horizontal line

**CSML1030: Machine Learning Capstone**

Sheep Transportation Tracking

Final Report for Canadian Sheep Federation (CSF)

**18th May 2025**

| ***Author:***  Matin Mazid |  |
| --- | --- |
| ***Author:***  Daniel (Jee) Lee | ***Professor:***  Saber Abini |

This work aims to discover possible approaches to improving data processing accuracy by applying machine learning methodologies to handwritten forms. A step-by-step analysis will be conducted to identify patterns and inconsistencies, with a focus on ensuring accurate parsing and interpretation of the scanned form.

# I. INTRODUCTION

## PROJECT BACKGROUND

**Canadian Sheep Federation (CSF -** [**About CSF | cansheep**](https://www.cansheep.ca/about-csf)**)** helps farmers navigate the sheep industries, value chain. It uses handwritten forms submitted to CSF to track the movement of sheep. The forms contain large amounts of data, often encountering inconsistencies like how dates and time are formatted across different submissions. The lack of a standardized format can lead to errors in data extraction and processing.

For example, dates may appear in various formats, such as:

* **MM/DD/YYYY** (e.g., 03/10/2024)
* **DD-MM-YYYY** (e.g., 10-03-2024)
* **Month DD, YYYY** (e.g., March 10, 2024)

Similarly, times may appear in these formats:

* **24-hour format (Military Time):** Uses hours from 00 to 23, with no AM/PM.
* **12-hour format:** Uses hours from 1 to 12, with AM and PM.

These discrepancies pose problems when trying to standardize and extract accurate information from handwritten records. This creates a challenge for CSF, where misinterpreted dates could lead to inaccurate records.

In addition, the form allows submitters to circle values from amongst a set of values. This means we need to be able to identify selected circled text as the intended text.

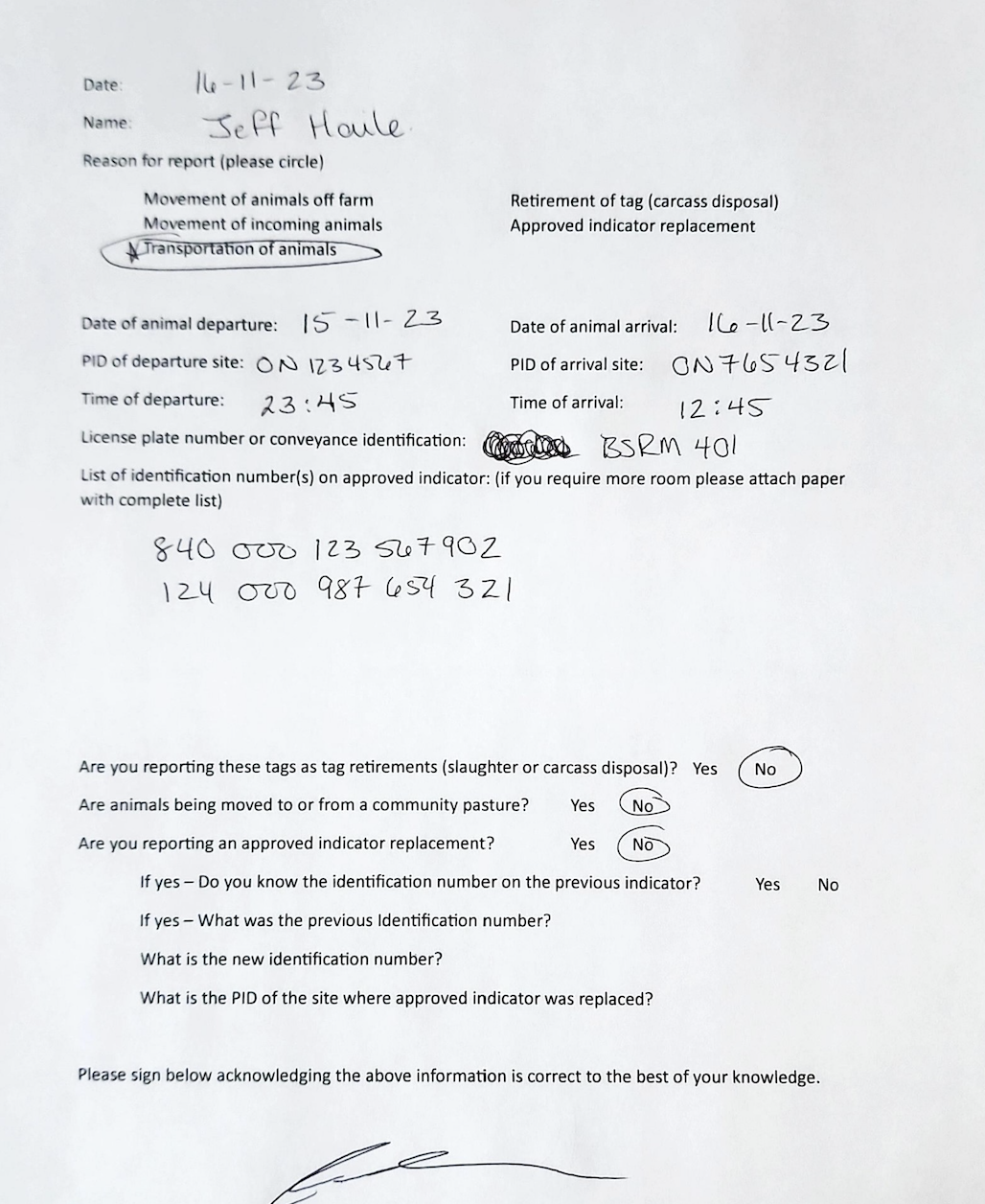
## OBJECTIVE

The objective of this project is to assist Can-sheep in automating livestock record processing by addressing challenges with unstructured data interpretation from scanned paper forms. We aim to implement a dynamic date format detection algorithm that analyzes contextual relationships—such as between departure and arrival dates—to accurately parse various date formats without depending on fixed assumptions. This will reduce data entry errors and enhance reliability in digitizing transportation records.

Additionally, the project will utilize image classification tools to detect circled responses, enabling automated interpretation of multiple-choice answers. These innovations will improve efficiency in the data entry workflow while supporting the continued use of paper forms, which are necessary due to limited connectivity in rural areas. By combining intelligent date parsing and image recognition, the proposed solution ensures compatibility with Can-sheep’s existing practices and significantly reduces the need for manual input in their livestock tracing system.

# II. EXPLORATORY DATA ANALYSIS

Image scans of the forms signed by a farmer are inputted as the data. A sample form is shown below:



## SPECIAL CONSIDERATION

There will always be handwriting variability, as individuals have different writing styles and the quality of handwriting can vary greatly.

* **Legibility Issues**: Handwritten forms can be difficult to read, with varying handwriting styles and quality of writing (e.g., smudging, inconsistent characters). The algorithm might need to handle these variations effectively.
* **Character Recognition Errors**: OCR tools may misinterpret characters or digits (e.g., "1" vs "I" or "0" vs "O"), which can lead to errors in date parsing. Incorporating error-checking mechanisms could help improve the system.

# III. PROJECT OVERVIEW AND APPROACH

## GOALS

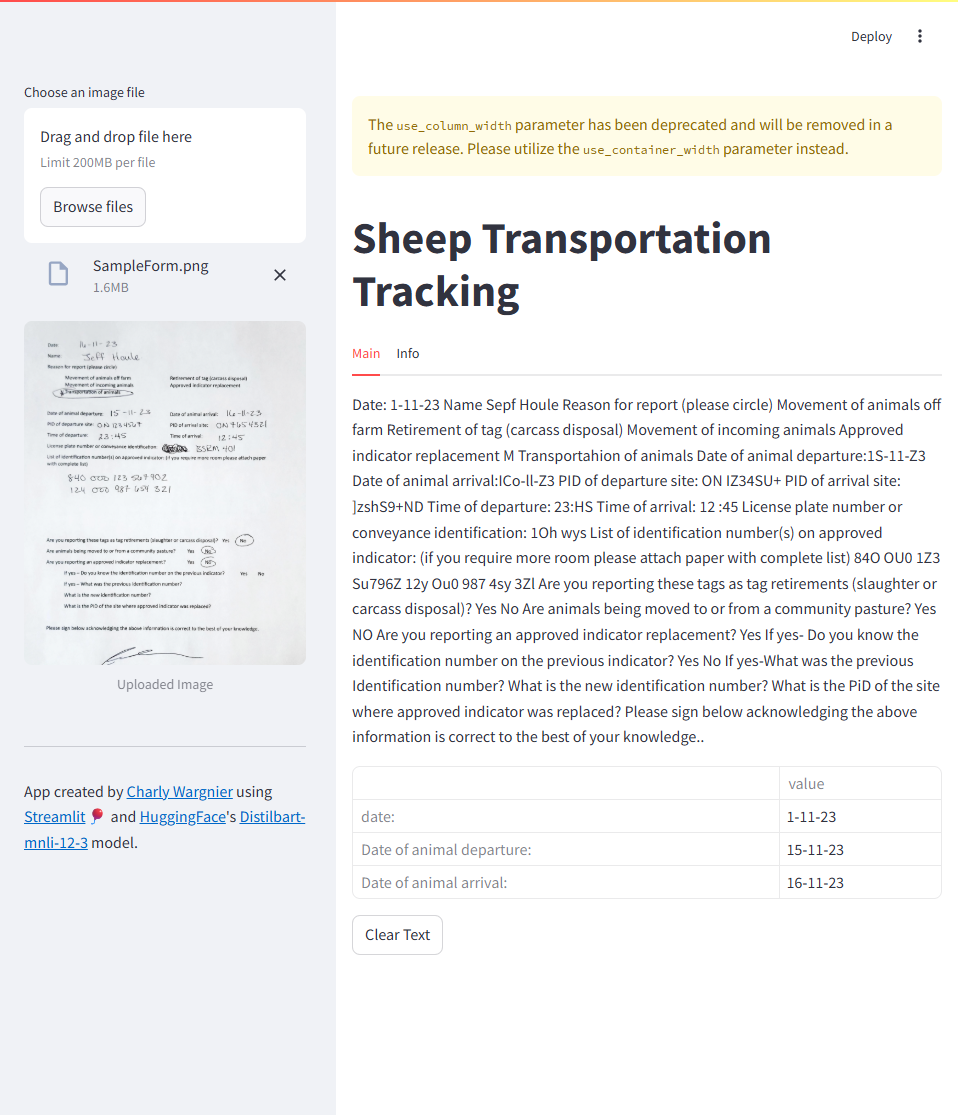
1. Populate and store the information found in the form.
2. Provide a heuristic to guess the date format so we can increase the operational accuracy of the information in the current CSF data store.
3. Have a final application that extracts handwritten data and stores it in the database.

## METHODOLOGY EXPLORED

Throughout the development of our project, we explored several methodologies for text extraction, including widely used tools like **Pytesseract** and **PaddleOCR**. While these tools offered quick implementation and broad functionality, we encountered significant challenges with accuracy.

### INTERIM RESULTS

The interim results proved unsatisfactory using local image recognition tools. As you can see, we uploaded the sample form (refer to the image above), some crucial values, like the date, were omitted in the output, leading to incomplete data. Interpreting such corrupted or partial data can lead to misleading or unreliable outcomes. These challenges prompted us to look for alternative approaches to ensure the reliability and integrity of the extracted information.



## FINAL METHODOLOGY

## Methodology Highlights

### IMAGE INTERPRETATION

We considered the following strategies:

* [Google Cloud Compute](https://cloud.google.com/document-ai/pricing)
* [Amazon Web Services (AWS)](https://aws.amazon.com/textract/pricing/)
* ChatGPT

The requirement to host additional infrastructure in the Google Cloud Compute case and AWS case, Led us to start with ChatGPT.

ChatGPT allowed us to combine business logic, parsing, data standardization and image OCR (Optical Character Recognition) into one step. To do do this we gave ChatGPT be below prompt:

*“You are an excellent data entry specialist that is really good at understanding handwritten data, extracting the text from the image and returning it in a JSON format. Change all the dates to the format YYYY-MM-DD. Assume that all the dates are three within three weeks of {the current date} and animal departure is always before animal arrival.”*

The use of a provided current data and bounding possible dates to within a reasonable time frame increased the accuracy of the interpreted dates without resulting in extra coding for developers.

To send the data via Rest API to ChatGPT, we had to base-64 encode the image and specify the content type in the request. This allowed us to refer to the image in the context window of the chat api, without counting it towards the token count of the request. The entire spend for the development of this functionality was approximately $3.62 USD [3]

This returned process returned accurate results and given the relative accuracy of the results we decided to submit this solution for evaluation.

### HUMAN INTERFACE

For the purposes of this application, a classic web based interface was the most appropriate. The client can use web browsers to upload images and also to inspect the results of the image analysis before saving it to their data store. The use of a web server and web client establishes a well known technical pattern with many mature technologies to choose from. For this project we chose to use streamlit as it helps to constrain the technical stack to python while still allowing for a rich user experience.

## HUMAN IN THE LOOP

Also, the application should allow staff members to review interpreted image data. This would allow us to correct poorly interpreted data.

With respect to the work done for the purposes of this exploration, we have included the following functionality

1. Upload functionality: Users can upload a .png formatted image and view it on a web browser.
2. Automatic Image interpretation: Users do not need to do any further action, the image will automatically be interpreted and the form will be parsed to show the extracted data.
3. Saving the data to mongo. The user can elect to save the data to their data store, in this case mongo.
4. Resetting the image without saving: The user can clear the form without saving if the interpretation is not accurate enough to save.

With further development we can allow users to correct the automatic image interpretation.

## TEST DATA

We used mostly scanned data at a low resolution black and white scans of a sample form that we extracted based on the test data. To properly test the system’s ability to interpret different handwriting styles we recruited family members to fill in the form with the expressed instruction to use different date formats than numeric formats such as YYYY/MM/DD. We also created test forms that included spelling mistakes and inconsistent date formats.

The test cases used are attached [here](https://www.dropbox.com/scl/fi/6dx37hute11tipyq1l104/Testcases.zip?rlkey=1ghvma1fldgzk4wrlf7lusrc7&st=uc69m1wk&dl=0).

# IV. PROJECT SETUP INSTRUCTIONS

## GIT REPOSITORY

The *CanSheep Transportation Tracking* application is open-source and available on GitHub. You can find the repository here:<https://github.com/matinmazid/capstone>. Follow the steps below to deploy and run the project locally or online.

## DEMPLOYMENT

The *CanSheep Transportation Tracking* application was developed using **Streamlit**. To deploy and run the application, follow these steps:

### LOCAL DEPLOYMENT

### **1. Clone the Repository**

Start by cloning the project from GitHub:

| git clone https://github.com/matinmazid/capstone.git cd capstone/can-sheep-project |
| --- |

### **2. Requirements**

Ensure you have the required packages installed by running:

| pip install -r requirements.txt |
| --- |

Python 3.8+ is recommended.

### **3. Secret Keys & Environment Variables**

The application requires several sensitive credentials:

* OpenAI API Key
* MongoDB Username and Password

To keep these secure, store them in a .env file in the project root:

| OPENAI\_API\_KEY=your\_openai\_api\_key MONGODB\_USERNAME=your\_mongodb\_username MONGODB\_PASSWORD=your\_mongodb\_password |
| --- |

#### **4. Running the Application**

To launch the app locally:

| streamlit **run** app.py |
| --- |

### REMOTE DEPLOYMENT

To deploy on [Streamlit Community Cloud](https://streamlit.io/cloud):

1. Create a fork to [GitHub repo](https://github.com/matinmazid/capstone/fork).
2. Create a new app on Streamlit Cloud.
3. Under *Settings > Secrets*, input your secret values:

| OPENAI\_API\_KEY="your\_openai\_api\_key" MONGODB\_USERNAME="your\_mongodb\_username" MONGODB\_PASSWORD="your\_mongodb\_password" |
| --- |

# V. SUMMARY

Interpreting handwritten values is challenging, especially without the corrective capabilities of a large language model (LLM). Even when handwriting is accurately transcribed into text, issues such as inconsistent or non-standard date formats, spelling errors, and ambiguous intent remain difficult to resolve.

By leveraging a language model, we can simplify much of this complexity—using the model to generalize and interpret the data as a whole, rather than relying solely on exact character recognition. Additionally, integrating ChatGPT as a service, rather than hosting a custom model on Amazon Web Services (AWS) or Google Cloud, helps streamline our infrastructure and reduce operational overhead.

# REFERENCE

[1] **Rosebrock, A.** (2018, September 17). *OpenCV OCR and text recognition with Tesseract*. PyImageSearch. [https://pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract](https://pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/)

[2]https://chatgpt.com/share/682a5c2a-be84-800a-be2f-d92514d2aa1a

[3] screenshot of actual spend

