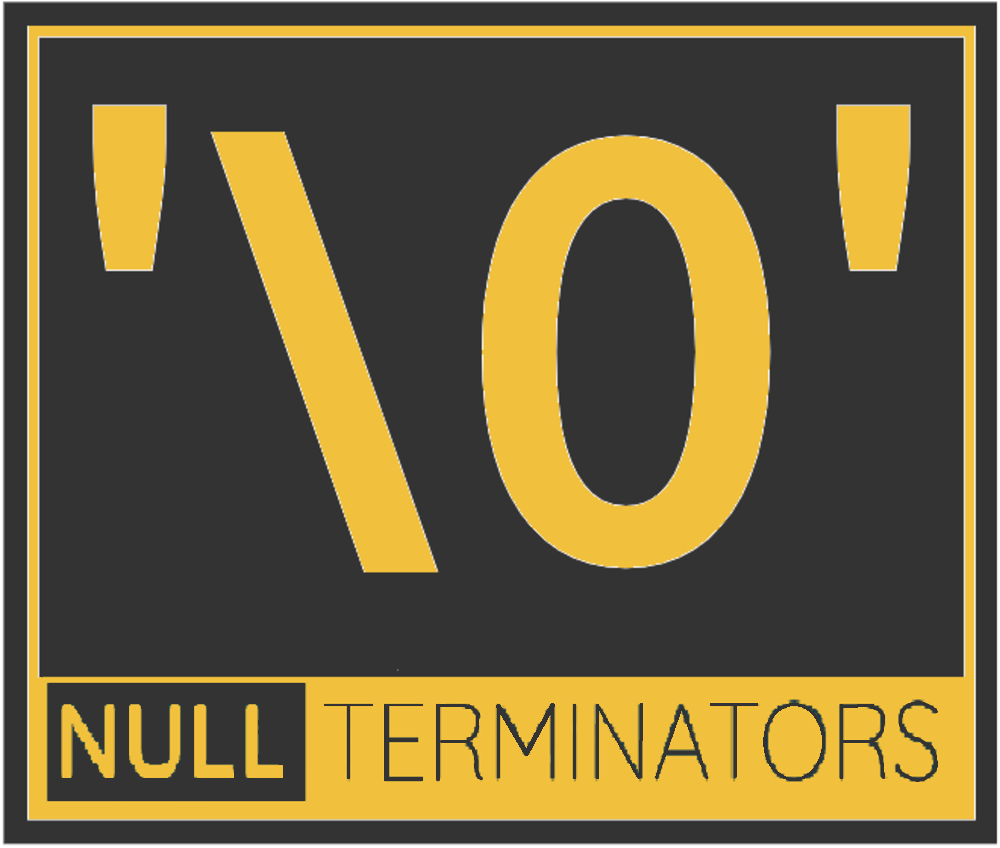
ATICS-

Automated Toolbox Inventory Control System

Requirement Specification

Null Terminators

Version: 2.2





Senior Project Requirement Specifications

CptS 421/423

WSU Tri-Cities

1710 Crimson Way, Richland, WA 99354

**Title:**

Automated Toolbox Inventory Control System

**Project Team:**

Null Terminators

**Participants:**

Reem Osman (Project Manager)

Caitlyn Powers

Steven Pixler

Navin Sabandith

Caleb Thomas

**Client:**

Troy Stokes, HiLine Engineering

**Client Technical Contact:**

Andrew Tolman, HiLine Engineering

**Supervisor:**

Dr. Neil Corrigan, WSUTC

**Table of Contents**

[**Revision History 4**](#_tul8t8cqyjgg)

[**1.0 Introduction 5**](#_pz46ovoyh9lq)

[**2.0 Background 6**](#_rikxtn27jlq1)

[**3.0 Overview 7**](#_cwnz924zf311)

[**4.0 Environment 12**](#_clsnsqu7pbis)

[4.1 Input 12](#_lcluqb3ci28c)

[4.2 Output 13](#_9do6r58wainp)

[**5.0 Operation 14**](#_p7kjnjos2n46)

[5.1 Program operation 14](#_ffssatlygkr7)

[5.1.1 Beginning of operation: 14](#_m80yf82umbvt)

[5.1.2 During operation: 14](#_6xm4h4wg0ner)

[5.1.3 After operation: 16](#_eifrvpw508bj)

[5.2 Database Operation 16](#_qr835mf1lvk7)

[5.2.1 API Specification 16](#_nijg1twhcj6i)

[5.3 Additional Script 17](#_sk0sghy8ypwv)

[**Appendix 18**](#_unzi7lt5w54j)

[1.0 FlowChart 18](#_jcf1n2totbq)

[A diagram of a software applicationDescription automatically generated with medium confidence 18](#_j8c5w8aq8nw5)

[2.0 Program Operation Overview Diagram 19](#_mrm6viiaz4ua)

[Refer to the design document for up to date program operation 19](#_h1g7nbpbbp5e)

[3.0 Prototyping 20](#_holvefkfkpim)

[3.1 Database Prototype 20](#_itgl0qlm3kg5)

[3.2 Test Mode Prototype Output 21](#_41xdyx7yum4v)

[3.3 Record Mode Prototype Output 22](#_pca5wycvshcm)

[3.4 Problems caused by shadows 23](#_c1nhhif50w2i)

[4.0 Configuration files 23](#_5vzd72kmw5xy)

[4.1 Variables in the global configuration file: 23](#_behnlw9whoq0)

[4.2 Variables in the drawer configuration file (stored in the database): 24](#_bvehaxf2odq3)

[5.0 Camera research 24](#_h35j847oid80)

# Revision History

2.2 - 04/27/2024 Update images and cleared ambiguity auth. Reem Osman

2.1 - 02/02/2024 Add UID recognition research conditions auth. Null Terminators

2.0 - 01/30/2024 Updated to add Unique Identifier Recognition auth. Caleb Thomas

1.1 - 12/08/2023 Changed requirement to preferred auth. Caitlyn Powers

1.0 - 12/06/2023 First Published Version auth. Null Terminators

# 1.0 Introduction

As a condition of graduation, Washington State University (WSU) requires bachelor’s students in their final year to complete a “Senior Design Project” showing mastery of the course material covered throughout the program. For students in the Computer Science program, this takes the form of a yearlong project where students form a team and act as a small development firm given the responsibility of completing a software development project for a commercial sponsor.

This document details the requirement specification for the senior design project being completed by the Null Terminators - one of this year’s teams of students. This document includes background material for understanding the problem domain, an overview of the project purpose, and the context in which the final product will operate. It will also specify the input, output, repositories, and operations of the program.

# 2.0 Background

HiLine Engineering approached the Null Terminators and explained the ongoing issues with employees not returning tools to the shop toolboxes from which they were taken. Tools may be taken off-premise, returned to the wrong toolbox, returned to a different drawer within the same toolbox, or otherwise mishandled. HiLine would like an automated, camera-based system that identifies which tools are taken by employees and whether or not they have been returned. This system needs to operate with a minimum amount of cooperation from employees, with the only change from the current tool checkout procedure being scanning an RFID badge to unlock the box.

# 

# 

# 

# 

# 

# 

# 3.0 Overview

* Purpose
  + To develop a toolbox accountability system triggered by RFID badges and utilize video image processing to identify tools checked out by employees.
* Platform
  + Operating System
    - Ubuntu Linux
  + Programming Languages
    - Detection Program: retrieves the camera footage, analyzes the footage and creates records.
      * Python
        + The following libraries will be used:

OpenCV

Template matching and video footage analysis

PyTorch

Train tool recognition model

Numpy

Support for large, multi-dimensional arrays and matrices

Which is what pictures are

PyYAML

For parsing configuration files

ONNX Runtime

Using Onnx tool recognition model.

More libraries may be added, but all libraries will be installable using the command below.

Flask

For the API server.

imutils

For retrieving images from webserver.

jsonschema

For validating signals and responses.

Requests

mysql\_connector\_repackaged

For the API server

Shapely

For use in image processing

Other open source libraries may be used as needed

All libraries will be listed in a requirements.txt file distributed with the finished product

* + - * + All libraries can be installed on the system with the following command:

pip install <library name>

* + - * + If system does not have pip, it may be installed using the command below:

sudo apt install pip

For Debian/Ubuntu Linux

Other platforms may support pip, but will not be used by the Null Terminators

* + - Configuration file: File used for configuring parameters
      * YAML
        + Comments will be included in the file describing the following:

Purpose and name of variable

Variable type (eg. int, string, etc.)

Allowed values (eg. range of valid numbers)

If applicable

default values

How to choose a value

* + - Database: Stores information about tool status, checkout history, and other information
      * HiLine will need to dedicate one server, Virtual machine, or other computer environment to running a Docker container which will contain the database.
      * MySQL
      * Will run inside Docker container
      * 24/7 uptime expected
    - API Server: Interface to retrieve data directly into the database and allows applications to communicate with one another.
      * The Null Terminators will develop an interface (called an API or Application Programming Interface)
        + An API is a special purpose web server able to access information from the database in a predefined and easy to use format.
        + HiLine will be able to use the functions exposed by the API to easily develop their own reporting software, auditing workflow, and any future work desired.
      * Apache Web server
      * Will run inside Docker container
      * 24/7 uptime expected
      * Key based authentication to prevent abuse
      * Single point of contact for queries from database
      * Separate container from Database for ease of migration/maintenance
      * HiLine can use API documentation to develop reporting/auditing tools in the future
* Software execution flow
  + The computer vision portion of the application should be running 24/7
    - Run the script, with options, if trying to debug, run the record mode option; the system will need to restart the program if shut down while the program is running, or if the global configuration file is modified.
      * Options are: record mode, and test mode <video file>
        + Record mode: Saves a video of the login event with debug information

More information provided later in the document

* + - * + Test mode: Accepts a video file of tools being checked in and out as input, and outputs record information to terminal or file instead of sending it to the database

More information provided later in the document

* + - * Must have the database setup with the records, tools, and drawers tables. Tools and Drawers tables must already be filled in.
        + The Null Terminators will give Hiline a script to assist with taking individual pictures of every tool, the location of each tool within each drawer, and the drawer configuration file.

Look at [5.3 Additional Script](#_sk0sghy8ypwv) under operations for more information.

* + - * If the test mode option is not used, every toolbox monitored by the system must have a valid RTSP camera feed associated with it in the configuration file
  + Employees will use the system by scanning their ID badge to unlock one of the toolboxes (set up in the configuration file and database), then taking tools as normal.
* Context
  + The camera is attached to the box such that the top drawer of the box fills up as much of the frame as possible without “cutting off” any part of the drawer
  + The camera should not move in relation to the box, with the exception of slight shaking from opening and closing the drawers.
  + The program begins to analyze camera footage after the box is unlocked by an employee scanning their RFID badge
  + The program stops analyzing when the “box locked” signal is sent
  + Drawers are different and may require different analysis configurations, so each drawer will use its own configuration file, which should be kept in the database.
* Toolbox requirements:
  + Each drawer must be set up as a shadow box.
    - Each tool must have a unique “home” within a drawer where it is always returned
    - Tools must be put away clean. Grease and grime interferes with the tool identification algorithm
    - Ideally the tool should be a different color from the shadowbox.
      * Ie. If the tool has a black handle do not use a black shadow box background
    - For the fastest and most accurate detection of tools, the shadowbox would ideally not have a notable texture, sheen, or have any other properties that could cause color inconsistency on the background.
      * The shadowbox should also be a different color from the toolbox itself.
        + Ie: The slots where tools reside should ideally not be the same color as the body of the toolbox
      * The classifier should continue to work in suboptimal conditions, but speed and accuracy will be greatly increased if the shadowbox follows the above suggestions.
      * Bad Example:



* + - * Good Example:



* + Each drawer will need to be marked with three instances of a symbol unique to the drawer
    - Three instances ensures the employee's hands won’t cover all instances of the symbol, which would prevent the program from recognizing that a drawer is open.
    - The symbol should be visible to the camera when the drawer is open and not visible at all when the drawer is closed.
  + The database must be updated to reflect any changes in camera position, drawer contents, and/or configuration
  + Each toolbox camera should have a dedicated light source to help with consistency and help eliminate shadows
    - An example of why shadows are detrimental may be found in the appendix under “Problems caused by shadows”
    - If tools appear “washed out” or “grainy”, the program will still attempt to classify tools, although speed and accuracy will be improved with closer to ideal lighting conditions
      * Config file will have a method of disabling drawer segmentation - the process affected most severely by glare and poor lighting. This will give the system the best possible chance at recording tool status.
* Database Requirements:
  + Will be distributed as Docker container
    - Docker must be installed on host server
  + Will run MySQL database server
  + View “[3.1 Database Prototype](#_itgl0qlm3kg5)” under the appendix for database architecture diagram
  + There will only be one database server which communicates with all toolboxes
* API Requirements:
  + API server distributed as Docker container
    - Docker installed on host server
  + Open Source web server will be running in Docker container
  + Supports commands to quickly and easily query information from Database
  + Supports commands to quickly and easily add new records to Database
  + Detailed documentation will be provided, allowing HiLine to write their own reporting solution
  + There will only be one API server which communicates with all toolboxes and reporting software
  + Refer to “[5.2.1 API Specification](#_nijg1twhcj6i)“ for function provide by the API

# 4.0 Environment

*See Appendix entry “*[*1.0 FlowChart*](#_jcf1n2totbq)*” to view a diagram showing a simplified flow chart that presents the program's input and output along with its interaction with other factors.*

## 4.1 Input

* Normal operation
  + Camera footage of tools being checked in and out.
    - Will be retrieved using the RTSP protocol.
    - Requisite information for each toolbox and camera will be stored in the global configuration file.
  + RFID Badge Swipe
    - A port will be listening for a trigger from HiLine’s prox card system
    - The trigger must contain the employee ID and the identifying information of what toolbox is being unlocked.
    - In this iteration of the product, only one RFID login will be handled at a time.
      * If another RFID login is received while an existing login is being processed, the newer login(s) will be ignored
  + Database information
    - The program will retrieve information stored in the tools and drawers tables in the database.
    - Refer to “[3.1 Database Prototype](#_itgl0qlm3kg5)” in the appendix
  + Onnx Tool Recognition Model
    - The program will use a trained neural network in Onnx format to help with tool recognition.
      * Null Terminators will be providing a working Tool Recognition Model (in Onnx format) along with its source code
      * Onnx model will output the tool type when given the image of a specific tool.
      * The global configuration file will contain information to normalize images to be put into the Onnx model.
      * Expected output is an array of numbers with each index corresponding to a class, where the class with the highest number in its index is the predicted object.
      * The tool model has a specific method of how it normalizes images, if a different normalization method is desired, modification of the detection program source code will be needed
        + Variables used in the normalization may be modified using variables contained within the global configuration file
* Additional modes of operation
  + Test Mode
    - Program -test {path/to/file}
    - Users must provide a path to an existing video of tools being checked in and out.
      * The accepted video file formats are MPEG, VCI, and H264.
      * A valid file path must be provided
    - Database information
      * Same as normal operation.
    - Onnx Tool Recognition Model
      * Same as normal operation.
  + Record Mode
    - Program -record
    - No additional input required

## 4.2 Output

* Normal operation
  + Event records
    - Event records will be stored in the Events table within the database.
    - The entry in the CheckedOut column of the Tools table will also be modified if the tool was checked in or out.
* Additional modes of operation
  + Test Mode
    - Event Records
      * A simulation of event records will be output to the terminal.
      * Refer to “[3.2 Test Mode Prototype Output](#_41xdyx7yum4v)” in the appendix for a reference as to what this will look like.
  + Record Mode
    - Event Records
      * No changes to creation of event records
    - Video File
      * Will be saved to the location specified by the global configuration file.
      * Will be in a MPEG format.
      * Records a video from the point of login to the point of logout of the toolbox.
        + Overlayed on the video will be boxes showing what the program thinks are objects or places a tool should be.
        + There will also be text describing what the camera thinks the object is and a number describing how certain the system is.
        + Refer to “[3.3 Record Mode Prototype Output](#_pca5wycvshcm)” in the appendix for a reference as to what this will look like.

# 5.0 Operation

## 5.1 Program operation

*View “*[*1.0 Program Operation Overview Diagram*](#_mrm6viiaz4ua)*” under the Appendix to view a diagram showing a general overview of the program.*

### 5.1.1 Beginning of operation:

* If “record mode” debug variable is set, detection program will begin to save footage showing what it thinks an object is
  + Bounding boxes will be overlaid on the video along with the name of the tool the program thinks it is.
* If “test mode” is set, the program will look for the file. If the file name is invalid, an error will be sent and the program will exit. If it is a valid file, analysis of the footage will be conducted without waiting for any RFID triggers. A text output will be produced showing tools checked in and out, as well as any other records produced, but no information will be sent to the database. The program will then close after output is produced.
* Will read/import the global configuration file
* Will check that links/ locations are valid
* Will wait for the RFID trigger unless it is in test mode

### 5.1.2 During operation:

* Receive RFID trigger
  + RFID trigger must contain employee ID and which toolbox the employee is logging into
  + It will be assumed that the RFID trigger is real-time so the timestamp will be the system time.
* Get footage
  + Access footage with RTSP
  + Stop retrieving footage when the logout trigger has been received.
  + Retrieve drawer information for the box from the database
* For each frame:
  + - Check for drawer
      * If the drawer is open for the first frame, note the drawer opening
      * If the drawer was open and is now closed, note the drawer closed and create tool records for the database by inserting the current status of each tool
      * To achieve best possible tool recognition, only one drawer will be processed at a time
      * The system will recognize the drawer using the image of the symbol saved in the database along with template matching or a similar computer vision technology.
      * Read in tools that are in the located drawer using the location information in the database.
    - Check which tools should be viewable
      * Tools considered viewable will include tools that are partially visible to the camera.
        + The amount that needs to be seen will vary by tool, with the minimum required height (given in pixels) described by the drawer configuration document.
        + Partially visible means the toolbox drawer would need to be pulled out more to see it, and does not refer to a tool obscured by someone's body or another object.

If an object is obscured, the program will assume that the tool’s status has not changed, since the last frame it was visible.

If the hand or obstruction is moved, the system will then process the unknown tool and determine whether or not it is present

* + - Analyze whether tools are there
      * Drawer will be segmented using either the stored pixel location or the segmentation algorithm.
        + Segmentation will always be used to check to see if there are extra tools, or tools not in any tool location in the drawer
        + There will be an option to turn off the search for extra tools in the drawer configuration file.
      * Template matching will be used with both tool viewable and tool missing templates to determine if there is a tool in a particular location within the drawer.
        + Will have a minimum similarity threshold to check if there are objects blocking the tool from view.
        + To facilitate this, two pictures of each tool location must be stored in the database, one with the tool and one without the tool.
      * After template matching and segmentation is complete, tools will be evaluated for the presence of any unique identifying marks which would distinguish them as a particular tool.

This helps ensure similar tools within a drawer are returned to their correct home

Identifying by unique feature should continue to work even in poor lighting, although speed and accuracy may be reduced

* + - * Program will use a combination of the onnx tool model, similarity thresholds to the templates and stored size/location data to see if an item is the correct tool.
        + If it is not the correct tool it will be deemed a misplaced tool.
      * Program will wait until the drawer is closed to create any records in the database, and will create records from the last frame in which the tool was visible.
        + This is to ensure that there are not a bunch of records of someone picking up a tool and then putting it back down in the same login session.

### 5.1.3 After operation:

* Will save records, and update the tool status in the database as well as who checked it out ( if anyone checked it out), otherwise will go back to waiting as long as test mode is not set
* If test mode is set, the program will end and print the records as output.
  + Reference the [3.2 Test Mode Prototype Output](#_41xdyx7yum4v) for how the output will look like.
* If record mode is set, footage will be saved to a location specified in the configuration file. If also in test mode, the program will exit, otherwise it will wait for the next RFID login event.
  + The titles of the video files will contain the timestamp of when the employee logged in.
  + Reference the [3.3 Record Mode Prototype Output](#_pca5wycvshcm) for how the output will look like.

## 5.2 Database Operation

### 5.2.1 API Specification

(Pseudocode outlining which queries will be made to the database.)

*(Note: See design document for full list of parameters and descriptions.)*

**getToolsInfo**() { }

**getDrawersInfo**(){}

**addEvent**(){ }

**addTool**(){}

**addDrawer**(){}

**updateToolsInfo**(){}

* getToolsInfo will return all tool entries from the specified toolDrawerID.
* getDrawers will return all drawers from the specified drawerBoxNum.
* addEvent will add an event record into the events table.
* addTool will add a tool to the tools table.
* addDrawer will add a drawer to the drawers tables.
* updateToolsInfo will update the ‘toolCheckedOut’ status of a tool.

## 5.3 Additional Script

* This script is separate from the program, and will be used by the Null terminators to facilitate testing.
  + This will help HiLine determine if a drawer will be easily analyzable by the program, and will assist with gathering the information necessary for the database.
* A brief document explaining how to use the script will also be provided.
* If this script is unable to segment the drawer without issues, this is an indication the system will not be able to differentiate between tools within the current drawer configuration
* Issues identified by this script should not be considered fatal, as the secondary classifier focusing on unique symbols may still be able to identify a tool’s status even if the primary classifier has failed
  + Tools with variations in color (handles, twists causing shadows, etc.) may not be differentiable
  + Tools may be too close together, or otherwise obscured from the camera
  + The lower the amount of error in the segmentation, the more accurate the tool detection.
  + Further explanation and reference photographs will be provided in documentation
* Operator will provide two pictures as input to the script: one of the empty drawer with no tools, and one where the drawer is full of tools.
* It is ***highly*** recommended that these pictures come from the same camera used for computer vision with the drawer and camera in the same position as they will be during actual use.
  + If the pictures are not from the same camera and in the same position, the output of the script will not have the correct values for the database.
  + Initial reference pictures must be free of shadows, hands, or other obstructions

# Appendix

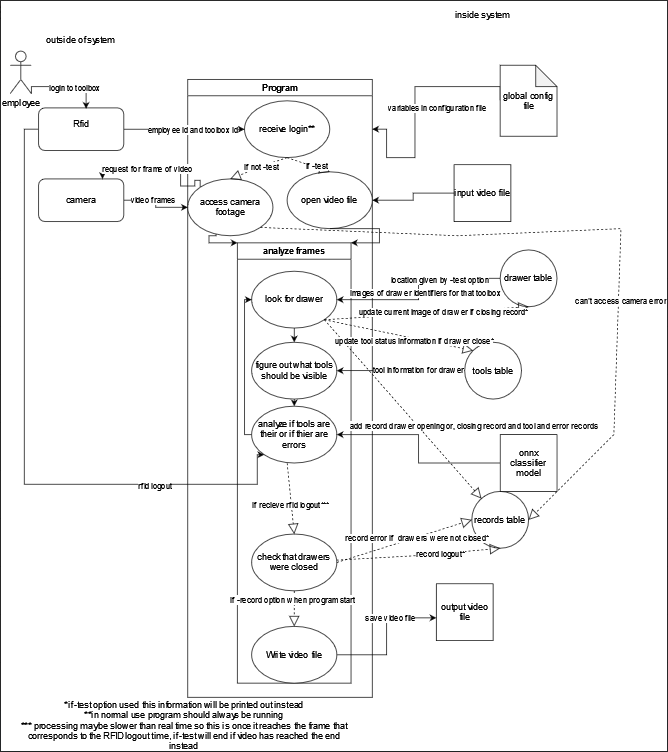
## 1.0 FlowChart

## A diagram of a software application Description automatically generated with medium confidence

(Zoom in on browser to inspect)

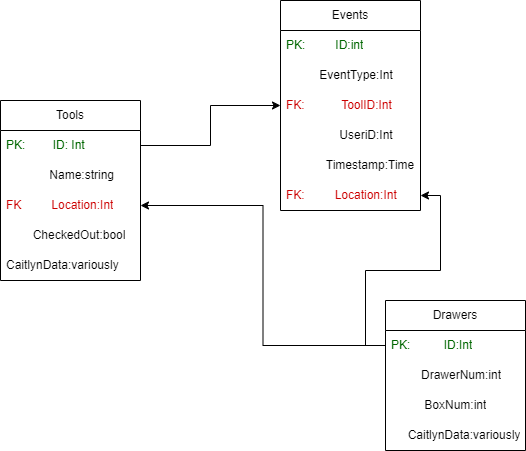
## 2.0 Program Operation Overview Diagram

## *Refer to the design document for up to date program operation*



## 3.0 Prototyping

### 3.1 Database Prototype



The figure above shows the proposed layout of tables and data fields within the database - typically referred to as a *schema*

* 3 primary tables - Tools, Events, and Drawers
* Tools table
  + ID - unique integer value identifying a particular tool, Called the *primary key*
  + Name - The human readable name of the tools, stored as a string
  + Location - an int value identifying a particular drawer within a particular toolbox. This value is called a *foreign key,* as this is the primary key ID of a drawer entry in the drawers table
  + CheckedOut - called a *bool*, this value can either be true or false, reflecting whether or not the tool is currently in the drawer
  + CaitlynData - a placeholder type, there will be some additional data like pictures of the drawers, and similar information. More experimentation is required to determine the exact layout of this field, but HiLine should expect to see an amount of data stored here in support of the camera system
* Events table
  + ID - unique integer value identifying a particular event record. The *primary key* of this table
  + EventType - an integer value identifying what information is conveyed by the record. Is the tool being checked out? Returned? Was a drawer opened? Closed? Etc. A map of these types will be provided in the documentation as well as in the API utility
  + ToolID - int value, a foreign key listing which tool from the tools table is being manipulated by the record
  + UserID - int or alphanumeric value - the unique employee ID provided by the API triggers given from HiLine’s authentication and prox card system
  + Timestamp - the time at which the event occurred
  + Location - int value, a foreign key from the drawers database which identifies which drawer and in which box an event occurred in
* Drawers Table
  + ID - unique integer value identifying a particular drawer and toolbox combination. The primary key of this table
  + DrawerNumber - integer value identifying the number assigned to a manipulated drawer
  + BoxNum - integer value identifying the toolbox being manipulated
    - Collapsing both of these numbers into one ID field allows for faster searches for all tools or activity in a particular drawer, as only one column must be searched in the tools or events database.
  + CaitlynData - a placeholder type, there will be some additional data like pictures of the drawers, and similar information. More experimentation is required to determine the exact layout of this field, but HiLine should expect to see an amount of data stored here in support of the camera system

### 3.2 Test Mode Prototype Output

Outputted records will be in chronological order.

Opened: Toolbox <toolbox identifier> Drawer <toolbox identifier>: <time> <employee id>

Tools checked out:

<Tool identifier> <employee id> <time> <location>

. . . (repeat for each tool checked out

Tools checked in:

<Tool identifier> <employee id> <time> <location>

. . . (repeat for each tool checkout)

Error:

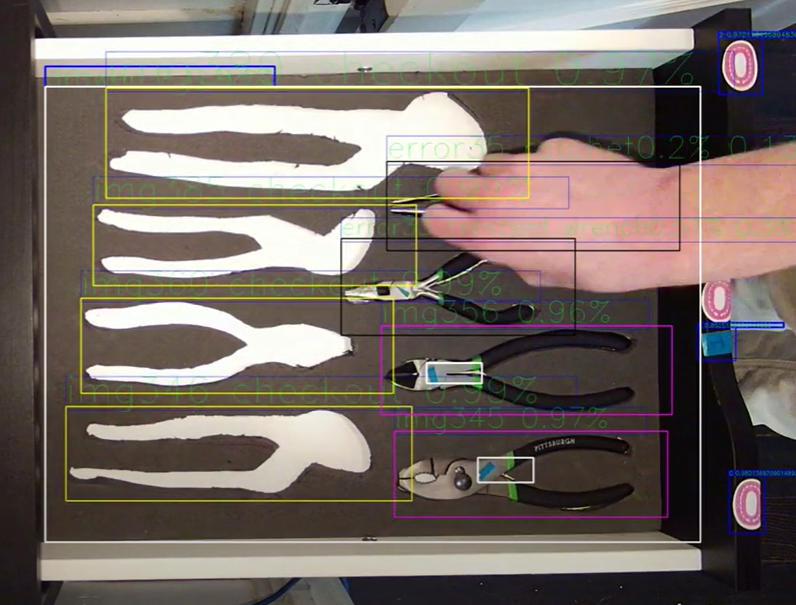
<error type> <tool identifier> <employee id> <time> <location>

. . . (repeat for each tool checkout)

Closed: Toolbox <toolbox identifier> Drawer <toolbox identifier>: <time> <employee id>

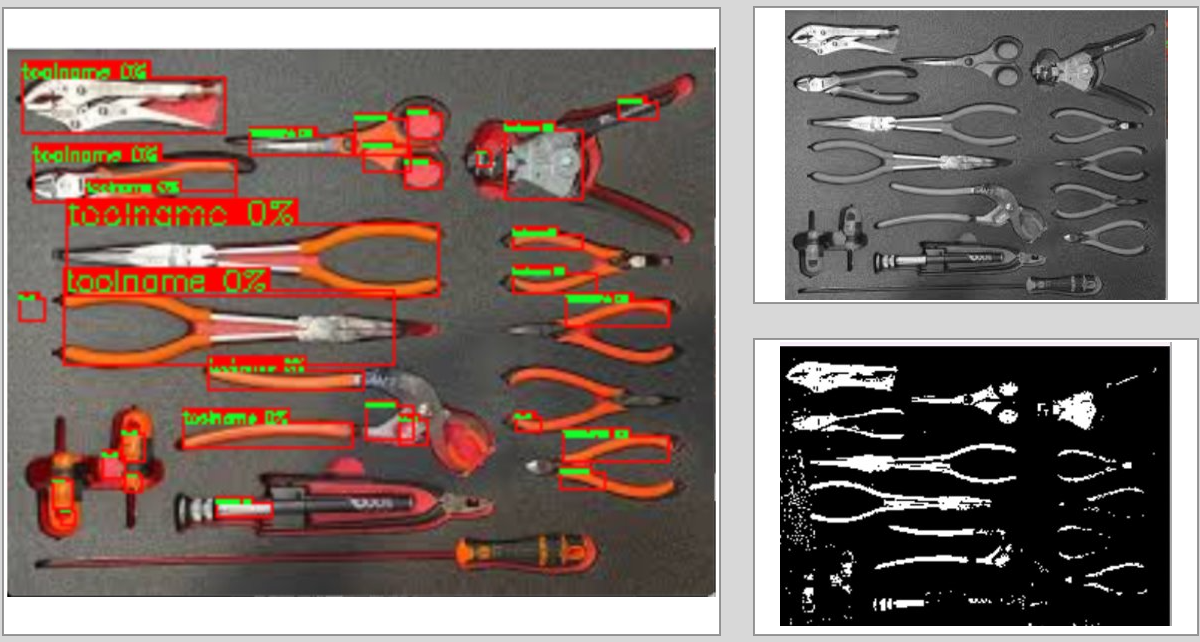
… (repeat for each opening of drawer)

### 3.3 Record Mode Prototype Output



A Magenta box means tool is checked in, a yellow box means tool is checked out, black means error - wrong tool.The large white box shows the outline of the recognized drawer and the smaller white boxes indicate drawer symbols. Blue boxes outline both drawer symbols and extra tools. If there is no tool in a tool location, the label will say “checkout”. If there is an error the label will say error and then the tool ID number.

### 3.4 Problems caused by shadows



The above photos show an example of a drawer that is difficult to process effectively. The glare in the top left corner, as well as the shadow in the bottom right corner adds complexity to the detection process, and the primary method of determining the presence of a tool may not be effective. The tools with black handles overlayed on the black background also pose issues for the traditional method. Matching tools by their engraved unique character should continue to function, but detection speed and accuracy will be greatly improved by more consistent lighting.

## 4.0 Configuration files

### 4.1 Variables in the global configuration file:

* Rtsp link for each toolbox
* File location of onnx classifier
* Port for RFID trigger listener
* Image file normalization variables for classifier
* Database location/access information
* Location to save video files when using record mode option
* Similarity threshold for template matching (decides whether or not someone's hand or head is in the way)
* Location +- in drawer int value for the allowable difference from stored location values
* Size +- in drawer int value for the allowable size difference from stored width and height values
* More variables may be added

### 4.2 Variables in the drawer configuration file (stored in the database):

* Cv2 enum int value to turn picture into grayscale
* Cv2 enum int value for threshold type
* Cv2 enum value for finding contours
* Int value for the threshold
* Int value for Minimum width of object in pixels
  + Should at least be 1
* Int value for Minimum height of object in pixels
  + Should be at least 1
* Int value for which picture tool location in drawer information comes from
  + 0 = no tools image
  + 1= tools image
  + -1 = neither ( location in the picture was found manually, so do not segment drawer)
  + 2 = both (ie going through the algorithm every segmentation in both match up)
* More variables may be added

## 5.0 Camera research

* Minimum camera requirements
  + Illumination
    - 1 lux
      * Will probably still want a light specifically for the camera for consistency.
    - It should also be noted that all the cameras the team found were true wide-range which means that they can capture a larger spectrum of lighting conditions.
  + Lens
    - Field of view
      * The vast majority of cameras should be fine, it really just determines how far away the camera needs to be, the wider the viewing angle the closer it can be. This means the less likely it is for someone to have their head in the way the entire time. So less than 130 more than 80.
  + Resolution
    - 2 mp
      * The more resolution typically the better, but in the end looking at the license plate pictures, the bottom drawer should be about 5 times the width and 4.8 times the height times the size of that license plate using the given measurements, with the plate being about 100 x 50 pixels on a 2 MP camera which times 5 and 4.8 would be 120,000 pixels. An 8 mp camera in comparison will be about 200 x 100 so about 480,000. A smaller viewing angle makes it a little bit wider, but we will also have to move it farther away from the box. So an 8 mp is unmistakably better, but a 2mp camera should still theoretically work.
      * 130\* 49/(120,000) =.053
      * It should also be noted that for debugging an 8mp or greater camera is desired since the minimum text thickness and bounding boxes are 1 pixel.
  + Protocols
    - RTSP
  + Other
    - The camera should not automatically zoom or move. If a camera has these features, it must have the ability to turn the features off.
    - To help facilitate the digital image recognition process, small objects should be placed in drawers near the top of the toolbox when possible. This allows the computer vision system to have the best possible chance at recognizing the tool quickly and accurately.