

**A REPORT
ON
AADHAR MASKING AND OCR BASED
TEXT EXTRACTION**

Submitted by,
Mr. ALLEN CONROY DSOUZA – 20211CSE0227

Under the guidance of,
Dr. M CHANDRA SEKHAR
in partial fulfillment for the award of the degree of
BACHELOR OF TECHNOLOGY

**IN
COMPUTER SCIENCE AND ENGINEERING**

At

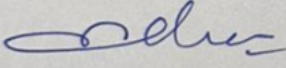


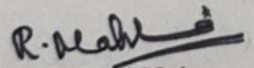
**PRESIDENCY UNIVERSITY
BENGALURU
MAY 2025**

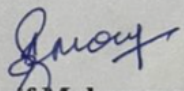
PRESIDENCY UNIVERSITY
PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND
ENGINEERING

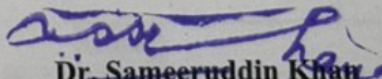
CERTIFICATE

This is to certify that the Internship/Project report “AADHAR MASKING AND OCR BASED TEXT EXTRACTION” being submitted by “Mr. Allen Conroy Dsouza” bearing roll number “20211CSE0227” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.


Dr. M Chandra Sekhar
Professor
PSCS
Presidency University


Dr. Mydhili Nair
Associate Dean
PSCS
Presidency University


Dr. Asif Mohammed
HOD, CSE
PSCS
Presidency University


Dr. Sameeruddin Khan
Pro-Vice Chancellor -
Engineering
Dean –PSCS / PSIS
Presidency University


PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled “**AADHAR MASKING AND OCR BASED EXTRACTION**” in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering**, is a record of my own investigations carried under the guidance of **Dr. M Chandra Sekhar, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

Name	Allen Gmroy D'souza
Roll No	2021CSE0227
Signature	

ABSTRACT

This internship report presents the work conducted during my tenure at Abylle Solutions, focused on developing an automated Aadhar card number masking system using Optical Character Recognition (OCR) and image processing techniques. The core objective was to ensure privacy compliance by detecting and masking sensitive Aadhar digits in scanned or photographed documents. Leveraging Python-based tools such as Tesseract OCR and OpenCV, I engineered a pipeline to extract text from images, accurately identify Aadhar numbers using regular expressions, and apply masking algorithms to conceal the digits. The project involved a hands-on understanding of text extraction challenges from diverse image formats, handling noise, orientation corrections, and ensuring consistent masking across varying layouts. Through iterative testing and optimization, the model achieved robust accuracy in detecting and anonymizing Aadhar numbers, thereby contributing to secure data handling in line with data protection protocols. This experience enhanced my skills in image preprocessing, text recognition, and real-world deployment of data privacy solutions using computer vision.

ACKNOWLEDGEMENTS

Firstly, I am indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

I express my sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC – Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting me permission to undergo the Internship. I express my heartfelt gratitude to our beloved Associate Dean Dr. Mydhili Nair, Presidency School of Computer Science and Engineering, Presidency University, and Dr. Asif Mohammed, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in this internship successfully.

I am greatly indebted to our guide **Dr. M Chandra Sekhar, Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his inspirational guidance, and valuable suggestions and for providing me with a chance to express my technical capabilities in every respect for the completion of the internship work.

I would like to convey my gratitude and heartfelt thanks to the CSE7301 Internship Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K**, Department Project Coordinator **Mr. Jerrin Joe Francis** and Git hub coordinator **Mr. Muthuraj**. I thank my family and friends for the strong support and inspiration they have provided me in this internship.

Allen Conroy Dsouza

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	I
1.	INTRODUCTION	1
2.	WEEK 1 & 2: ORIENTATION AND REQUIREMENT ANALYSIS	2
3.	WEEK 3: RESEARCH ON OCR AND PRIVACY REGULATIONS	4
4.	WEEK 4 & 5: TECHNOLOGY STACK FINALISATION AND DATA SET PREPERATION	6
5	WEEK 6: AADHAR PATTERN RECOGNITION USING REGEX AND PREPROCESSING	8
6.	WEEK 7 & 8: OCR IMPLEMENTATION AND TESTING WITH TESSERACT	10
7.	WEEK 9: IMAGE MASKING AND ANONYMIZATION PIPELINE	12
8.	WEEK 10: ERROR HANDLING AND EDGE CASE VALIDATION	14
9.	WEEK 11: INTEGRATION, LOGGING AND USER TESTING	16
10.	WEEK 12: FINAL DEPLOYMENT, DOCUMENTATION AND HANDOVER	18
11.	TIMELINE AND WORK BREAKDOWN	20
12.	OUTCOMES	21
13.	CONCLUSION	22

Chapter 1

INTRODUCTION

During my three-month internship at **Abylle Solutions**, I worked on building a privacy-focused automation tool aimed at detecting and masking Aadhar numbers in digital documents. With increasing concerns about data privacy and regulatory compliance, especially surrounding personally identifiable information (PII), the project was designed to ensure that documents containing Aadhar numbers could be processed and anonymized efficiently.

The core objective was to build a robust system capable of handling diverse document formats (images, PDFs), detecting Aadhar numbers using advanced deep learning and text recognition techniques, and masking them automatically. The solution combined **YOLO (You Only Look Once)** — a real-time object detection model — with **Tesseract OCR** to achieve high accuracy in both identifying and verifying sensitive numeric patterns.

The YOLO model was used to detect candidate regions likely containing Aadhar numbers based on visual patterns, independent of textual recognition. Once these regions were identified, **Tesseract OCR** was applied to extract and validate the text using regular expressions. This dual-stage architecture allowed the system to operate with greater resilience across varied layouts, fonts, image noise, and rotation issues common in real-world document scans.

The project also involved:

- Developing a command-line interface for batch automation.
- Creating a user-friendly graphical interface using **Tkinter**.
- Implementing detailed logging for traceability.
- Preparing the system for internal deployment through modular coding and documentation.

Through this internship, I gained hands-on experience in deep learning integration, computer vision, OCR pipelines, image preprocessing, and privacy-aware software development. I also learned how to align technical design with real-world compliance standards and usability requirements.

Chapter 2

Week 1 & 2

Orientation and Requirement Analysis

The initial two weeks of my internship at **Abylle Solutions** were focused on understanding the project scope, use-case constraints, and privacy implications of processing sensitive identity documents such as Aadhar cards. I was introduced to the organization's data protection objectives, software development practices, and the regulatory backdrop driving the need for automated Aadhar number masking.

Project Goal

The primary objective was to build a tool capable of automatically detecting and redacting Aadhar numbers from document images and PDFs. This was to be achieved using a hybrid approach:

- **YOLO (You Only Look Once)** for detecting regions in documents likely to contain Aadhar numbers based on visual cues.
- **Tesseract OCR** to extract and verify text from those regions using regular expressions.
- **OpenCV** to perform the actual masking of detected Aadhar numbers by overlaying black rectangles or obscuring the text.

Functional Requirements

- Detect Aadhar numbers in various formats (e.g., 1234 5678 9012, 1234-5678-9012, 123456789012).
- Accurately locate and mask Aadhar number regions across:
 - Scanned Aadhar cards
 - KYC documents
 - Multi-page PDFs
- Provide both **CLI-based** and **GUI-based** tools for different user profiles.
- Ensure modularity so that components (detection, OCR, masking) can be improved or replaced independently.

Non-Functional Requirements

- High accuracy and low false-positive rate in both detection and masking.
- Robust performance on rotated, blurred, or low-resolution documents.
- Fast execution on standard desktop hardware.
- Maintain an audit trail via logging.

Technical Orientation

- Understood how **YOLOv8** architectures can detect structured information like ID numbers based on labelled training data.
- Studied OCR systems, especially **Tesseract**, for text extraction from image regions.
- Explored OpenCV's capabilities in drawing masks, handling image formats, and supporting preprocessing pipelines.
- Reviewed Python libraries for PDF manipulation, image conversion, and UI creation (e.g., `pdf2image`, `Tkinter`).

Output Expectations

The final deliverables would include:

- A functional Python-based masking tool.
- Integration of YOLO and OCR-based detection techniques.
- A basic Tkinter GUI for ease of use.
- Documentation, test cases, and a logging system for transparency and traceability.

This foundational phase set the direction for the architecture and modular design of the Aadhar masking tool. The next step was to perform deep research into OCR tools, YOLO model training/inference, and legal constraints surrounding Aadhar handling.

Chapter 3

Week 3

Research on OCR and Privacy Regulations

During Week 3, I conducted in-depth research on the technical foundations and regulatory requirements necessary for building a privacy-compliant Aadhar masking system. The focus was to understand the roles of **OCR (Optical Character Recognition)** and **YOLO (You Only Look Once)** in a document processing context and to ensure the project aligned with Indian data protection guidelines.

Understanding OCR (Tesseract)

- Tesseract is an open-source OCR engine capable of recognizing printed text in multiple languages.
- I explored Tesseract's `image_to_string()` and `image_to_data()` methods for full-text extraction and bounding box-level metadata.
- Key observations:
 - Performance is highly sensitive to image quality.
 - OCR works better when combined with preprocessing like binarization, noise reduction, and skew correction.
 - Without prior localization (e.g., bounding boxes), OCR may extract a large volume of noisy or irrelevant text.

Understanding YOLO (Object Detection for Aadhar Regions)

- YOLO is a real-time object detection model that detects objects within images based on bounding box regression and class probability.
- In this project, YOLO was used to detect **regions of interest (ROIs)** likely to contain Aadhar numbers.
- Benefits of YOLO:
 - It allows region-specific OCR, significantly improving accuracy and reducing noise.
 - Faster and more scalable than exhaustive text-based detection in large images or multi-page PDFs.
- I reviewed examples and documentation for YOLOv5 and YOLOv7, focusing on:
 - Model inference using pre-trained weights.
 - Custom dataset creation for detecting Aadhar number formats.
 - Bounding box output format (class, confidence, x/y coordinates).

Hybrid Architecture: YOLO + OCR + Regex

The architecture established during this phase was:

1. **YOLO** detects bounding boxes around Aadhar number candidates.
2. **OCR** extracts text from those boxes.

3. **Regex** filters out non-Aadhar patterns and validates true matches using patterns like:

```
r"\b\d{4}[\s\-\]?\d{4}[\s\-\]?\d{4}\b"
```

This three-layer system allowed for precise, high-confidence detection and masking.

Privacy Compliance and Regulations

- Reviewed the **UIDAI Guidelines** and relevant clauses from the **Information Technology Act (India)** concerning the storage and sharing of Aadhar numbers.
- Aadhar is classified as **sensitive personal information (SPI)** under Indian law.
- Key points:
 - Aadhar numbers must be masked or redacted unless explicitly required by law.
 - Systems must avoid storing unmasked numbers in logs or outputs.
 - An audit trail is essential for tracking data masking operations.

Conclusions

- YOLO and OCR serve complementary roles—YOLO narrows down the search space, and OCR verifies and interprets text within those regions.
- Regular expressions act as the final validation layer to reduce false positives.
- Regulatory research helped shape the masking strategy and logging policies.

With these insights, I began building the initial components of the YOLO and OCR integration pipeline in the following weeks.

Chapter 4

WEEK 4 & 5

TECHNOLOGY STACK FINALISATION AND DATASET PREPERATION

During Weeks 4 and 5, I finalized the tools and frameworks required for building the Aadhar number masking system and integrated a pretrained YOLO model into the processing pipeline. The focus shifted from data collection (which was not feasible due to privacy concerns) to setting up a reliable inference mechanism using publicly available models.

Finalized Technology Stack

Based on the requirement to process documents without storing sensitive user data or manually annotating Aadhar card images, I opted for a completely prebuilt, inference-based solution using the following tools:

- **Python 3.x** – Primary scripting language for building the system.
- **YOLOv8 (Nano)** – A pretrained object detection model specifically designed to detect Aadhar numbers, downloaded from:
 - <https://huggingface.co/arnabdhar/YOLOv8-nano-aadhar-card>
- **Ultralytics YOLOv8 framework** – Used for loading the pretrained model and running inference on images.
- **Tesseract OCR (pytesseract)** – Applied to YOLO-detected bounding boxes to extract and verify text using regular expressions.
- **OpenCV** – Used for image preprocessing, masking, and drawing bounding boxes.
- **Pdf2image** – Enabled PDF-to-image conversion.
- **Tkinter** – Provided a GUI for non-technical usage.
- **argparse** – Enabled command-line interface for batch automation.
- **Logging** – Used to track all activity and file-level status during processing.

Pretrained YOLOv8 Model Integration

Since the team did not have access to real Aadhar card image datasets (due to privacy constraints), we leveraged a pretrained YOLOv8 Nano model specifically trained on Aadhar cards:

- **Model Details**
 - Model Type: YOLOv8-nano
 - Hosted on Hugging Face: [arnabdhar/YOLOv8-nano-aadhar-card](#)
 - Purpose: Detect text regions corresponding to 12-digit Aadhar numbers on scanned documents.
- **Implementation Steps**
 - Downloaded model weights and configuration from Hugging Face.

- Integrated the model using the ultralytics Python package (from `ultralytics import YOLO`).
- Used the `model.predict()` method to infer bounding boxes from images.
- For each detected region, passed it to OCR + regex for content verification.

This approach allowed us to:

- Skip the time-consuming and resource-heavy step of training a custom YOLO model.
- Immediately test the pipeline with realistic data, maintaining compliance by not using or storing real Aadhar data.

Image Preprocessing

To optimize both YOLO and OCR performance:

- Images were cleaned and resized.
- Rotated or skewed images were deskewed using OpenCV.
- Thresholding and blurring helped reduce background noise.

These preprocessing steps were modular and reusable across both detection and OCR components.

Conclusion

By the end of Week 5, I had:

- Integrated a ready-to-use YOLOv8 model specifically trained to detect Aadhar number regions.
- Eliminated the need for internal dataset curation.
- Built a clean and modular architecture to support hybrid inference using deep learning and OCR.

Chapter 5

WEEK 6

AADHAR PATTERN RECOGNITION USING REGEX AND PREPROCESSING

With the pretrained **YOLOv8 Nano** model integrated successfully into the pipeline, Week 6 was focused on building a layered detection mechanism. The system combined **deep learning-based visual detection**, **OCR-based text extraction**, and **regex-based pattern validation** to ensure accurate and reliable masking of Aadhar numbers across a variety of document types.

YOLO-Based Detection of Aadhar Regions

- The pretrained YOLOv8 model was used to identify visual regions on the image that were likely to contain Aadhar numbers.
- Each bounding box predicted by YOLO represented a high-confidence “Aadhar candidate region”.
- This step significantly narrowed the OCR scope to relevant parts of the document, reducing false positives and improving speed.

```
from ultralytics import YOLO
model = YOLO("YOLOv8-nano-aadhar-card.pt")
results = model.predict(image_path)
```

- The output `results` contained bounding boxes with coordinates, confidence scores, and class IDs.
- These coordinates were then cropped from the original image and sent to the OCR pipeline.

OCR and Regex for Pattern Validation

Once a region was identified by YOLO:

- It was passed through **Tesseract OCR** to extract the textual content.
- The extracted text was cleaned and validated using a regular expression pattern to confirm whether it matched an actual Aadhar number:

```
python
r"\b\d{4}[\s-]?\d{4}[\s-]?\d{4}\b"
```

- This step filtered out false detections (e.g., 12-digit invoice numbers or phone numbers).

Image Preprocessing for OCR

To ensure high OCR accuracy, the cropped image regions were pre-processed before being passed to Tesseract:

- Converted to grayscale.
- Applied adaptive or Otsu thresholding to improve text contrast.
- Used median blur to reduce image noise.
- Aligned rotated images using contour-based skew correction.

These preprocessing enhancements significantly improved OCR output quality, especially for low-resolution and scanned images.

Hybrid Detection Pipeline

The overall workflow at this stage was:

1. **YOLOv8 Inference** → Generate bounding boxes for likely Aadhar number areas.
2. **OCR Extraction** → Extract text within each detected region.
3. **Regex Filtering** → Confirm the text matches Aadhar formatting.
4. **Coordinate Logging** → Store bounding box coordinates for downstream masking.

This hybrid pipeline combined the best of computer vision and traditional pattern matching to produce high-confidence results.

Conclusion

By the end of Week 6, the project had a robust Aadhar detection mechanism:

- YOLO handled spatial localization of sensitive content.
- OCR and regex provided semantic validation.
- Preprocessing steps improved accuracy and robustness on challenging images.

The next step was to implement the masking logic that would visually redact these verified Aadhar numbers from the image or PDF.

Chapter 6

WEEK 7-8

OCR IMPLEMENTATION AND TESTING WITH TESSERACT

With the YOLOv8 model successfully detecting regions of interest in documents, Weeks 7 and 8 were dedicated to integrating and fine-tuning **Tesseract OCR** to accurately extract text from those regions. The main focus was to verify whether the YOLO-identified regions indeed contained Aadhar numbers, and to reject any irrelevant content through pattern validation.

OCR Setup and Integration

- Integrated **Tesseract OCR** using the `pytesseract` Python wrapper.
- Extracted text from cropped bounding box regions output by the YOLOv8 model.
- Configured Tesseract with appropriate page segmentation mode (PSM 6) and OCR Engine Mode (OEM 3) for best performance on small cropped regions.

```
import pytesseract
text = pytesseract.image_to_string(cropped_region, config="--
psm 6 --oem 3")
```

- Each OCR result was then cleaned and checked for Aadhar-like structure using regex.

Regex-Based Validation

To avoid false positives, extracted text was passed through a regular expression pattern:

```
r"\b\d{4}[\s\~]?\d{4}[\s\~]?\d{4}\b"
```

- Only bounding boxes whose OCR content matched the regex pattern were marked for masking.
- This ensured that visual matches by YOLO (e.g., license numbers, account numbers) weren't falsely masked unless they followed the Aadhar format.

OCR Performance Challenges

While Tesseract OCR is a mature tool, several real-world image artifacts introduced challenges:

- **Low Resolution or Blur**
 - OCR struggled with small or blurred fonts.
 - Solved by applying resizing and thresholding before OCR.
- **Rotation and Skew**
 - Detected and corrected orientation using OpenCV before OCR.

- Improved recognition on tilted or side-scanned documents.
- **Overlapping Text or Background Noise**
 - Applied median filtering and morphological operations to reduce clutter.
- **OCR Misreads**
 - Common digit-letter confusion (e.g., 0 as O, 1 as I, 5 as S).
 - Added filters to correct obvious errors or apply confidence thresholds to OCR results.

Testing and Evaluation

- Ran batch tests on dozens of documents including:
 - Cleanly scanned Aadhar cards
 - Noisy, low-quality mobile captures
 - PDF pages with embedded ID sections
- Measured performance using:
 - True Positive (TP): Correct Aadhar numbers detected
 - False Positive (FP): Non-Aadhar text mistakenly flagged
 - False Negative (FN): Aadhar numbers missed
- Achieved strong precision due to strict regex validation, with acceptable recall due to preprocessing improvements.

Conclusion

By the end of Week 8:

- Tesseract was fully integrated into the YOLO-based pipeline.
- OCR + regex validation significantly reduced the chance of over-masking or missing sensitive information.
- The hybrid system demonstrated high reliability on noisy, varied real-world documents.

The next phase focused on developing the image masking logic to redact the detected Aadhar number regions securely.

Chapter-7

WEEK 9

IMAGE MASKING AND ANONYMIZATION PIPELINE

With detection and validation mechanisms in place, Week 9 was centred around implementing the **image masking system** — the core anonymization functionality responsible for visually redacting Aadhar numbers from documents.

Objective

The goal was to take the bounding boxes of verified Aadhar number regions (output from the YOLO + OCR + regex pipeline) and mask them using image processing techniques while preserving the integrity of the rest of the document.

Masking Strategy

Two masking techniques were considered, but the implementation focused on the more secure and consistent method:

- **Black Rectangle Masking (Implemented)**
 - Drew solid black rectangles over the bounding boxes of Aadhar numbers using OpenCV's `cv2.rectangle()` function.
 - Padding was added to the box dimensions to account for OCR margin errors and ensure full coverage.
 - This approach provided a clean, irreversible masking that met data privacy compliance.
- **Character Substitution (Not used in production)**
 - An experimental method attempted replacing Aadhar digits with 'X' using OCR output.
 - However, challenges with text alignment and rendering on varied layouts led to this approach being excluded from final deployment.

Workflow Implementation

The masking logic followed this flow:

1. YOLO detected bounding boxes in the input image.
2. Each bounding box was passed to OCR.
3. If the OCR text matched the Aadhar pattern, the box was marked for masking.
4. OpenCV applied a black rectangle to cover the bounding box.

```
cv2.rectangle(image, (x1, y1), (x2, y2), (0, 0, 0), thickness=-1)
```

Output Generation

- Masked images were saved with filenames appended with `_masked` for differentiation.
- For multi-page PDFs:
 - Each page was converted to an image, masked, then optionally stitched back into a new PDF using `img2pdf`.

Exception Handling

- Checks were added to avoid drawing boxes outside the image boundary.
- Bounding box coordinates were validated before masking.
- Logs were maintained for every mask operation, including:
 - Image name
 - Coordinates of the box
 - Detected text
 - Masking status (success/failure)

Testing and Visual Validation

- Ran tests on various sample documents with one or more Aadhar numbers.
- Verified that the masked output:
 - Covered all digits fully.
 - Did not leak any visible Aadhar data.
 - Preserved the rest of the document content for usability.

Conclusion

By the end of Week 9:

- The anonymization module was complete and reliable.
- It visually removed sensitive content while preserving document structure.
- The pipeline was now capable of processing images and PDFs end-to-end from detection to masking with a high level of accuracy and security.

Chapter 8

WEEK 10

ERROR HANDLING AND EDGE CASE VALIDATION

With the end-to-end Aadhar masking pipeline working reliably under normal conditions, Week 10 was dedicated to **error handling** and validating the system against **non-standard or edge-case scenarios**. This phase focused on ensuring the tool was production-ready and capable of gracefully handling unpredictable document inputs.

Objective

The goal was to simulate real-world inconsistencies—such as poor-quality images, unexpected formats, or failed detections—and reinforce the system’s ability to:

- Fail safely without crashing.
- Provide meaningful logs.
- Avoid incorrect masking decisions.

Error Handling Enhancements

1. Unsupported File Types

- Added validation for allowed file formats (.jpg, .jpeg, .png, .pdf).
- Skipped unsupported files with a warning logged.

2. Corrupted or Empty Files

- Handled cases where:
 - PDFs had no extractable pages.
 - Images were unreadable or had zero dimensions.
- Logged these issues and continued processing the remaining files.

3. Missing OCR Output

- Sometimes, even valid bounding boxes yielded empty OCR results due to poor image quality.
- These cases were logged and skipped from masking to avoid blacking out unrelated content.

4. Out-of-Bounds Bounding Boxes

- Verified that YOLO bounding box coordinates did not exceed image dimensions.
- Implemented clipping or rejection for boxes extending outside the valid range.

5. Exception Safety

- Wrapped all major steps (YOLO inference, OCR, masking, file writing) in try-except blocks.
- Caught and logged exceptions like:
 - FileNotFoundError
 - IndexError (common in PDF page parsing)
 - ValueError (from image format conversion)

Edge Case Testing

The tool was tested against a series of edge cases to evaluate its reliability:

- **Skewed or Rotated Images**
 - Confirmed deskewing handled moderate rotations.
 - For extreme cases, YOLO detection remained reliable due to visual features.
- **Low-Quality or Blurry Scans**
 - Observed degradation in OCR accuracy.
 - However, bounding box detection remained usable, so these were masked if confidence was high.
- **Multiple Aadhar Numbers**
 - Verified that the system correctly handled and masked all valid detections across the page.
- **Aadhar-Like Number Sequences**
 - Ensured that 12-digit numeric patterns unrelated to Aadhar (like invoice IDs or phone numbers) were not masked unless regex and OCR matched.
- **Multilingual Documents**
 - The system was robust since detection was visual (YOLO-based) and validation relied only on numeric recognition.

Logging and Transparency

- Logs included:
 - File name
 - Detected regions and OCR outputs
 - Regex matches
 - Masking decision
 - Errors or warnings (if any)
- Provided a reliable audit trail for every processed file.

Conclusion

By the end of Week 10:

- The tool had built-in safety against crashes and false operations.
- It demonstrated robust performance even under edge conditions like rotated scans, partial occlusion, and low-quality inputs.
- Comprehensive logging ensured traceability and maintainability.

The next milestone was to complete GUI integration and test the tool with real users internally.

Chapter 9

WEEK 11

INTEGRATION LOGGING AND USER TESTING

By Week 11, the core detection and masking engine was complete. The focus now shifted to integrating the different components into a cohesive software tool and exposing it through both a **command-line interface (CLI)** and a **Tkinter-based graphical user interface (GUI)**. The system was then tested with real users for feedback and refinement.

System Integration

All functional modules — YOLO-based detection, OCR and regex validation, and OpenCV-based masking — were combined into a unified Python package. The integration followed a modular structure:

- `yolo_model.py` – Handled object detection using the pretrained YOLOv8 Nano model.
- `ocr_engine.py` – Managed text extraction and validation with Tesseract and regex.
- `masker.py` – Contained image masking logic using bounding box coordinates.
- `main.py` – Served as the CLI runner and orchestrator for all modules.
- `gui.py` – Provided a Tkinter-based interface for non-technical users.

Each part communicated cleanly through standardized data formats (bounding boxes, text strings, flags), making the pipeline easy to debug and extend.

Command-Line Interface (CLI)

A user-friendly CLI was developed using the `argparse` module to allow flexible execution:

```
python main.py --input-folder ./data/ --output-folder ./masked/
--log-level DEBUG
```

CLI features included:

- Input and output folder selection
- Verbose logging toggle
- Optional preview of detected regions (for debugging)

This mode was ideal for batch processing and automation in technical environments.

Graphical User Interface (GUI)

To make the tool accessible to users without programming knowledge, I built a simple GUI using **Tkinter**:

- Allowed users to:
 - Select input image or PDF
 - Choose output folder
 - Run masking with one click
 - View processing status and messages

The GUI made it easy for internal staff to test the tool without needing to understand the underlying codebase.

Logging and Audit Trail

A structured logging system was finalized:

- Each run generated a timestamped `.log` file.
- Included details like:
 - File processed
 - Bounding boxes detected
 - OCR output
 - Regex match status
 - Masking decision
 - Errors or exceptions (if any)

Logs were essential for validation, debugging, and proving that masking was applied where necessary.

Internal User Testing

The integrated system was tested internally with non-technical users including:

- Business analysts
- QA engineers
- Document review staff

User feedback led to the following refinements:

- Improved GUI layout and labels
- Added tooltips and status bar messages
- Enhanced error messages for common issues (e.g., empty folders, unsupported files)
- Implemented success/failure summary dialogs after processing

Conclusion

By the end of Week 11:

- The system had matured into a production-grade tool with both CLI and GUI modes.
- Integration was smooth and modular, making it extensible for future formats or detection targets.
- User testing validated the tool's effectiveness and usability across technical and non-technical audiences.

Chapter 10

WEEK 12

FINAL DEPLOYMENT, DOCUMENTATION AND HANDOVER

In the final week of my internship, I focused on **consolidating and deploying** the complete Aadhar masking solution. The goal was to make the tool production-ready, maintainable, and accessible to both developers and non-technical stakeholders after handover.

Final System Overview

The complete pipeline included:

- **YOLOv8 Nano (Pretrained)** model from Hugging Face for detecting Aadhar number regions.
- **Tesseract OCR** and **Regex** for confirming if detected text matched valid Aadhar patterns.
- **OpenCV** for masking verified regions with black rectangles.
- **CLI and Tkinter GUI** to serve different user preferences.
- **Logging** for auditability and transparency.

All components were modularized into Python scripts and packaged in a way that new contributors could easily understand and run the tool.

Deliverables

The final codebase included:

- `main.py` – CLI entry point.
- `gui.py` – Tkinter-based graphical interface.
- `yolo_model.py` – YOLOv8 inference logic using Hugging Face weights.
- `ocr_engine.py` – OCR and regex utilities.
- `masker.py` – Image masking engine.
- `utils.py` – File handling and support utilities.
- `requirements.txt` – Dependency list for setup.
- `README.md` – Setup and usage instructions for both CLI and GUI.
- `sample_inputs/` and `sample_outputs/` – For quick testing and validation.

Handover and Walkthrough

- Conducted a live walkthrough session with the QA and product teams.
- Demonstrated:

- Single-file and batch processing
 - Masking verification using logs and outputs
 - CLI automation for use in larger processing pipelines
- Shared documentation and provided access to a private GitHub repository for future collaboration.

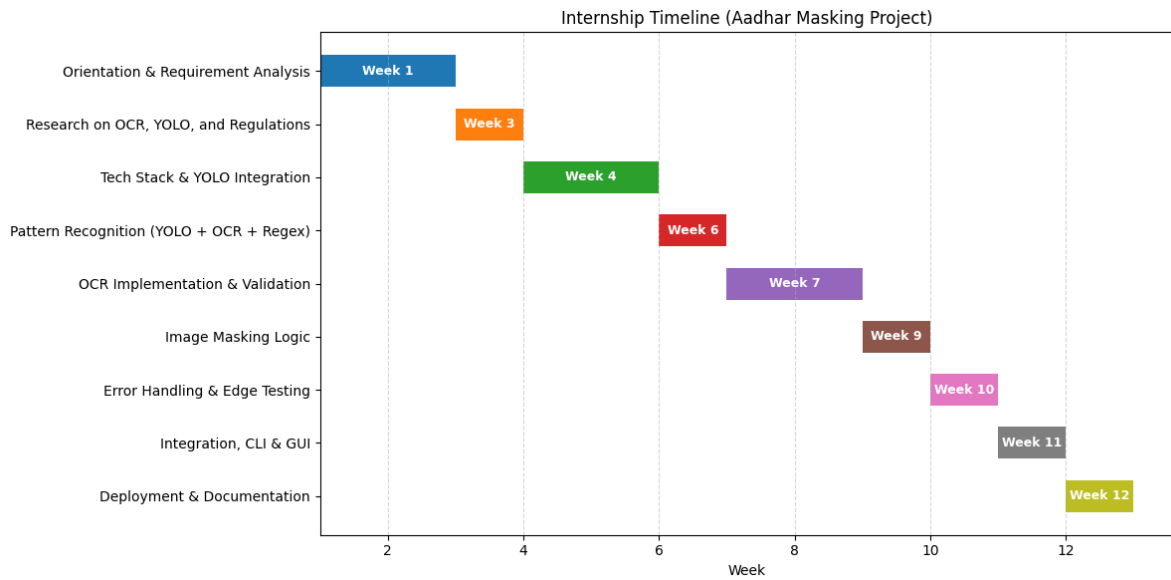
Final Outcome

By the end of Week 12:

- The project was successfully deployed and tested in a usable format.
- All documentation and usage support was delivered to the internal team.
- The tool was positioned to scale with minimal effort, and could later be enhanced with features like cloud deployment or multi-language support.

Chapter 11

TIMELINE AND WORK BREAKDOWN (GANTT CHART)



Chapter 12

OUTCOMES

During my three-month internship at **Abylle Solutions**, I had the opportunity to develop a complete, production-ready solution for privacy-focused document processing. The project centred on automatically detecting and masking Aadhar numbers in scanned documents and PDFs using a hybrid deep learning and OCR approach.

Through this experience, I gained practical expertise in:

- Integrating **pretrained YOLOv8 models** into a custom image-processing pipeline.
- Working with **OCR (Tesseract)** and **regular expressions** for content validation.
- Preprocessing real-world scanned documents for improved detection accuracy.
- Developing **modular, scalable Python codebases** with error handling and logging.
- Building both **CLI** and **GUI applications** using Python and **Tkinter**.

In addition, I learned how to balance technical design with **data privacy regulations**, ensuring that sensitive information was handled responsibly. The skills developed during this project — from model inference to GUI design — will serve as a strong foundation for future roles in computer vision, privacy-tech, or automation.

Chapter 13

CONCLUSIONS

Overall, this internship has been an invaluable learning experience that enhanced both my technical and problem-solving abilities. I worked end-to-end on a real-world privacy automation problem, gaining hands-on exposure to core technologies in **deep learning, OCR, and image processing**.

By leveraging a **pretrained YOLOv8 model** and combining it with **OCR-based text validation**, I successfully built a robust pipeline capable of identifying and redacting sensitive Aadhar numbers with high accuracy. I also developed a **user-friendly GUI** to enable broader usability and ensure that the solution could be operated by non-technical users.

This project helped me understand the importance of building **modular, production-ready systems** that are not only accurate but also transparent, auditable, and aligned with regulatory standards.

Looking forward, I intend to continue working on privacy-first solutions and expand my expertise in domains like document automation, AI for governance, and real-time document analysis. The confidence and skillset I've gained through this internship will help me contribute meaningfully to future roles in both the research and industry space.

SCREENSHOTS

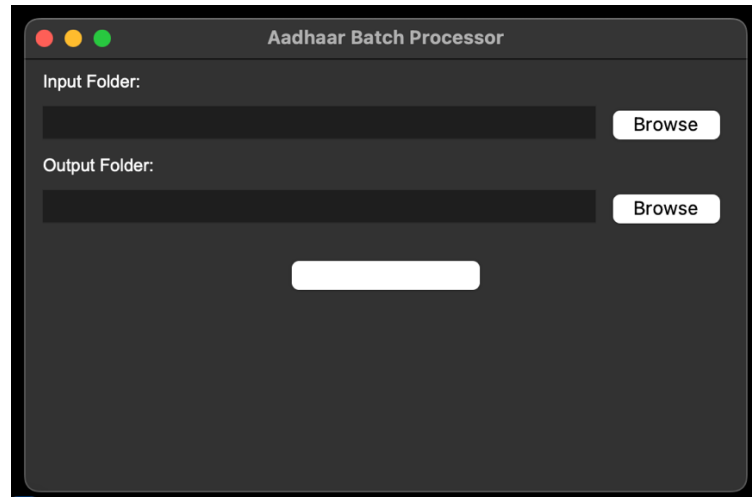


Fig 1.1 GUI developed using Tkinter



Fig 1.2 Example of Aadhar image post processing

SUSTAINABLE DEVELOPMENT GOALS



SDG 4: Quality Education

Target 4.4: Substantially increase the number of youth and adults with relevant skills, including technical and vocational skills, for employment, decent jobs, and entrepreneurship.

How it aligns:

- This project served as a **learning platform** to develop hands-on skills in:
 - AI model integration (YOLO)
 - OCR and data validation
 - Secure software development
 - GUI/CLI application development
- It reflects **practical application of emerging technologies** in solving real-world problems.
- Encourages other students and interns to pursue **ethical AI and privacy engineering** as a professional path.

SDG 8: Decent Work and Economic Growth

Target 8.2: Achieve higher levels of economic productivity through diversification, technological upgrading and innovation.

How it aligns:

- The system **automates manual data redaction**, improving organizational efficiency.
- By reducing human error and operational risk in sensitive document handling, it supports **sustainable business process automation**.

SDG 9: Industry, Innovation, and Infrastructure

Target 9.5: Enhance scientific research, upgrade the technological capabilities of industrial sectors.

How it aligns:

- The project uses **advanced technologies like YOLOv8 and OCR** for real-time privacy automation.
- It demonstrates **innovation in data handling infrastructure**, especially in digitized identity management.
- Integrating AI in document security pipelines promotes technological advancement in digital governance systems.

SDG 11: Sustainable Cities and Communities

Target 11.6: Reduce the adverse per capita environmental impact of cities.

How it aligns:

- The digital-first approach to data processing and masking reduces the need for **paper-based manual document workflows**, supporting eco-friendly digitization.
- Promotes **secure digital infrastructure** in urban governance, banking, and public welfare systems.

SDG 16: Peace, Justice, and Strong Institutions

Target 16.10: Ensure public access to information and protect fundamental freedoms.

How it aligns:

- By **automatically masking sensitive Aadhar numbers**, the tool strengthens **data protection and privacy rights**, aligning with constitutional protections of identity.
- The project encourages **responsible data governance and compliance**, crucial for digital institutions handling citizen data.

SDG 17: Partnerships for the Goals

Target 17.8: Enhance the use of enabling technology, in particular information and communications technology.

How it aligns:

- This project is an excellent demonstration of how **open-source AI tools (Tesseract, YOLO)** and platforms like Hugging Face can be used in responsible digital solutions.
- Encourages **collaborative innovation** and tech adoption in privacy engineering.

PLAGIARISM REPORT



Page 2 of 35 - Integrity Overview

Submission ID trn:oid::1:3249770739





13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography

Match Groups

-  **20 Not Cited or Quoted 13%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 9%  Internet sources
- 6%  Publications
- 12%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.





A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.






Page 2 of 35 - Integrity Overview

Submission ID trn:oid::1:3249770739

**Match Groups**

-  **20 Not Cited or Quoted 13%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 9%  Internet sources
- 6%  Publications
- 12%  Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Presidency University		11%
2	Internet	
www.kamarajengg.edu.in		<1%
3	Publication	
"Second International Conference on Image Processing and Capsule Networks", S...		<1%
4	Internet	
naac.cuh.ac.in		<1%
5	Internet	
programtalk.com		<1%
6	Internet	
jimsd.org		<1%
7	Internet	
mite.ac.in		<1%
8	Internet	
www.manufacturerssuppliersexporters.com		<1%



INTERNSHIP CERTIFICATE



Abylle Solutions Private Limited

2, Ratna Complex, Gurappa Avenue, Primrose Road
Off M.G. Road, Bengaluru, Karnataka, India - 560 025
Phone: +91 80 4090 1198 website: www.abylle.com

Ref # ASPL/HRD/ICL/68/2025/02

13th May 2025

Sub: Internship Completion Letter

TO WHOM IT MAY CONCERN

This is to certify that **Allen Conroy Dsouza (Reg #: 20211CSE0227 – Presidency University)** has successfully completed an internship with Abylle Solutions Pvt Ltd for a period of 03 months between 03rd February 2025 to 02nd May 2025.

He has completed his internship on AI and ML.

Besides showing high comprehension capacity, managing assignments with the utmost expertise, and exhibiting maximal efficiency, **Allen Conroy Dsouza** has also maintained an outstanding professional demeanor and showcased excellent moral character throughout the internship period.

During his internship with us, he was exposed to various processes and was found to be diligent, hardworking, and inquisitive.

I hereby certify him overall work as excellent to the best of my knowledge and wish him every success in his future endeavors.

For
Abylle Solutions Pvt Ltd

A handwritten signature in blue ink, appearing to read "Allen Conroy Dsouza", written over a horizontal line.

Authorized Signatory



Accepted by