TED UNIVERSITY

**CMPE492/SENG492**

**Computer Engineering Senior Project**

**Software Engineering Senior Project**

# PyroGuard Fire Detection System

**Final Report**

**06.06.2024**

**Team Members:**

Ecem Sıla Gök (1118505616)

Ece Selin Adıgüzel (1988345282)

Zeynep Beyza Uçar (1004263330)

Doruk Aydoğan (1354737307)

# Content Table

Content Table ........ 2
Project Overview ........ 3
  Project Scope ........ 4
  Stages of Development ........ 4
  Deployment ........ 5
  Final Status ........ 5
Impact Analysis ........ 5
  Economic Impact ........ 5
  Environmental Impact ........ 6
  Social Impact ........ 6
  Potential Disadvantages ........ 6
Contemporary Issues ........ 7
Tools and Technologies ........ 9
Research and Resources ........ 11
Testing and Results ........ 12
  Fire Detection Algorithm Subsystem ........ 12
  Testing and Results for Camera ........ 14
  Testing and Results for Alarm ........ 15
  Location Identification Subsystem ........ 15
  Fire-Extinguishing Stations Subsystem ........ 16
  Notification Subsystem ........ 16
User Interface ........ 17
Maintenance and Configuration Subsystem ........ 18
Conclusion ........ 20
Future Works and Improvements ........ 20
References ........ 21

## Project Overview

With the use of deep learning technology and IP cameras, this project includes a system for real-time fire detection and alerting in emergency situations. The key parts of the system architecture are as follows:.

Fire Detection and Analysis Engine: The YOLO (You Only Look Once) algorithm, which we employed to process the live video stream from IP cameras, is the brains behind our system. This is the central component of the system and includes deep learning models for intelligent fire event detection.

Real-Time Processing: This project uses real-time video stream processing to identify any fire events right away. Our layer of data processing and management is designed to process massive amounts of data quickly. Additionally, the IP cameras record the stream in case of an emergency so that it may be accessed at a later time.

Activation of Alarm Subsystem: As soon as a fire becomes apparent, the alarm should sound to alert staff to the situation and to let them know where the fire extinguisher is so they can get to the victims. In order to alert everyone within the building to the possible threat, an audible and visual alarm will be raised.

Notification Module: In addition to setting off the alarm, the system notifies particular users. The messages include details on the location of the fire and the closest firefighting stations.

User Interface: We designed a user-friendly dashboard using Qt Designer and implemented it with PyQt5. The key features of the UI include:

Username Display: Shows the current logged in user's name in the top-left corner. (Sign out etc.)

IP Camera Input Field: Allows users to enter the IP address of the camera they want to monitor.

Camera Feed Display: Displays camera feeds in real-time after the user enters the camera IP into the input field.

Log History: Shows a history of logs for easy reference for any situation.

Available Cameras List: Lists all cameras and marks the available cameras.

Network Indicators: Displays network latency and connection levels.

System Health: Displays CPU, Storage usage and overall System Health.

Alarm Buttons: There are 3 buttons for the alarm system to enable, disable the alarm system and manual emergency alarm button for any cases.

Other Buttons: There are Assistance, Delete Recordings and Restart System buttons for any case of usage.

## Project Scope

The PyroGuard project develops an end-to-end solution that accurately detects the presence of fire from camera footage in real-time. The YOLOv8x deep learning algorithm is used to process video frames to pinpoint the fire's location and activate an alarm system promptly. It additionally detects the closest fire extinguishing system to the fire and sends locations via SMS and email to ensure a quick response and a safe approach.

## Stages of Development

**Requirement Analysis**

The first and foremost task was to gather the requirements of a real-time fire detection system via IP cameras using the YOLO algorithm. It shall comprise high accuracy, real-time processing, easy integration, user-friendly interface and timely notifications.

**Design Phase**

**We focused on designing the system architecture for the following::**

- Fire Detection Engine: Our fire detection engine is based on YOLO, which is precise and time-efficient.
- Data Processing Layer: The layer that handles a gargantuan amount of data with the least latency.
- User Interface:  Descriptive Dashboard using Qt Designer and PyQt5
- Alarm and Notification System: Real-time alarms and notifications.

**Development Phase**

**We implemented:**

- YOLO-Based Fire Detection Engine: Real-time, accurate detection.
- IP Camera Integration: To assure the processing of video feeds without any obstruction.
- Alarm Subsystem: Timely alarms with instant alarms have both visual and audible modes.
- Notification Subsystem: Users' timely alert regarding the exact location and the details of the extinguisher.

**Testing Phase**

We tested the following thoroughly::

- Accuracy: High detection and minimal false alarms.
- Real-Time Processing: Detection and response is instantaneous.
- System Reliability: In different conditions.
- User Interface: Usability and Clarity of Information

## Deployment

We deployed the system, ensuring that it satisfied all the following:

- Installation and Configuration:  Over the existing infrastructure
- Training: About the system and using the User Interface to all its users
- Final Testing: Ensuring the system is actually operational in a live environment

## Final Status

The project was successfully implemented with the following actual results:

- Real-Time Fire Detection: The system is capable of detecting fire in real-time with accuracy based on the YOLO model and IP camera feeds.
- Reliable Alarm System: The system is able to activate the alarm the moment a fire is detected to ensure immediate responses.
- Effective Notification System: The right key people are timely alerted to ensure a quick response.
- User-Friendly Dashboard: Users can effectively monitor live feeds, view logs, and manage settings on the dashboard designed with Qt Designer and PyQt5.

# Impact Analysis

## Economic Impact

Impact of PyroGuard on economic aspects can be examined under three articles which are cost effectiveness, insurance benefits and long term savings.

Cost Effectiveness: Preventing fire damage by using only cameras that are already installed on the structure is making it cost effective as PyroGuard does not need any other physical installation by the businesses, it comes with its own alarm system.

Insurance Benefits: Businesses that use PyroGuard will be more protected against fire damage so that they will have lower insurance costs.

Long Term Savings: As businesses use PyroGuard will be more protected from fire damages, they will avoid repair costs caused by fire damages thanks to the ability to detect fire at an early stage.

## Environmental Impact

Prevention of Major Fires: PyroGuard system detects the risk of a fire building up into a major disaster, so that it prevents larger environmental damages.

Reduction in Carbon Emission: PyroGuard system leads to a reduction in carbon emission from larger scaled fires.

Sustainability: This is achieved by making effective use of the current infrastructure, IP cameras, and sophisticated algorithms to ensure resource efficiency and environmental sustainability.

## Social Impact

Improved Safety: The main benefit is that early fire incident detection and reporting improves people's safety in their homes, workplaces, or other industrial settings.

Community Trust: By providing people with assurances about their safety, advanced fire detection systems foster the trust of the community in which they are installed.

Awareness and Preparedness: Encourages the public to take proactive safety precautions by raising their awareness of fire safety.

## Potential Disadvantages

One-time Cost: The initial expenses of investing in advanced fire detection equipment and technology may be prohibitive for smaller towns or enterprises.

Maintenance and Training: Over time, maintenance and user training increase operating costs and necessitate the allocation of specific resources.

Privacy Concerns: Since consumers may have privacy issues when using IP cameras for fire detection, stringent data security procedures and explicit privacy regulations are required.

## Contemporary Issues

### Global, Economic, Environmental, and Societal Context

The PyroGuard Fire Detection System addresses several contemporary issues, particularly in the context of increasing fire incidents in industrial and commercial settings such as ports and factories.

- Increasing Fire Incidents: There has been a notable rise in fire incidents in critical environments like ports and factories. These areas are highly flammable and require robust fire detection systems to prevent catastrophic damage. PyroGuard's early detection capabilities are crucial in such high-risk settings.
- Early Detection of Sparks and Smoke: PyroGuard excels in detecting fires at their earliest stages, including sparks and smoke. This early detection is vital for preventing fires from escalating and causing significant damage. The system's real-time detection capabilities ensure that any fire is identified and addressed immediately, minimizing potential harm.
- Real-Time Alerts and Notifications: The system provides real-time alerts to operators, including the location of the nearest fire extinguisher. This immediate information allows for swift and effective firefighting efforts. Additionally, PyroGuard's robust database stores user information, IP addresses, and fire incident data, ensuring that all registered users are promptly notified in case of an emergency.
- Comprehensive Database: PyroGuard maintains a powerful database that logs all user interactions and fire incident data. This database supports the system's ability to send notifications to all registered users, ensuring widespread awareness and quick response in case of a fire. The data collected also aids in analyzing fire patterns and improving detection accuracy over time.
- Integrated Alarm System: The system includes its own alarm subsystem, which activates immediately upon fire detection. This integrated approach ensures that all necessary alerts and notifications are managed seamlessly within the system, reducing response times and enhancing overall safety.
- User-Friendly Interface: PyroGuard is designed with a user-friendly interface that simplifies interaction for operators. The clear and intuitive design ensures that users can quickly understand and respond to fire alerts, enhancing the system's effectiveness. The interface also supports easy configuration and maintenance, addressing any potential compatibility issues efficiently.

Impact and Benefits

- Economic Efficiency: By utilizing existing camera infrastructures, PyroGuard minimizes additional hardware costs, making it a cost-effective solution for advanced fire detection.
- Environmental Protection: Early detection and intervention help prevent large-scale fires, reducing environmental damage and pollution. This contributes significantly to environmental conservation efforts.
- Enhanced Safety: Real-time detection and immediate alerts enhance safety in high-risk environments like factories and ports. PyroGuard helps protect both people and property from fire-related hazards by providing timely and accurate information.

Subsystem Integration: PyroGuard integrates multiple subsystems, including alarm systems, user interfaces, and notification systems, to provide a comprehensive fire detection and response solution. This holistic approach ensures all aspects of fire safety are effectively addressed.

PyroGuard's advanced features and integrated approach make it a critical tool in addressing contemporary fire safety challenges, providing reliable, efficient, and cost-effective protection in critical environments.

## Tools and Technologies

The PyroGuard Fire Detection System project employed various new tools and technologies to achieve its goals of early fire detection, real-time alerts, and comprehensive fire safety management. These tools were integral to the development, implementation, and effectiveness of the system.

Arduino IDE and Serial Communication
Purpose: Used for developing the alarm subsystem.

Usage: The Arduino IDE, along with the Serial library, was utilized to program and control the alarm system components. The alarm subsystem includes a buzzer connected to an Arduino UNO board, which communicates with the main PyroGuard system via USB (COM6 port).
Effectiveness: This setup allowed for reliable and immediate activation of the alarm upon fire detection, ensuring that audible and visual alerts are triggered to warn users of a fire incident.

Ekomesaj API and Gmail API
Purpose: Used for the notification subsystem.
Usage: The Ekomesaj API was employed to send SMS notifications to users' phones, while the Gmail API was used to send email alerts. These APIs were integrated into the PyroGuard system to ensure that all registered users receive real-time notifications about fire incidents.
Effectiveness: The integration of these APIs enabled the system to promptly notify users through multiple channels, enhancing the overall responsiveness and effectiveness of the fire detection system.

Google Cloud (GCloud)
Purpose: Used for cloud-based notification and data storage.
Usage: GCloud services were leveraged to handle the backend processes of sending notifications and storing user data. This included managing user registrations, logging fire incidents, and ensuring data availability and security.
Effectiveness: Utilizing GCloud provided scalability and reliability, ensuring that the system could handle a large number of users and fire incidents without performance degradation.

YOLO (You Only Look Once)
Purpose: Used for the fire detection model.
Usage: YOLO, a state-of-the-art deep learning model for real-time object detection, was used to identify fire and smoke in camera footage. The model was trained using a comprehensive dataset of fire images.
Effectiveness: YOLO's high accuracy and speed were crucial for the real-time detection capabilities of PyroGuard, allowing for immediate identification of fire and smoke, thereby enabling prompt alerts and responses.

Roboflow
Purpose: Used for data labeling and management.
Usage: Roboflow was utilized to create and manage the dataset for training the YOLO model. This involved annotating images with labels indicating the presence of fire, smoke, and other relevant features.
Effectiveness: The use of Roboflow streamlined the data preparation process, ensuring high-quality annotations and an efficient workflow for training the fire detection model.

QGIS (Quantum GIS)
Purpose: Used for determining the coordinates of fire extinguishers.

Usage: QGIS was employed to map and identify the exact locations of fire extinguishers within the monitored area. This geographical data is integrated into the PyroGuard system to provide precise information about the nearest fire extinguisher when a fire is detected.
Effectiveness: Using QGIS ensures accurate spatial data, enabling the system to direct users to the closest fire extinguisher swiftly, thus enhancing the effectiveness of the fire response.

Integration and Effectiveness

- Alarm System: The use of Arduino and Serial communication ensured that alarms are triggered immediately upon detection, providing a critical first response to fire incidents.
- Notification System: The Ekomesaj and Gmail APIs, supported by GCloud, ensured that all users are promptly informed about fire incidents through multiple channels, enhancing overall safety.
- Fire Detection Model: YOLO, trained with data managed by Roboflow, provided accurate and real-time detection of fire and smoke, which is essential for early intervention and minimizing damage.
- Geographical Mapping: QGIS enabled the precise identification of fire extinguisher locations, which is crucial for efficient and effective firefighting efforts.

## Research and Resources

Primarily, the following articles, which our supervisor supported, were scanned as our main problem was fire detection. Model performances were compared. The most suitable attributes for our own project were selected and included in our project. The main articles and studies we used are as follows: Li, P., & Zhao, W. (2020). Image fire detection algorithms based on convolutional neural networks. *Case Studies in Thermal Engineering, 18*, 100625, Jadon, A., Omama, M., Varshney, A., Ansari, M. S., & Sharma, R. (Year). FireNet: A specialized lightweight fire & smoke detection model for real-time IoT applications. *Proceedings of the IEEE Conference/Journal*, pages  and Author, A. A. (2020). Recent advances in fire detection and monitoring systems: A review. In *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETITi 18)* (Vol. 1, pp. 332-340). Springer. After our research, we used the YOLO library, which employs an end-to-end neural network to simultaneously predict bounding boxes and class probabilities. Links to a sample project [1] that provided inspiration to our designs can be found in the references.

Apart from this, we did a small-scale microprocessor research and research on programming for alarm activation. We did this using Arduino Uno and tutorials.

# Testing and Results

## Fire Detection Algorithm Subsystem

The Fire Detection Algorithm Subsystem of PyroGuard, utilizing the YOLOv8x model trained on a dataset of 5200 images, underwent comprehensive testing.

Accuracy Calculation

The model achieved an accuracy of approximately 83%, indicating that the majority of instances were correctly classified.
Accuracy ≈ 0.830

Precision, Recall, and F1 Score

- Fire Precision ≈ 0.94
- Smoke Precision ≈ 0.96
- Fire Recall ≈ 0.79
- Smoke Recall ≈ 0.90
- Fire F1 Score ≈0.86
- Smoke F1 Score ≈ 0.93

The precision and recall metrics are crucial for understanding the model's performance. High precision (0.94 for fire and 0.96 for smoke) indicates that most of the detected instances are correct, while high recall (0.79 for fire and 0.90 for smoke) shows that the model successfully identifies most of the actual instances.

Curve Analysis
F1-Confidence Curve (Figure 1): This curve shows how the F1 scores change with confidence levels. The model maintains high F1 scores for both fire and smoke across various confidence levels, indicating a balanced precision and recall.

Precision-Confidence Curve (Figure 2): High precision across different confidence thresholds demonstrates the model's effectiveness in minimizing false positives.

Precision-Recall Curve (Figure 3): With a mean average precision (mAP) of approximately 0.89, the model shows strong performance in distinguishing fire and smoke from other classes.

Recall-Confidence Curve (Figure 4): High recall across confidence levels ensures that most fire and smoke instances are detected, crucial for early intervention.

Overall Assessment
The testing phase demonstrated that the YOLOv8x model is highly effective in detecting fire and smoke in real-time.

High Accuracy: The model's accuracy of 83% indicates its reliability in classifying fire and smoke instances correctly. In real-world applications, where precise detection might avert serious harm, the system's high accuracy is essential to its credibility.

Balanced Precision and Recall: With high precision values of 0.94 and 0.96 for fire and smoke, respectively, the model indicates that most of the identified cases are real fire and smoke. The model successfully detects the majority of fire and smoke occurrences, as evidenced by the recall values of 0.90 for smoke and 0.79 for fire. This balance ensures that the model minimizes false positives and false negatives, providing a robust detection mechanism.

Robust Performance Across Confidence Levels: The F1-Confidence and Precision-Confidence curves demonstrate that the model maintains high performance across various confidence thresholds. Robustness is essential for real-world applications since the detecting system needs to perform consistently in a range of situations.

Effective Early Detection: Because of the high recall values, the majority of fire and smoke occurrences are found early on, which is essential for prompt response and halting the spread of fire incidents. The model's capacity to identify flames in the spark and smoke phases guarantees that possible calamities are prevented before they result in considerable harm.

Real-World Effectiveness: The testing findings show that the model functions well in both controlled testing settings and real-world situations. The high precision and recall values, along with the robust performance across confidence levels, validate the model's effectiveness in practical applications. The model's ability to handle various fire and smoke conditions makes it suitable for deployment in critical environments such as factories, ports, and industrial sites.

Potential for Further Improvement

Dataset Expansion: Extending the dataset and adding more varied fire and smoke situations can improve the model's accuracy and robustness.

Model Fine-Tuning: Modifying the parameters of the model can help increase recall and precision while lowering the incidence of false positives and negatives.

Comprehensive Real-Time Testing: To confirm the model's efficacy and pinpoint areas for development, more comprehensive real-time testing under various environmental circumstances should be carried out.

## Testing and Results for Camera

During the integration of the camera with the fire detection model, it was discovered that it had missing data when calculating the fire location. That's why database attributes were updated. Variables such as height, angle, azimuth and focal length (to calculate the field of view) were added to enable the necessary trigonometric calculations to identify the fire location. As our system is only suitable for working with cameras for which we have such information, therefore integration with various coordinate and compass tools were evaluated to provide this.

After all these problems were resolved, our camera was included in our program with its IP address, and the results were successfully obtained, where calculations were made in accordance with the data flow.

## Testing and Results for Alarm

In our demo app, we used an Arduino processor to test the alarm system. This design has two LEDs and a buzzer. The test was done to test whether the LEDs light up and the buzzer rings when a fire detected frame is captured.

In a test application we tried with our application, our demo alarm was triggered and beeped as expected at the time of fire detection. The integration is set via CH340 Driver.

After testing the detect scenario, to turn the alarm off an alarm override code that was previously specified was used to test the alarm overriding functionalities.

## Location Identification Subsystem

The location identification subsystem aims to correctly identify the location of the fire using the information from the captured image, as well as the stored camera information.

This subsystem was subjected to unit testing, as well as accuracy and integration testing in various different scenarios, such as weather conditions and varying fire sizes.\

For integration tests all functions of the location identification code were checked to ensure that there are no problems with the data flow. Here, the connections between the YOLO model and the database are crucial for accurate calculations.
Although no problems occurred during the integration or unit tests, during the accuracy testing it became obvious that the accuracy of the algorithms we designed to calculate distances were simply not enough, as the algorithm based the calculations on mostly two dimensions rather than three.
Since accurate identification of the fire’s real-life coordinates is a crucial part of this project, we then decided to update or algorithm with a new algorithm that uses the pixel coordinates of the fire in the detected image, alongside the height of the camera, the angle between the camera and the pole or the wall it is attached to, the focal length of the camera so that the field of vision can be calculated, the azimuth angle of the camera, and

the real-life coordinates of the camera. Using these, we were able to use trigonometric functions and geometry to accurately calculate the location of the fire.

After this change in the calculation algorithm, we were able to obtain an accurate estimation of fire locations.

## Fire-Extinguishing Stations Subsystem

The fire-extinguishing stations subsystem utilizes the location identification subsystem to pinpoint the fire and uses this with the information held in the database to compare the location of the fire with the fire extinguishers.

This was provided by using an optimal comparison algorithm that works in linear time as swift responses are amongst our biggest concerns in emergency situations, alongside the accuracy of the response. To ensure this, response time and accuracy values were tested to ensure they were within acceptable limits. Data flow was observed and tested to ensure integrations to the location identification subsystem, the database, and the maintenance subsystem were done correctly. It is especially important that the values stored in the database are correct and are maintained properly in subsystems that hold coordinate values.

## Notification Subsystem

The notification subsystem is used to notify the necessary personnel of the fire situations that have occurred. In order to do this, it interacts with the fire-extinguishing stations subsystem, alongside the database and two external APIs.

The integration between the subsystems were checked to make sure no data was lost in the transaction and everything was transferred correctly.

The connection between the subsystem and the APIs were tested in many cases through sending SMS messages and emails. Although at one point Google Cloud did not recognize the credentials, which caused the emails that were being sent to look like spam mails, this was fixed through corrections of the credentials and correctly permitting authorizations.

After ensuring the API connections were done accurately and stably, they were tested with various test-cases, which they completed with no errors.

## User Interface

### User Login and Logout

The system should display the correct username in the "HELLO, USERNAME" field when logged in with the correct username and password and when the logout button is pressed the system should log the user out and return to the login screen. When this was tested, the login and logout operations were performed correctly and the username was accurately displayed.

### Entering and Monitoring Camera IP

The system should verify that the user can enter an IP address for a camera and monitor the feed. The testing process showed that the input was correctly taken, displayed, and updated.

### Enabling and Disabling Alerts

Alerts should have an option to be enabled and disabled. This is provided by disable alert and enable alert buttons, which change the preference of enabling alerts upon pressing them. This was tested by disabling the alert and then manually activating the alarm. The test was passed by not activating the alarm when it is disabled.

### Emergency Alert Activation

The emergency alert system was tested to manually activate alarm systems and emergency protocols, so if the system happens to fail, there is a backup option to manually warn people. This was tested by clicking the emergency alert button, which immediately triggers all alarms and notifications.

### Monitoring System Health and Performance

The system metrics such as network latency, connection level, CPU usage, storage usage, and overall system are monitored to make sure the system does not fail in any way, and to be able to quickly detect and intervene when it does. When tested all metrics were displayed accurately.

### Maintenance and Configuration Subsystem

The maintenance and configuration subsystem is responsible for checking and updating the data in the database as well as amongst the subsystems, mainly through the use of triggers in the database. These triggers are set to check conditions such as inconsistent data entry, removing and altering components that affect other subsystems. In these situations, it either does not allow the insertion or the deletion of the component without fixing the dependencies or inconsistencies, or where possible updates the altered information on the other subsystems as well.

To ensure all systems were correctly integrated, data flow was monitored and tested extensively. Triggers were also tested in the processes of altering the database to add more values into the ‘cameras’ schema that is mentioned under the title Location Identification Subsystem.

No problems were detected in this subsystem through any of these processes.

## Conclusion

The PyroGuard Fire Detection System offers an end-to-end approach to the age-old fire detection problem. With the emphasis on swift response in emergency situations, it combines hardware and software components to not only detect fire, but to ensure the necessary authorities are alerted as well as the people in the building, so that the situation can be handled rapidly and averted with minimal damage and loss.

The system has a high accuracy rate of 83% in detecting fires, with minimal false positives and negatives while monitoring and processing video feeds in real-time, ensuring prompt detection and response. All subsystems are integrated seamlessly to provide correct and fast outcomes. Updates and maintenance processes are handled without fault to keep the system up-to-date and reliable in all conditions.

## Future Works and Improvements

While the PyroGuard Fire Detection System has successfully fulfilled its main requirements, there are many ways in which it could be improved in terms of accuracy, adaptability, and effectiveness.

The fire detection subsystem, however already has a high accuracy, could improve significantly in means of accuracy if more advanced deep learning algorithms can be utilized using a higher computation power and a more diverse and larger dataset.

Although already utilized in the project, QGIS or other systems can be integrated further into the system, especially for keeping precise locations, and calculating azimuth locations in order to get much more accurate results in calculating the location of the fire and determining the closest fire extinguishers.

This system could be combined with audible warning systems alongside alarms to give instructions to the people that may be in the building.

Instead of only notifying people of the quickest way to put out the fire, automated fire suppression mechanisms and other prevention and emergency technologies could be combined with the existing system to stop the fire automatically as soon as it is detected. Alongside this local emergency numbers could be automatically notified upon the detection of fire.

For larger-scaled projects, cloud systems could be utilized for scalable storage, efficient data processing and computing, and optimization.

# References

[1] YOLOv8 Fire and Smoke Detection on GitHub:

Abonia1. (n.d.). YOLOv8 Fire and Smoke Detection. GitHub. Retrieved June 6, 2024, from https://github.com/Abonia1/YOLOv8-Fire-and-Smoke-Detection

[2] YOLO: Real-Time Object Detection:

Redmon, J. (n.d.). YOLO: Real-Time Object Detection. PJ Reddie. Retrieved June 6, 2024, from https://pjreddie.com/darknet/yolo/

[3] Google Cloud Console:

Google. (n.d.). Google Cloud Console. Retrieved June 6, 2024, from https://console.cloud.google.com/

[4] Ekomesaj SMS API Documentation:

Ekomesaj. (n.d.). Ekomeaj SMS API Documentation. Retrieved June 6, 2024, from https://panel4.ekomesaj.com/api-docs/?id=eyJwcm9kdWN0IjoiRWtvbWVzYWoiLCJkb21haW4iOiJwYW5lbDQuZWtvbWVzYWouY29tIiwicil6dHJ1ZX0=

[5] Gmail API Guides:

Google. (n.d.). Gmail API Guides. Retrieved June 6, 2024, from https://developers.google.com/gmail/api/guides?hl=tr

[6] Using Gmail API in Python:

The Python Code. (n.d.). How to Use Gmail API in Python. Retrieved June 6, 2024, from https://thepythoncode.com/article/use-gmail-api-in-python

[7] Sending Email with Python and Gmail:

Mailtrap. (n.d.). How to Send Email Using Gmail API in Python. Retrieved June 6, 2024, from https://mailtrap.io/blog/python-send-email-gmail/

[8] Focal Length and Field of View on Wikipedia:

Wikipedia. (n.d.). Focal Length. Retrieved June 6, 2024, from https://en.wikipedia.org/wiki/Focal_length#:~:text=Focal%20length%20(f)%20and%20field,the%20width%20of%20the%20film

[9] Understanding Focal Length and Field of View:

Edmund Optics. (n.d.). Understanding Focal Length and Field of View. Retrieved June 6, 2024, from https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/

[10] Finding Focal Length from Image Size and FOV:

Photo Stack Exchange. (n.d.). Finding Focal Length from Image Size and FOV. Retrieved June 6, 2024, from https://photo.stackexchange.com/questions/97213/finding-focal-length-from-image-size-and-fov

[11] Azimuth Angle on PV Education:

PV Education. (n.d.). Azimuth Angle. Retrieved June 6, 2024, from https://www.pveducation.org/pvcdrom/properties-of-sunlight/azimuth-angle

[12] Understanding Azimuth and Elevation:

PhotoPills. (n.d.). Understanding Azimuth and Elevation. Retrieved June 6, 2024, from https://www.photopills.com/articles/understanding-azimuth-and-elevation

[13] Calculating Distance Between Two Points in 3D:

GeeksforGeeks. (n.d.). Program to Calculate Distance Between Two Points in 3D. Retrieved June 6, 2024, from https://www.geeksforgeeks.org/program-to-calculate-distance-between-two-points-in-3-d/

[14] Calculating Camera Scene Distance:

Ambarish, A. (n.d.). How to Calculate Camera Scene Distance. LinkedIn. Retrieved June 6, 2024, from https://www.linkedin.com/pulse/how-calculate-camera-scene-distance-ambarish-34msf

## Appendix A: Model Curves of the Trained Deep Learning Model For Fire Detection

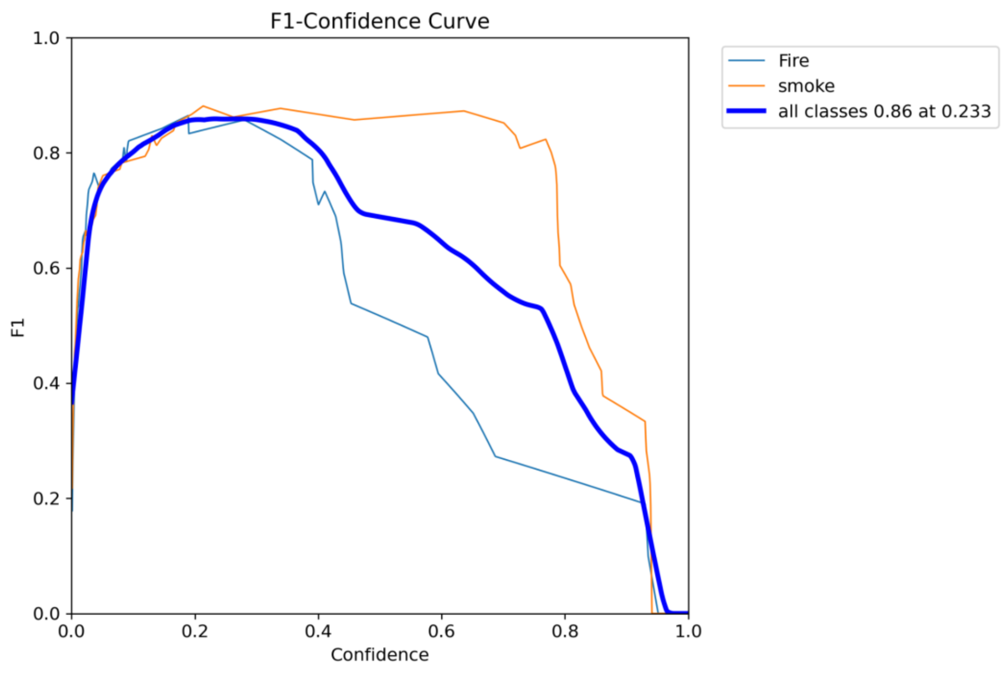


Figure 1: F1-Confidence Curve

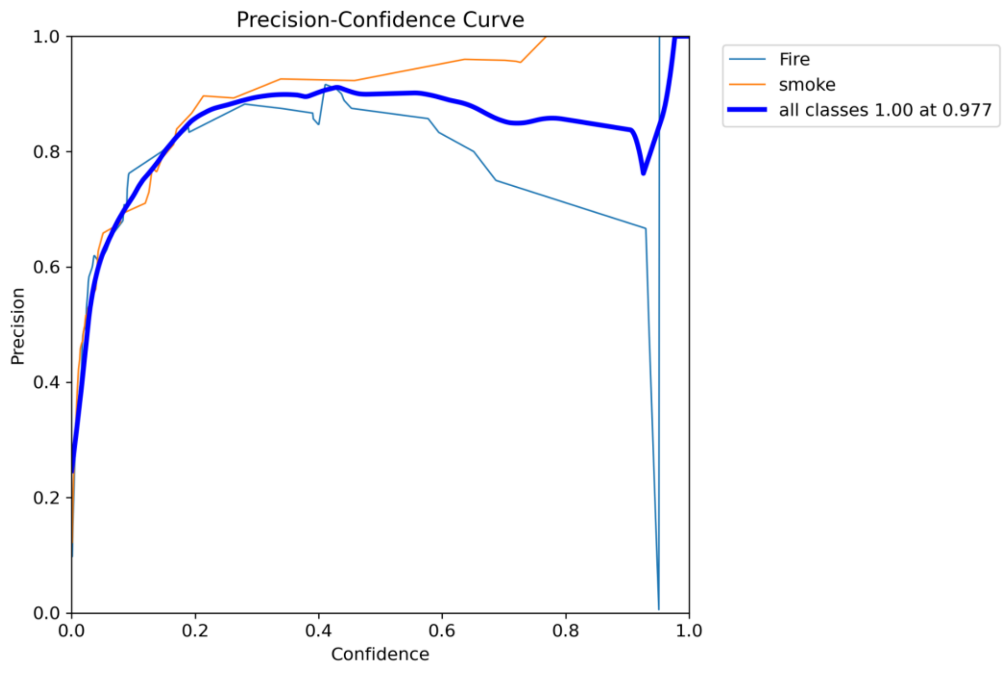


Figure 2: Precision-Confidence Curve

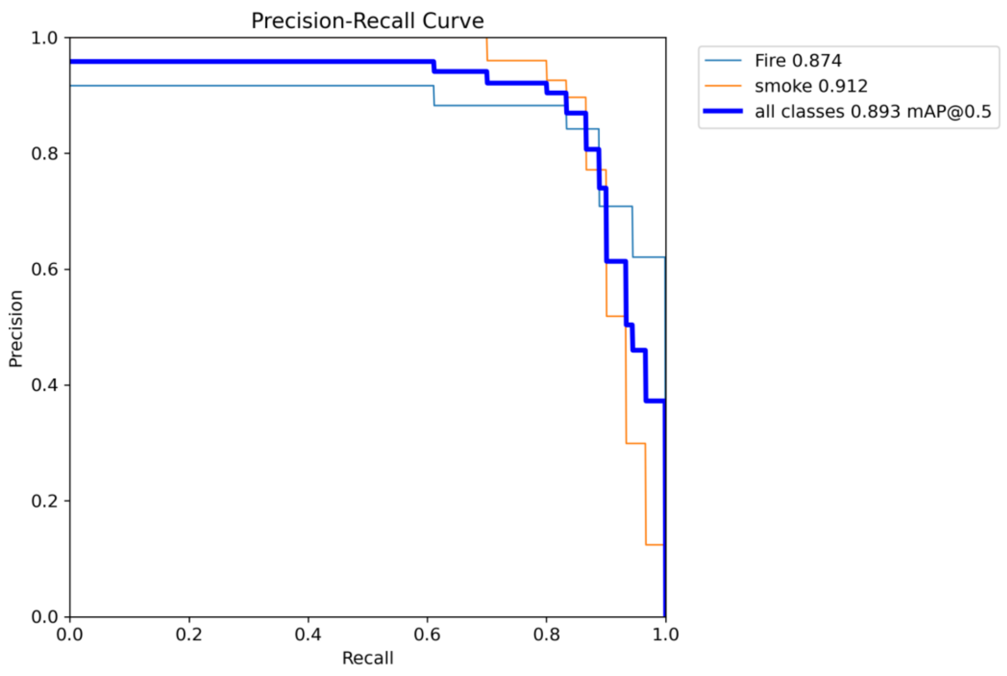


Figure 3: Precision-Recall Curve

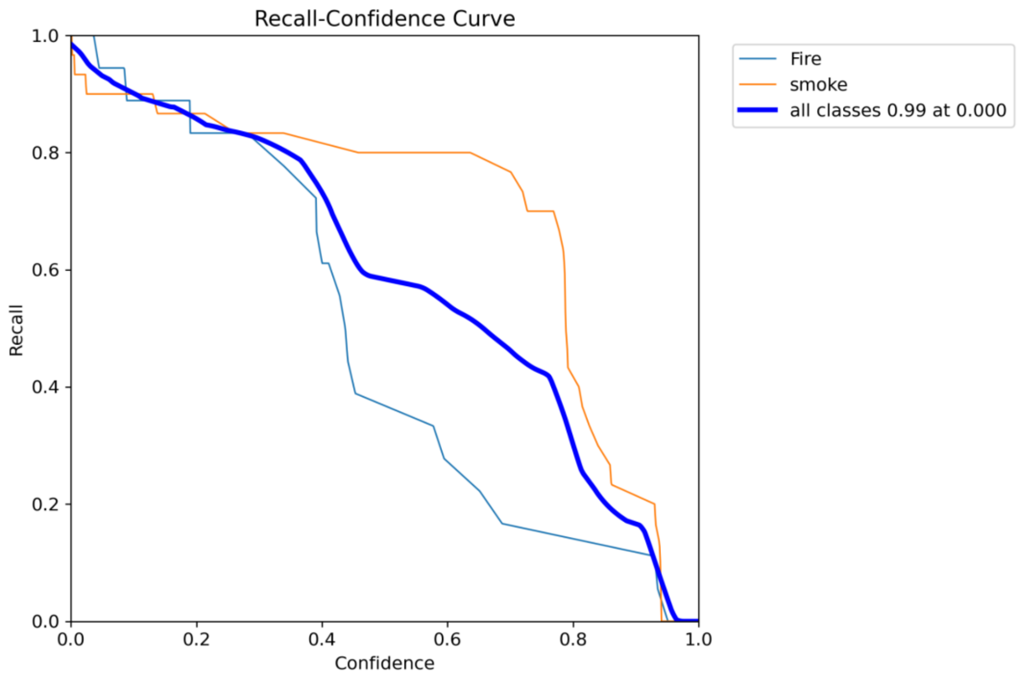
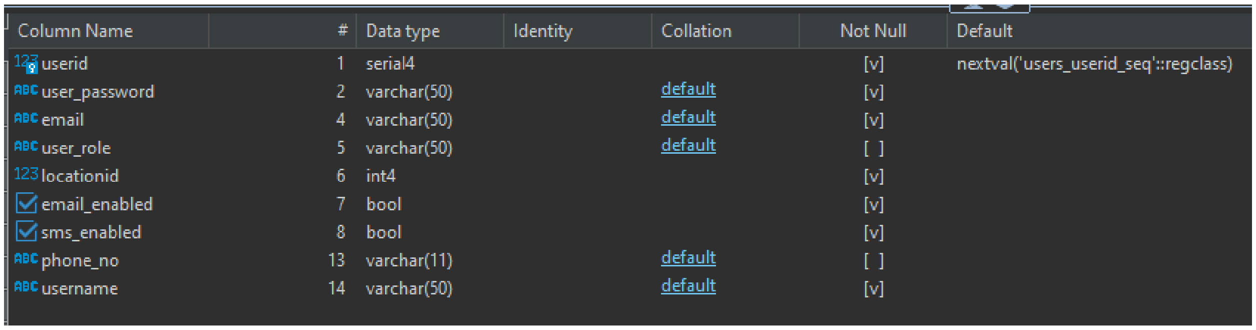


Figure 4: Recall-Confidence Curve

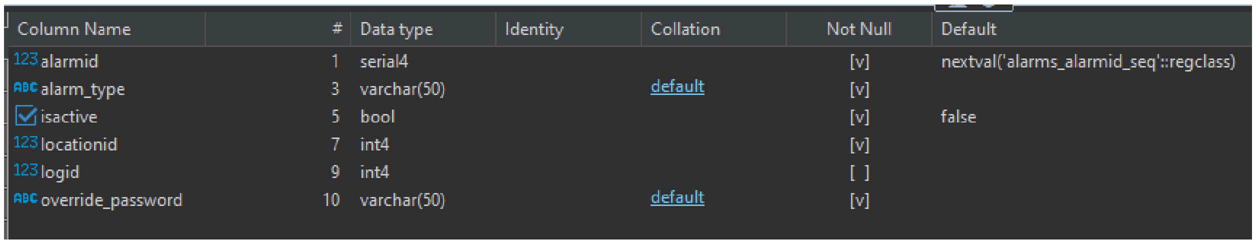
## Appendix B: Database Schemas and Attributes



| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| userid | 1 | serial4 | | | [v] | nextval('users_userid_seq'::regclass) |
| user_password | 2 | varchar(50) | | default | [v] | |
| email | 4 | varchar(50) | | default | [v] | |
| user_role | 5 | varchar(50) | | default | [ ] | |
| locationid | 6 | int4 | | | [v] | |
| email_enabled | 7 | bool | | | [v] | |
| sms_enabled | 8 | bool | | | [v] | |
| phone_no | 13 | varchar(11) | | default | [ ] | |
| username | 14 | varchar(50) | | default | [v] | |

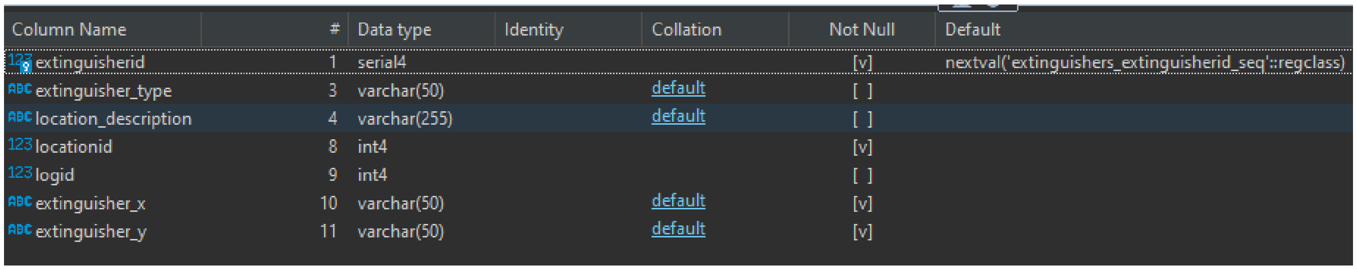
Figure 5: Users Schema

a



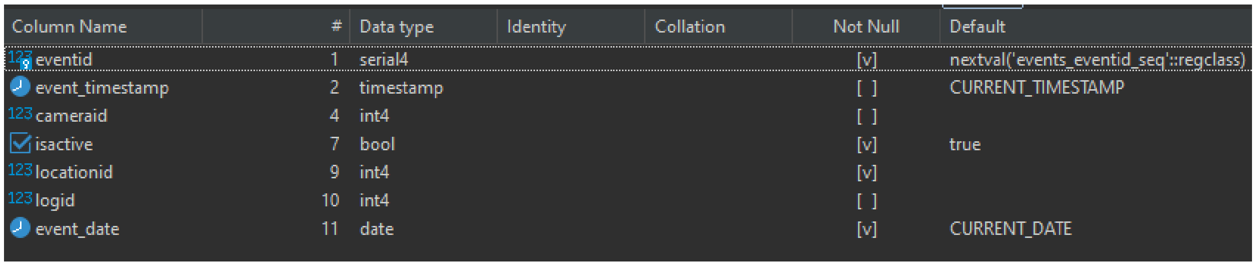
| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| alarmid | 1 | serial4 | | | [v] | nextval('alarms_alarmid_seq'::regclass) |
| alarm_type | 3 | varchar(50) | | default | [v] | |
| isactive | 5 | bool | | | [v] | false |
| locationid | 7 | int4 | | | [v] | |
| logid | 9 | int4 | | | [ ] | |
| override_password | 10 | varchar(50) | | default | [v] | |

Figure 6: Alarms Schema



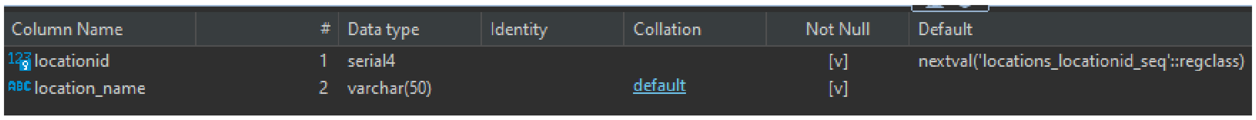
| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| extinguisherid | 1 | serial4 | | | [v] | nextval('extinguishers_extinguisherid_seq'::regclass) |
| extinguisher_type | 3 | varchar(50) | | default | [ ] | |
| location_description | 4 | varchar(255) | | default | [ ] | |
| locationid | 8 | int4 | | | [v] | |
| logid | 9 | int4 | | | [ ] | |
| extinguisher_x | 10 | varchar(50) | | default | [v] | |
| extinguisher_y | 11 | varchar(50) | | default | [v] | |

Figure 7: Extinguishers Schema



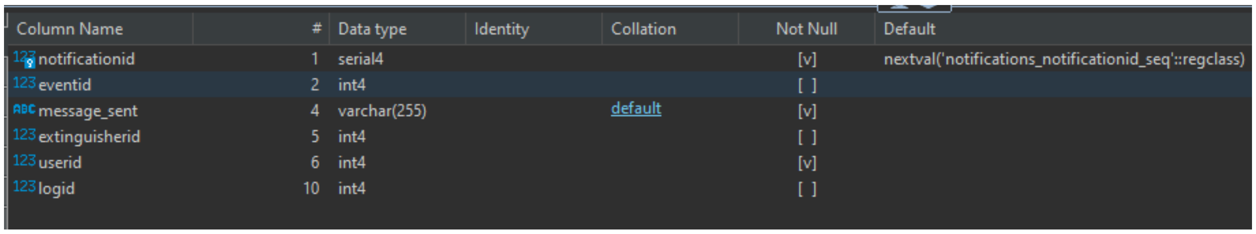
| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| eventid | 1 | serial4 | | | [v] | nextval('events_eventid_seq'::regclass) |
| event_timestamp | 2 | timestamp | | | [ ] | CURRENT_TIMESTAMP |
| cameraid | 4 | int4 | | | [ ] | |
| isactive | 7 | bool | | | [v] | true |
| locationid | 9 | int4 | | | [v] | |
| logid | 10 | int4 | | | [ ] | |
| event_date | 11 | date | | | [v] | CURRENT_DATE |

Figure 8: Events Schema



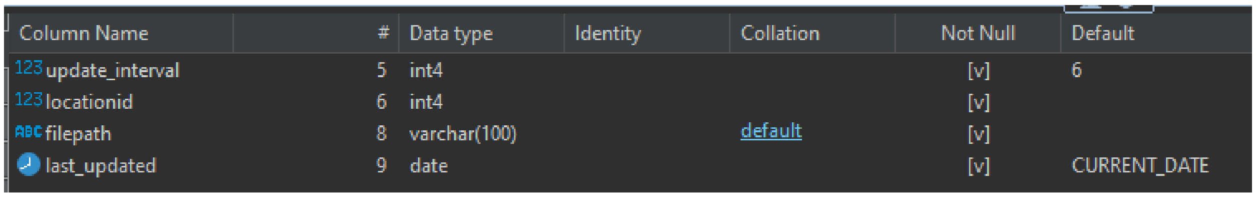
| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| locationid | 1 | serial4 | | | [v] | nextval('locations_locationid_seq'::regclass) |
| location_name | 2 | varchar(50) | | default | [v] | |

Figure 9: Locations Schema



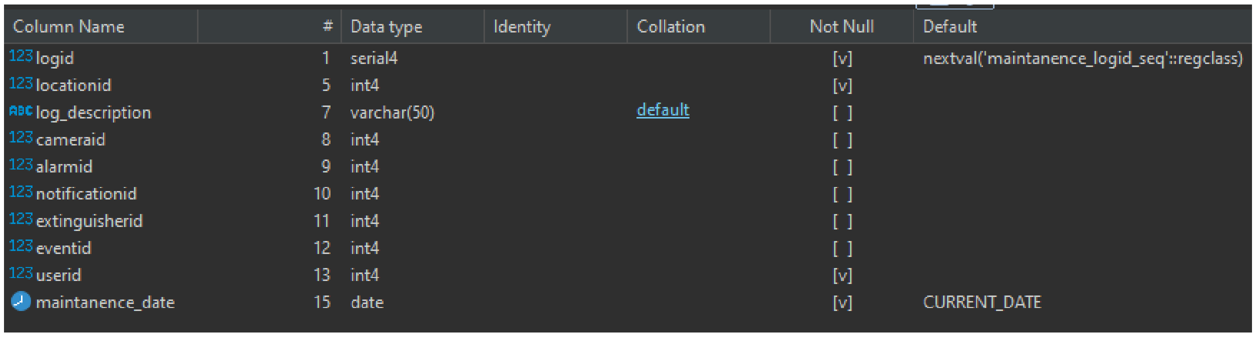
| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| notificationid | 1 | serial4 | | | [v] | nextval('notifications_notificationid_seq'::regclass) |
| eventid | 2 | int4 | | | [ ] | |
| message_sent | 4 | varchar(255) | | default | [v] | |
| extinguisherid | 5 | int4 | | | [ ] | |
| userid | 6 | int4 | | | [v] | |
| logid | 10 | int4 | | | [ ] | |

Figure 10: Notifications Schema



| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| update_interval | 5 | int4 | | | [v] | 6 |
| locationid | 6 | int4 | | | [v] | |
| filepath | 8 | varchar(100) | | default | [v] | |
| last_updated | 9 | date | | | [v] | CURRENT_DATE |

Figure 11: System Configuration Shema



| Column Name | # | Data type | Identity | Collation | Not Null | Default |
|---|---|---|---|---|---|---|
| logid | 1 | serial4 | | | [v] | nextval('maintanence_logid_seq'::regclass) |
| locationid | 5 | int4 | | | [v] | |
| log_description | 7 | varchar(50) | | default | [ ] | |
| cameraid | 8 | int4 | | | [ ] | |
| alarmid | 9 | int4 | | | [ ] | |
| notificationid | 10 | int4 | | | [ ] | |
| extinguisherid | 11 | int4 | | | [ ] | |
| eventid | 12 | int4 | | | [ ] | |
| userid | 13 | int4 | | | [v] | |
| maintanence_date | 15 | date | | | [v] | CURRENT_DATE |

Figure 12: Maintenance Schema