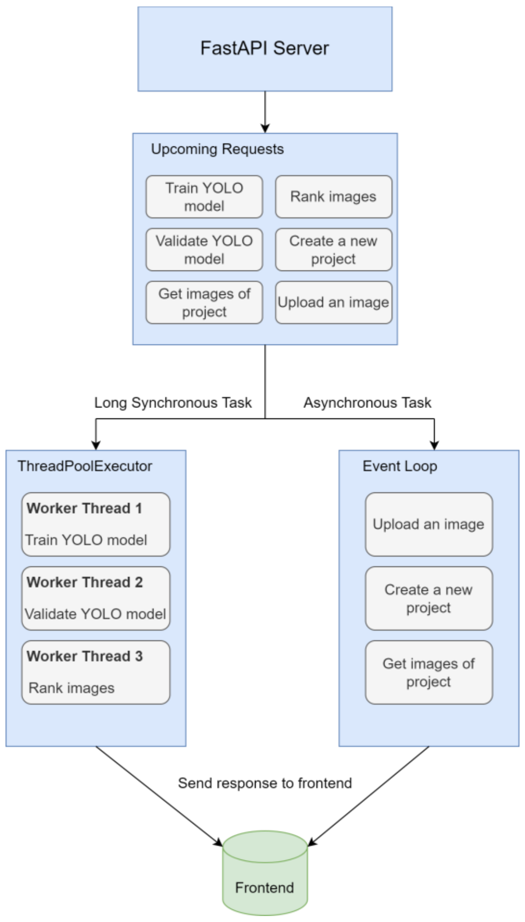


Figure 4: Visualization on how certain requests are handled.

To handle the storage and retrieval of data, the backend uses MongoDB, a document-oriented NoSQL database. MongoDB was selected for its flexibility and scalability, which make it well-suited for managing the unstructured data generated during fire safety inspections, such as images, annotations, and model results. The use of Binary JSON (BSON) allows MongoDB to efficiently store complex data types, enabling quick access and manipulation of data. The structure of the MongoDB database is visualized in the Entity-Relationship Diagram (ER) shown in Figure 5, which illustrates the relationships between various tables and their attributes.

The *Users table* contains essential user information, such as username, email, and password, which are required for authentication. This table is linked to other core components—*Projects, Images, Models*, and *Annotations*—through foreign keys. Each project, represented in the *Projects table*, has attributes such as the project name, description, and storage paths for images and annotations. The project is also linked to a user and can contain multiple models and annotations, as indicated by the foreign keys in the corresponding tables. The *Images table* stores metadata for each uploaded image, including its name, file type, and storage path. Additionally, it includes a *ranking* attribute, which is generated during the active learning process to prioritize images based on their importance for training. The *Annotations table* links each annotation to the image it was derived from through a one-to-one relationship, allowing each image to have at most one corresponding annotation. The *Models table* contains information about the trained ML models, such as the file type, storage path, and a Boolean *selected* attribute, which indicates whether a model has been chosen for validation or further training.

By structuring the database in this way, the backend ensures that all data components—users, projects, images, annotations, and models—are well-organized and easily accessible. This schema design enables efficient querying and manipulation of data, ensuring that the application can scale effectively as the number of users, projects, and data points grows.

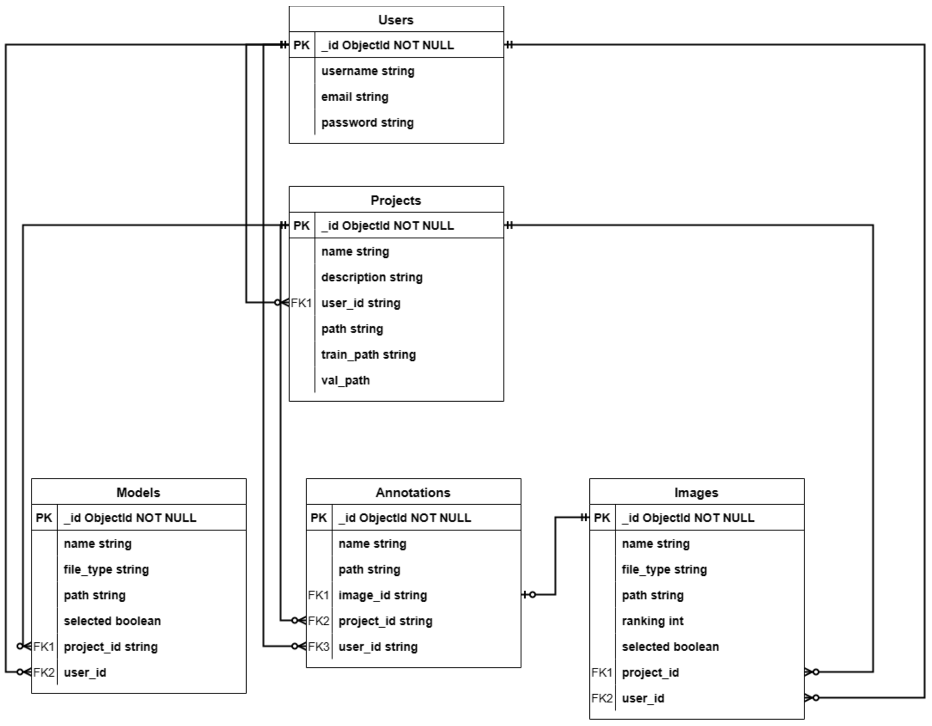


Figure 5: ER Diagram of the database.

3 Results

The web-based tool was thoroughly tested to evaluate the performance of the integrated ML FSE services. As shown in Table 2, each ML service demonstrated fast inference times once the models were loaded into the web application. However, the *FSE Detection* service, which relies on a YOLOv5 model, exhibited a slower execution time compared to other FSE services utilizing the more recent YOLOv7 models. The results confirm that the more advanced YOLOv7 models used for tasks like *Safety Sign Detection*, *Inspection Tag Detection*, and *Maintenance Information Detection* offer faster and more efficient performance. The user interface for analyzing a test FSE image and selecting an ML FSE service is shown in Figure 6. After successful detection, the original image is displayed with the detection results, and the output can also be downloaded directly.

ML FSE Service	Execution Time (s)
FSE Detection	2.5
Safety Sign Detection	1.36
Inspection Tag Detection	1.52
Maintenance Information Detection	1.32

Table 2: Execution time of the respective ML services.

In addition to performance testing, the active learning approach integrated into the tool was effective in reducing the annotation burden for users. The system allowed users to select the most informative images for annotation, improving the overall efficiency of the annotation process. By leveraging active learning, users were able to prioritize high-value images, which in turn enhanced the model's ability to learn from fewer labeled samples.

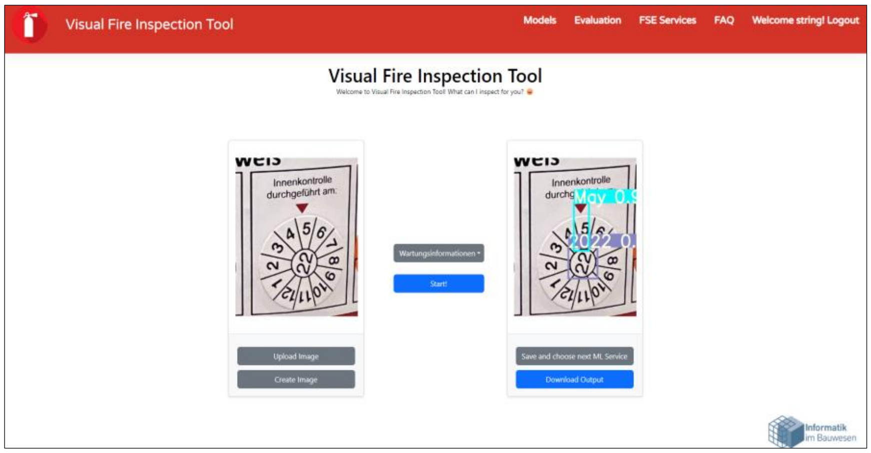

Visual Fire Inspection Tool
Models
Evaluation
FSE Services
FAQ
Welcome string! Logout
Visual Fire Inspection Tool
Welcome to Visual Fire Inspection Tool! What can I inspect for you?
Wartungsinformationen
Start!
Upload Image
Create Image
Save and choose next ML Service
Download Output
Informatik im Bauwesen

Figure 6: Execution of ML service - *Maintenance Information Detection* in the tool.

Testing also revealed that the integration between the frontend and backend functioned smoothly. The communication with CVAT for annotation purposes was seamless, enabling users to upload images, annotate them, and train models in a streamlined manner. This smooth interaction was critical in ensuring that the tool maintained a responsive and user-friendly experience.

The *Statistics Component* of the tool proved valuable in helping users visually compare model performance. It provided clear insights into the effect of active learning on model accuracy, allowing users to monitor and validate model improvements over time. As shown in Figure 7, the mAP comparison graph displayed the results of different trained models over various epochs, enabling users to see the progression of each model's performance.

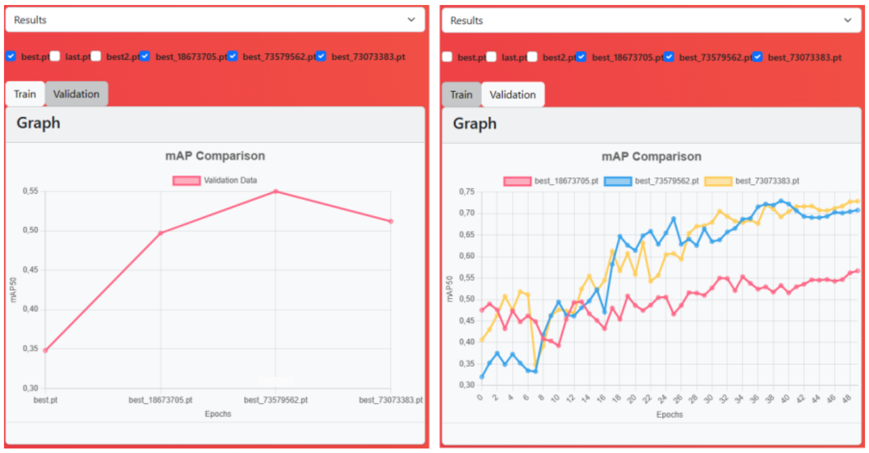


Figure 7: Visual mAP comparison of trained models in the tool. The left diagram shows mAP scores for different models on the training dataset, while the right diagram tracks validation mAP progression over epochs during training, with each line representing a different model.

4 Discussion

The testing of the visual fire safety inspection tool highlighted the strong performance of the integrated ML models in detecting FSE and documenting other FSE-related inspection subjects. Models such as the YOLOv5-based FSE detection model achieved an mAP of over 80%, proving their effectiveness under various conditions. However, several performance bottlenecks were identified during system testing that need to be addressed to optimize the user experience and overall system efficiency.

One of the main issues observed was slow loading times, particularly when handling large datasets or performing inference on multiple images. This was more apparent when switching between models trained on different architectures, such as YOLOv5 and YOLOv7. These delays likely stemmed from how the system manages requests and processes large amounts of data. Additionally, application freezing occurred during extensive image uploads or batch processing, which was traced back to synchronous backend functions causing blocking during intensive operations. Optimizing these backend processes by converting synchronous operations to asynchronous ones could significantly reduce these performance bottlenecks. Transitioning synchronous backend operations to asynchronous processes using tools like Python's asyncio could enhance the system's responsiveness. Additionally, integrating distributed computing for batch processing and leveraging cloud-based GPU resources could improve performance when dealing with large datasets or real-time applications.

Despite these challenges, the tool offered several benefits. The *ProjectSite Component* provided users with an intuitive way to navigate between projects, upload images, and run predictions. The integration with the MongoDB database ensured that all project data, images, and annotations were securely stored and easily retrieved, even during system restarts or failures. This was essential in maintaining data integrity throughout the inspection process.

The integration of the tool's active learning algorithm proved to be valuable in improving model accuracy while reducing the annotation burden on users. By focusing on selecting the most informative images for annotation, the active learning approach helped prioritize high-impact data for training. However, this process was computationally expensive, and slowdowns were observed when handling large batches of data. Streamlining the active learning process and improving model retraining efficiency could further enhance the tool's performance and usability.

5 Conclusion

The development of the web-based visual fire safety inspection tool, integrating ML services, represents a practical step toward automating fire safety inspections. It improves the accuracy and efficiency of detecting FSE and streamlines the documentation process, helping to reduce human error and save time. The tool's use of ML models, particularly YOLOv5 and YOLOv7, demonstrated high accuracy in identifying FSE, safety signs, inspection tags and maintenance information. Its user-friendly interface, combined with active learning, allows for efficient image annotation, model validation, and the creation of custom datasets, making it adaptable to various inspection needs.

Looking ahead, there are several key areas for future improvement. First, integrating real-time monitoring capabilities into the tool, such as connecting it with BIM, would enable building managers to access up-to-date fire safety documentation. Additionally, adding real-time notifications for FSE maintenance and inspections would further automate the fire safety management process.

Another important enhancement would be to ensure the tool’s compatibility with smartphones and mobile devices. This would enable on-site inspectors to perform fire safety inspections directly from their devices, eliminating the need for bulky equipment or manual data entry. Expanding the tool’s accessibility to mobile platforms would streamline the inspection process, making it more efficient and practical for fieldwork.

In conclusion, the integration of ML services into fire safety inspections has the potential to revolutionize the field by automating routine tasks, improving documentation accuracy, and enabling real-time monitoring. While the tool has already demonstrated its capabilities in partially automating FSE documentation, ongoing research and development will focus on optimizing performance, expanding functionality, and ensuring scalability for larger operations.

References

Alhaj, M. B., Liu, H., Abudayyeh, O., & Sulaiman, M. (2022). Development of a mobile application for occupant-centric facility maintenance management. In *2022 IEEE World AI IoT Congress (AIIoT)* (pp. 323–329). IEEE. https://doi.org/10.1109/AIIoT54504.2022.9817248

Aziz, A., Herbers, P., Bayer, H., & König, M. (2023). Fully Autonomous Fire Safety Equipment Inspection Missions on a Legged Robot. *Computing in Civil Engineering 2023*, 804–812.

Aziz, A., & König, M. (2024). Bilderkennungsmethoden für eine teilautomatisierte Inspektion von Brandschutzanlagen. *Bautechnik*, *101*(3), 159–165. https://doi.org/10.1002/bate.202300099

Aziz, A., König, M., Zengraf, S., & Schulz, J.-U. (2022). Instance Segmentation of Fire Safety Equipment Using Mask R-CNN. *International Conference on Computing in Civil and Building Engineering*, 121–135. https://doi.org/10.1007/978-3-031-35399-4_10

Bayer, H., & Aziz, A. (2022). Object Detection of Fire Safety Equipment in Images and Videos Using Yolov5 Neural Network. *Proceedings of 33. Forum Bauinformatik*.

Chen, J., Lu, W., Fu, Y., & Dong, Z. (2023). Automated facility inspection using robotics and BIM: A knowledge-driven approach. *Advanced Engineering Informatics*, *55*, 101838. https://doi.org/10.1016/j.aei.2022.101838

Clark, A., Lunyov, A., Abramov, D., McCabe, D., White, E., Bonta, J., Savona, J., Story, J., Middleton, K., Tan, L., Ruan, L., Wei, L., Carroll, M., Chen, M., Zhang, M., Hanlon, R., Susla, S., Gunasekaran, S., Keegan, S., . . . and Zheng, Y. (2022). React: Software(Version 18.2.0). https://github.com/facebook/react

Corneli, A., Naticchia, B., Vaccarini, M., Bosché, F., & Carbonari, A. (2020). Training of YOLO Neural Network for the Detection of Fire Emergency Assets. *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*(37), 836-843.

Heinbach, J.-H., & Aziz, A. (2023). *Visual partial inspection of fire safety equipment using machine learning*. Ruhr-Universität Bochum. https://doi.org/10.13154/294-10096

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Michael, K., & Jain, M. (2022). Ultralytics/yolov5: V7.0 - YOLOv5 SOTA Realtime Instance Segmentation. *Zenodo*. Advance online publication. https://doi.org/10.5281/zenodo.7347926

Ramírez, S. (2023). Fastapi: Software(Version 0.100.1). https://github.com/tiangolo/

Rudl, L. (2021). Analysis of CAFM software for construction companies in the Czech. *IOP Conference Series: Earth and Environmental Science*, *900*(1), 12037. https://doi.org/10.1088/1755-1315/900/1/012037

Sanzana, M. R., Maul, T., Wong, J. Y., Abdulrazic, M. O. M., & Yip, C.-C. (2022). Application of deep learning in facility management and maintenance for heating, ventilation, and air conditioning. *Automation in Construction*, *141*, 104445. https://doi.org/10.1016/j.autcon.2022.104445

Sekachev, B., Manovich, N., Zhiltsov, M., Zhavoronkov, A., Kalinin, D., Hoff, B., & … & Sidnev, D. (2020). *OpenCV/CVAT: v1.1.0*. Zenodo.

Soultana, A., & Aziz, A. (2023). Active learning approach for object detection in technical building equipment images. *34. Forum Bauinformatik, 2023*.

Stephens, A. (2023). MongoDB: Software(Version r.87.0.0-rc10), Article Version r.87.0.0-rc10. https://github.com/mongodb/

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (202366). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, *2023*, 7464–7475.

Wen, M.-H., Huang, W.-C., & Wu, Y.-W. (2017). Development of a web-based hybrid BIM cost-estimating system for fire safety engineering. *Civil, Architecture and Environmental Engineering*, 152–161.

Yousefli, Z., Nasiri, F., & Moselhi, O. (2020). Maintenance workflow management in hospitals: An automated multi-agent facility management system. *Journal of Building Engineering*, *32*, 101431. https://doi.org/10.1016/j.jobe.2020.101431