



TRIBHUVAN UNIVERSITY

Institute of Science and Technology

A Project

On

**“Vehicle Number Plate Detection and
Character Recognition using CNN”**

Submitted to:

Department of Computer Science and Information Technology

ASIAN COLLEGE OF HIGHER STUDIES

Ekantakuna, Lalitpur

*In partial fulfillment of the requirements for the Bachelors in
Computer Science and Information Technology*

Submitted by:

Alish Tuladhar (26705/077)

Arbin Shrestha (26707/077)

Shreejan Shrestha (26737/077)

Supervised by:

Radha Krishna Gajurel

TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY



SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project is prepared by **Mr. Alish Tuladhar (26705/077)**, **Mr. Arbin Shrestha (26707/077)** and **Mr. Shreejan Shrestha (26737/077)** under the supervision of **Mr. Radha Krishna Gajurel** entitled “Vehicle Number Plate Detection and Character Recognition using CNN” in partial fulfillment of the requirements for the degree of Bachelor of Computer Science and Information Technology be processed for the final evaluation.

Radha Krishna Gajurel

Project Supervisor

Department of Computer Science and IT

LETTER OF APPROVAL

This is to certify that this project is prepared by Alish Tuladhar, Arbin Shrestha and Shreejan Shrestha entitled “Vehicle Number Plate Detection and Character Recognition using CNN” in partial fulfillment of the requirements for the degree of Bachelor in Computer Science and Information Technology has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

<p>Signature of Supervisor</p> <p>.....</p> <p>Mr. Radha Krishna Gajurel Asian College of Higher Studies</p>	<p>Signature of Program Coordinator</p> <p>.....</p> <p>Mr. Pranaya Nakarmi Asian College of Higher Studies</p>
<p>Signature of Internal Examiner</p> <p>.....</p> <p>Asian College of Higher Studies</p>	<p>Signature of External Examiner</p> <p>.....</p> <p>Asian College of Higher Studies</p>

ACKNOWLEDGEMENT

Gaining knowledge in Information Technology often extends beyond theoretical concepts, particularly in projects like Vehicle Number Plate Detection and Character Recognition, where practical applications play a pivotal role in driving innovation. This project has been a rewarding experience, allowing us to address real-world challenges using advanced technologies like YOLO and Convolutional Neural Networks (CNN).

We are deeply grateful for the unwavering support of our teachers, staff, and everyone who contributed to this project, as their guidance and encouragement have been instrumental in its success. Our heartfelt thanks go to our supervisor, **Mr. Radha Krishna Gajurel**, for his invaluable advice, motivation, and continuous support throughout this journey. We also extend our gratitude to the Program Coordinator, **Mr. Pranaya Nakarmi**, for his guidance and support, which have been crucial in helping us achieve our goals.

Additionally, we express our sincere appreciation to the **Asian College of Higher Studies** and its administration for approving and supporting our project request. Finally, we sincerely thank all individuals who directly or indirectly assisted us with their cooperation, support, and encouragement, making this report possible.

Project Members:

Alish Tuladhar (TU Roll No.: 26705/077)

Arbin Shrestha (TU Roll No.: 26707/077)

Shreejan Shrestha (TU Roll No.: 26737/077)

Date: January 21, 2025

ABSTRACT

The Vehicle Number Plate Detection and Character Recognition system makes license plate recognition simple and efficient. Using YOLO for detecting plates and a custom CNN for recognizing characters, the system achieves 92% accuracy under ideal conditions like good lighting and standard angles. It's designed to handle different scenarios, including varied lighting, and plate styles. The web interface, created with FastAPI, is easy to use, allowing users to upload images or videos and get real-time results. All recognized plate data is securely stored in a MySQL database, ensuring reliability and easy access when needed. Testing has shown the system performs well, with high accuracy in recognizing characters. Planned improvements, such as real-time video processing, better handling of challenging inputs, and support for regional variations, will make it even more versatile. With applications in traffic management, parking systems, and law enforcement, this project offers a smarter, more efficient approach to transportation challenges.

Keywords: *YOLO, CNN, FastAPI, License Plate Recognition, MySQL, Character recognition*

TABLE OF CONTENT

SUPERVISOR’S RECOMMENDATION	ii
LETTER OF APPROVAL	iii
ACKNOWLEDGEMENT.....	iv
ABSTRACT.....	v
LIST OF ABBREVIATIONS	viii
LIST OF FIGURE	ix
LIST OF TABLES.....	x
CHAPTER 1 INTRODUCTION	1
1.1 Introduction.....	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Scope and Limitation	2
1.5 Development Methodology	3
1.6 Report Organization.....	5
CHAPTER 2 BACKGROUND STUDY AND LITERATURE REVIEW	7
2.1 Background Study.....	7
2.2 Literature Review.....	8
CHAPTER 3 SYSTEM ANALYSIS	9
3.1 System Analysis	9
3.1.1 Requirement Analysis	9
3.1.2 Feasibility Analysis.....	12
3.1.3 Analysis – Structured Approach	15
CHAPTER 4 SYSTEM DESIGN.....	19
4.1 Design - Structured approach.....	19
4.2 Algorithm Details.....	21

4.2.1	Number Plate Detection Using YOLOv8	21
4.2.2	Character Segmentation Using YOLOv8	22
4.2.3	Character Sorting	22
4.2.4	Character Recognition Using CNN	23
4.2.5	Text Reconstruction	24
4.2.6	Video Frame Processing	24
4.2.7	Activation Function:	24
CHAPTER 5 IMPLEMENTATION AND TESTING.....		26
5.1	Implementation	26
5.1.1	Tools Used	26
5.1.2	Implementation Details	27
5.2	Testing.....	29
5.2.1	Test Cases for Unit Testing	29
5.3	Result Analysis	32
5.3.1	Model Performance Overview	32
5.3.2	Performance Metrics Comparison	32
5.3.3	Hyperparameter Tuning Analysis	34
5.3.4	Data Augmentation and Preprocessing	35
5.3.5	Model Architecture Refinements	35
5.3.6	Confusion Matrix	35
CHAPTER 6 CONCLUSION AND FUTURE RECOMMENDATION		37
6.1	Conclusion	37
6.2	Future Recommendation.....	37
REFERENCE		
APPENDICES		

LIST OF ABBREVIATIONS

ANPR	Automatic Number Plate Recognition
API	Application Programming Interface
BGR	Blue-Green-Red
CNN	Convolutional Neural Networks
CUDA	Compute Unified Device Architecture
DFD	Data Flow Diagram
ER	Entity Relationship
GPU	Graphics Processing Unit
MySQL	My Structured Query Language
NPR	Number Plate Recognition
OpenCV	Open Source Computer Vision Library
RGB	Red-Green-Blue
ReLU	Rectified Linear Unit
SVM	Support Vector Machines
UAT	User Acceptance Testing
YAML	YAML Ain't Markup Language
YOLO	You Only Look Once

LIST OF FIGURE

Figure 1.1 Waterfall Model.....	3
Figure 3.1 Use Case Diagram.....	9
Figure 3.2 Gantt Chart	13
Figure 3.3 Data Modelling using ER Diagram	15
Figure 3.4 DFD Level 0.....	17
Figure 3.5 Process Modelling using DFD.....	17
Figure 4.1 System Design	19
Figure 5.1 Graphical Representation of training for 20 epoch	33
Figure 5.2 Graphical Representation of training for 40 epoch	33
Figure 5.3. Confusion Matrix of Segmentation	36
Figure 5.4 Confusion Matrix of Number Plate	36

LIST OF TABLES

Table 5.1: Unit Test Case: URL	29
Table 5.2: Unit Test Case for Uploading, Detecting, Segmenting, Saving.....	30
Table 5.3: Unit Test Case for System.....	30
Table 5.4: Unit Test Case for Output	31
Table 5.5: Result and Analysis.....	34

CHAPTER 1

INTRODUCTION

1.1 Introduction

Number Plate Detection systems are important systems in transportation management and surveillance. In the last few years, the Number Plate Recognition system has played a significant role in vehicle surveillance. The system is used to recognize a vehicle owner after the license plate is detected and then classified [1] [2]. The main goal of the ANPR is to identify and recognize number from their plate numbers. Nowadays, these ANPR systems are one of the important topics in the real world [3]. The ANPR has many applications in today's life, such as car traffic surveillance, traffic monitoring for safety and security, control and management of parking systems, and other service purposes [4]. Moreover, it is used in various aspects of life, such as urban logistics, identifying vehicle owners, electronic toll collection, and smart cities management. Vehicle number plate detection technology can help to automate many tasks that were previously done manually.

In today's digital age, automatic systems are everywhere. ANPR can capture images and text from license plates. But ANPR needs to handle plate variations from place to place. Control systems are increasingly replacing manual operators, while fully automated machines are replacing human labor.

The Department of Transport Management in Nepal has started issuing high-security, embossed number plates that ANPR cameras can read. These plates have a chip connected to the vehicle's GPS, which helps find the car if it's stolen.

1.2 Problem Statement

Originally designed for identification purposes by government officials such as police officers and transportation personnel, these systems aid in tracking and managing vehicles effectively. However, manual methods of identification can be time-consuming and prone to errors, particularly in high-traffic scenarios. Automated number plate recognition systems address these challenges by enabling efficient detection and processing of number plates. These systems can streamline operations such as access control in restricted parking areas, traffic management, and toll collection. By employing advanced camera technology capable of rapid focus, number plates can be detected accurately under various conditions, reducing human intervention and enhancing the overall efficiency of transportation systems.

1.3 Objectives

- To detect vehicle number plates and recognize characters on them using a trained CNN in a web application that allows users to upload images or videos for real-time recognition.

1.4 Scope and Limitation

I. Scope

- The ANPR system is designed to ‘detect and recognize number plates’, widely used in global traffic monitoring systems.
- It aids ‘law enforcement agencies’ by identifying suspicious or criminal vehicles and providing both ‘alibis’ and ‘incriminating evidence’.
- Enhances ‘security in workplace parking management’ and helps restrict access to authorized vehicles.
- Addresses the challenges posed by the ‘diverse shapes and sizes’ of license plates globally using tailored recognition methods.

II. Limitation

- The system raises ‘privacy concerns’ due to the need to store and access sensitive data, potentially subject to misuse.
- Ensuring privacy requires ‘secure storage mechanisms’ and strict ‘access control’ for legitimate purposes only.
- Variations in license plate design (e.g., fonts, languages, sizes, and lighting conditions) may affect the system's accuracy and require robust algorithms.

1.5 Development Methodology

The Waterfall Model was chosen for this project due to its straightforward, step-by-step process, which is ideal for following in a linear sequence. Each phase must be completed before moving to the next, making it easier to manage when the project’s requirements are clearly defined from the start. This approach is well-suited for academic projects like this one, where the requirements are known in advance. Although backtracking in the Waterfall model can be challenging, the testing phase allows for repeated iterations to achieve the desired results.

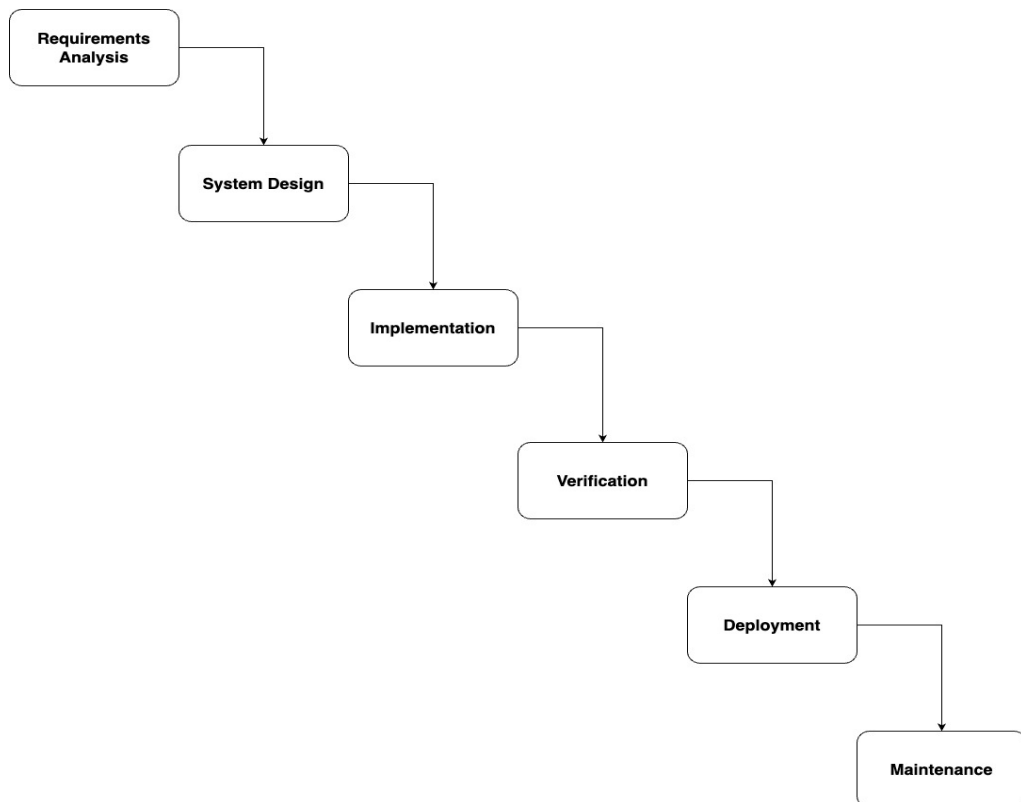


Figure 1.1 Waterfall Model

Requirement Gathering and Analysis:

In this phase, all necessary software requirements are gathered and documented by the project team. This process helps ensure that every critical aspect of the project is captured and provides important insights into what needs to be developed.

System Design:

During this phase, the system's design is finalized, determining the architecture and components based on the previously collected requirements. Both High-Level Design and Low-Level Design are developed and approved by the team. High-level diagrams like DFD and interface designs are created to outline the system's structure and specifications.

Implementation (Coding):

This is where the actual coding takes place, translating the finalized designs into functioning code. Developers bring the planned system to life, producing both the source code and the executable files. This step is the practical realization of the designs established in earlier stages.

Integration and Testing:

As the longest phase in the Software Development Life Cycle, this stage continues until the system meets the accuracy and requirements defined during the design process. Multiple testing cycles are conducted, including unit, integration, system, and UAT, to validate the system. This iterative process continues until the expected outcomes are achieved, producing detailed testing reports that will be included in the final project documentation.

Deployment:

At this point, the system is deployed, with thorough testing to ensure it runs smoothly without any issues in a production environment. The system is installed, configured, and tested to identify and resolve any potential production errors.

Maintenance:

In this phase, the system is monitored for any post-deployment issues. Any bugs that arise are fixed, and updates are made to accommodate new requirements. This step marks the conclusion of the Software Development Life Cycle, ensuring the system is both updated and bug-free.

1.6 Report Organization

This report consists of six chapters, where each chapter addresses different aspects of the project in a structured manner, providing a comprehensive understanding of the system.

I. Introduction:

The first chapter introduces the project and provides an overview of the automated vehicle number plate detection and recognition system. It outlines key elements such as the introduction, problem statement, objectives, scope and limitations, development methodology, and report organization.

II. Background Study and Literature Review:

The second chapter explores the fundamental theories, concepts, and technologies used in the project, such as YOLO, CNN, and FastAPI. It also includes a literature review of similar works and research studies, identifying gaps and positioning the project in the context of existing solutions.

III. System Analysis:

The third chapter details the system analysis, including the requirements analysis and feasibility study. It examines the functional and non-functional requirements, evaluates technical and operational feasibility, and uses structured modeling tools like ER diagrams and DFDs to design the system's processes.

IV. System Design:

The fourth chapter focuses on the system's design. It covers the database schema, frontend and backend integration, algorithms for detection and recognition, and

API design. The chapter provides insights into how these components are interconnected to deliver the desired functionality.

V. Implementation and Testing:

This chapter discusses the implementation process, including the tools, frameworks, and methods used to build the system. It also explains the testing phase, which involves unit testing for individual models like YOLO and CNN and system testing for the integrated workflow. The results demonstrate the system's accuracy and reliability.

VI. Conclusion and Future Plan:

The final chapter summarizes the completed work, highlighting the system's achievements in automating number plate detection and recognition. It also outlines future plans for enhancements, such as improving model performance, supporting video processing, and expanding the system's scalability for broader applications.

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

NPR, also known as ANPR, is a system employed to identify vehicle plates for various purposes, including traffic control, toll collection, and security monitoring. NPR systems typically comprise three stages: plate detection, character segmentation, and character recognition. Embossed plates, characterized by raised characters, pose challenges due to shadows and reflections, potentially impairing accurate recognition. Specialized preprocessing techniques, such as grayscale conversion, adaptive thresholding, and morphological operations, mitigate these effects, ensuring clearer images for recognition.

The YOLO model, a prominent choice for real-time object detection, demonstrates exceptional efficacy in swiftly and accurately locating number plates. OpenCV, a widely utilized computer vision library, supports YOLO by performing fundamental preprocessing steps, including noise reduction, image enhancement, and contrast augmentation [5]. Subsequently, character segmentation algorithms isolate each character, a particularly intricate task for embossed plates due to shadowed and blurred edges.

CNNs are widely used for character recognition, particularly in identifying patterns within image data. They are particularly well-suited for recognizing alphanumeric characters on embossed plates [6]. To train the model effectively, structured datasets with meticulously labeled character classes are essential. Additionally, hardware acceleration with CUDA-compatible GPUs is employed to efficiently train the model, even with large datasets [7]. This combination of technologies and preprocessing techniques forms a robust framework for accurate recognition of embossed number plates in real-world scenarios.

2.2 Literature Review

As briefly reviewed in the following ANPD approaches, research on Number Plate Detection methods has made considerable progress in recent years for the effectiveness of any Number Plate Recognizing system. The researchers discussed an ANPR system using extracted features based on template matching. They approach Number plate data detection by using gray scaling, dilation & erosion, normalization, and the Sobel Operator for vertical edge detection after that character segmentation was performed by binarizing the image. Template matching using cross-correlation was performed with the help of Optical character recognition where researchers designed a program to recognize Optical character recognition-based car number plates by using template matching and the bounding box method.

Prof. A.B.Gadicha [8] they proposed a straightforward approach using simple and efficient morphological operations and the Sobel edge detection method. They also presented a simple way to segment all the letters and numbers in the number plate. After reducing noise from the input image, they tried to boost the contrast of the binarized image using histogram equalization to identify each number separately.

M. M. Shidore [9] implemented the number plate extraction, character segmentation, and recognition work, utilizing English characters. Number plate extraction is achieved through the application of the Sobel filter, morphological operations, and connected component analysis. Character segmentation is accomplished through the utilization of connected components and vertical projection¹ analysis. Character recognition is performed using the SVM. The segmentation accuracy is 80%, and the recognition rate is 79.84%.

Prof. Pradnya Randive [10] proposed ALPR system works: the android phone captures the number plate photo, and the server processes it. The server extracts the plate region from the image, converts it to grayscale, and applies Otsu's algorithm to binarize it. Then, it uses morphological operations to enhance the number's visibility. Finally, the server sends the processed image back to the android phone, which can then display the extracted number.

CHAPTER 3

SYSTEM ANALYSIS

3.1 System Analysis

System analysis plays a crucial role in developing a robust **Vehicle Number Plate Detection and Character Recognition** system. This chapter provides a detailed examination of the system's requirements, feasibility, and design methodologies.

3.1.1 Requirement Analysis

i. Functional Requirements:

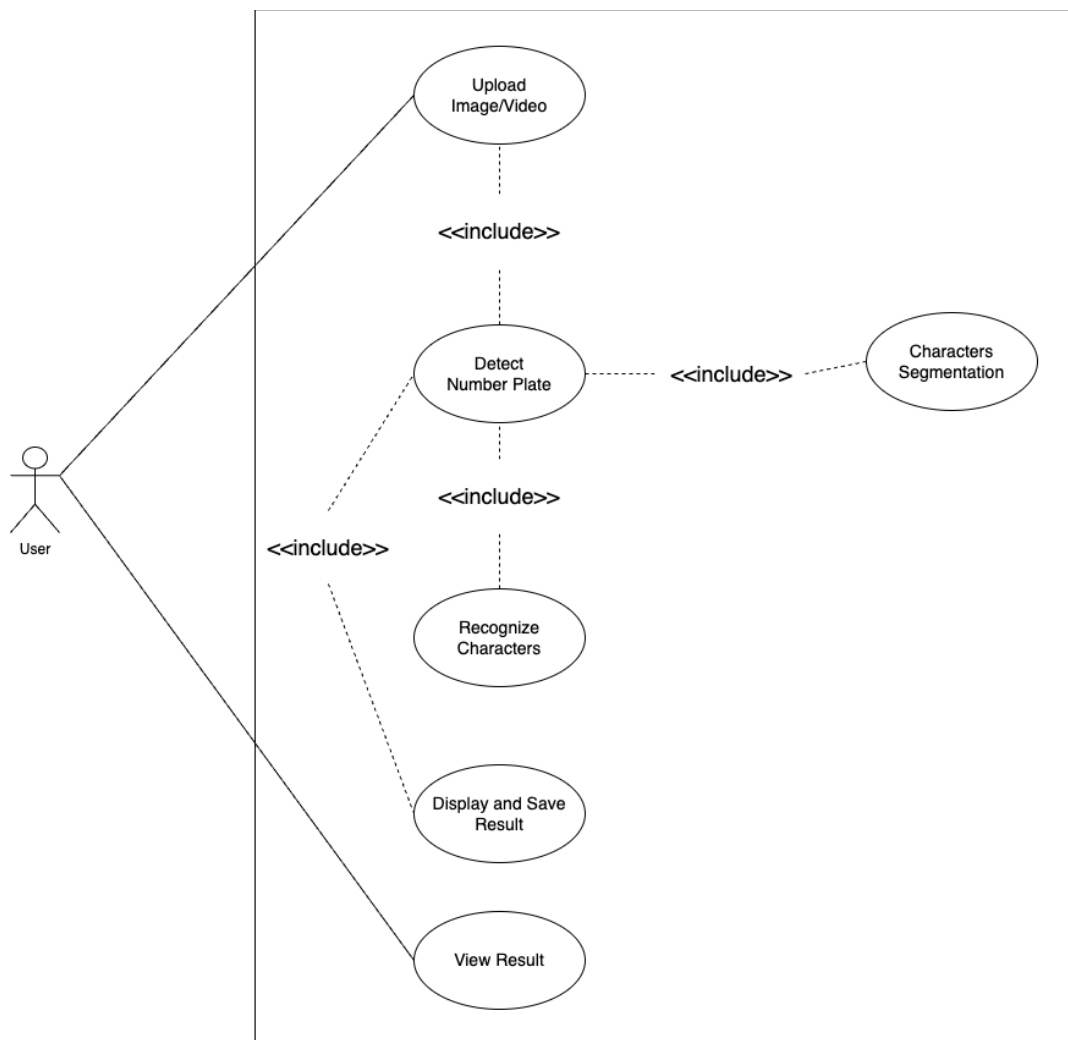


Figure 3.1 Use Case Diagram

- **Detect Number Plate**

The system should possess the capability to identify and detect vehicle license plates from input images or video streams. This entails applying image processing techniques to isolate the region of the license plate from the background. The detection process should be robust enough to handle various conditions, such as different lighting conditions, angles, and plate designs.

- **Segment Characters**

Once the system spots the number plate, it needs to break it down into its individual characters. This means splitting the plate text into separate characters, so it can be processed further. The system should be able to handle different fonts, sizes, spacing, and alignments of the characters.

- **Recognize Characters**

The system uses a special kind of computer program called a CNN to turn the broken-up characters into text. It does this by showing the CNN the pictures of the broken-up characters and telling it what each character is. The system is really good at recognizing characters, even if they're different fonts, sizes.

- **View Results**

The user should be able to easily view the number plate details through a friendly interface. The system should show the extracted text and maybe other useful info, making it clear and easy for the user to use.

- **Save to Database**

Let's save the recognized number plate details in a database. That way, we can keep track of them for future reference or use them for any other purposes. We want to make sure the storage process is secure, efficient, and easy to access.

ii. Non-Functional Requirements:

- **Performance**

The system must operate with minimal latency to enable real-time processing. Each step, from detection to recognition, should be optimized to ensure rapid processing, particularly when handling multiple images in sequence.

- **Accuracy**

Accuracy is key in both detection and recognition. Mistakes can lead to wrong info being captured, and that's no good, especially in legal or security situations where errors can have serious repercussions.

- **Scalability**

The solution needs to be able to handle more images and higher resolution inputs as the number of users and the amount of data increases. This is especially important for large-scale systems like urban traffic systems, where a lot of data is generated.

- **Reliability**

The system should work great in all sorts of conditions, like different lighting and weather. That way, it's reliable and effective in real life.

- **Security**

Since number plate data is super sensitive, we need to make sure the system is extra secure. This means using encryption and access controls to keep only the right people from seeing or using the system's data and outputs.

- **Usability**

The system should provide a straightforward interface for operators, necessitating minimal training to operate effectively. Usability enhancements, including transparent output displays and intuitive navigation, facilitate efficient interaction with the system.

3.1.2 Feasibility Analysis

I. Technical Feasibility

This project is highly achievable from a technical standpoint due to the availability of advanced tools and resources designed to meet its objectives. Technologies like OpenCV and YOLO are well-established for their capabilities in image processing and object detection. YOLO, in particular, is renowned for its speed and accuracy, making it ideal for detecting number plates efficiently and precisely. To complement this, a trained CNN will handle the segmentation and recognition of alphanumeric characters on these plates. With access to GPU resources, the project can tackle computationally intensive tasks such as model training and real-time processing effectively. The strong technical infrastructure ensures the system delivers accurate and timely results. Additionally, the structured approach of the waterfall model provides a clear, step-by-step development framework. Each phase analysis, design, implementation, and testing are completed thoroughly before moving forward, reducing risks and ensuring a reliable and stable system.

II. Operational Feasibility

The project is operationally practical and offers a user-friendly solution for real-time number plate recognition. Its straightforward web interface requires little to no user training, making it accessible even to individuals with limited technical knowledge. The system has wide-ranging applications, such as improving traffic management, streamlining automated tolling, enhancing parking systems, and strengthening security measures. These use cases emphasize its ability to automate repetitive tasks, boost efficiency, and minimize errors in scenarios where accurate and rapid number plate recognition is critical. By following the waterfall model, operational requirements are carefully identified and addressed early in the development process. This ensures the system is aligned with real-world needs and user expectations, resulting in a final product that is not only ready for deployment but also capable of performing consistently in practical settings.

III. Schedule Feasibility

The project is on track to meet its timeline due to a well-organized and phased development plan. Each stage has clearly defined goals and deadlines, enabling efficient time management. Key early steps, such as gathering requirements, collecting data, and conducting the initial system analysis, have already been completed. This provides a solid foundation for the remaining phases, including design, implementation, and testing. The use of tools like Gantt charts helps visualize the project timeline, marking start and end dates for each phase. This approach allows for real-time monitoring of progress and early detection of potential delays, enabling adjustments to stay on schedule. With its structured timeline and active monitoring, the project is set to achieve all milestones and be completed within the allotted timeframe.

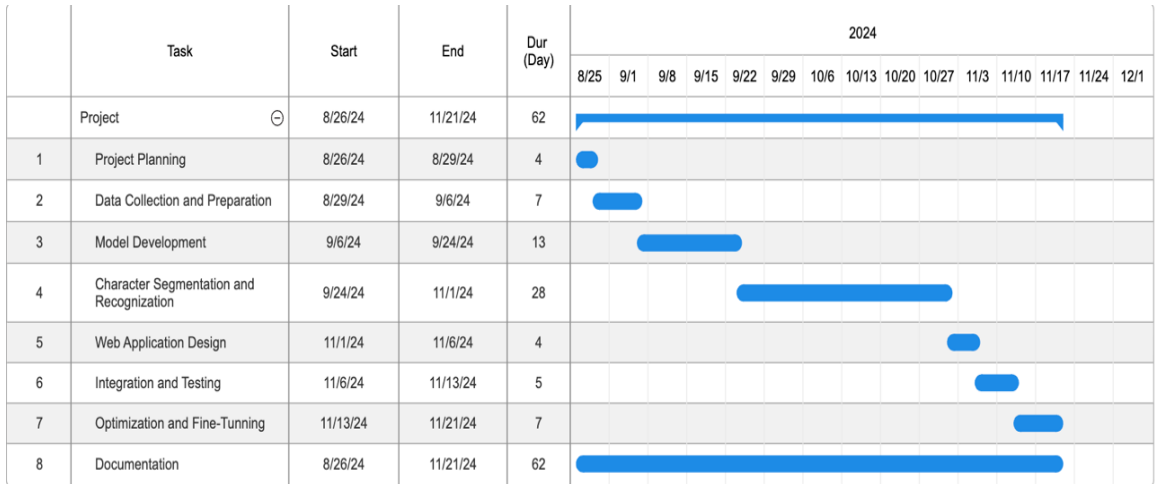


Figure 3.2 Gantt Chart

- **Project Planning:**

This initial phase focuses on setting up the project's scope, defining objectives, and establishing a timeline to ensure clear direction from the outset.

- **Data Collection and Preparation:**

In this phase, necessary data is gathered and prepared, including images of number plates and character samples, to create a robust dataset for model training.

- **Model Development:**

This stage involves configuring the dataset for the model by splitting data and setting up YAML files. It also includes managing resources and providing progress updates to maintain smooth development.

- **Character Segmentation and Recognition:**

This critical phase is focused on developing and refining the character recognition model. It involves segmenting characters from the number plates and training a CNN to recognize them accurately. Testing and fine-tuning are conducted to improve model accuracy.

- **Web Application Design:**

Here, the team designs the web application that will interface with the model, ensuring it provides a user-friendly and functional interface for users.

- **Integration and Testing:**

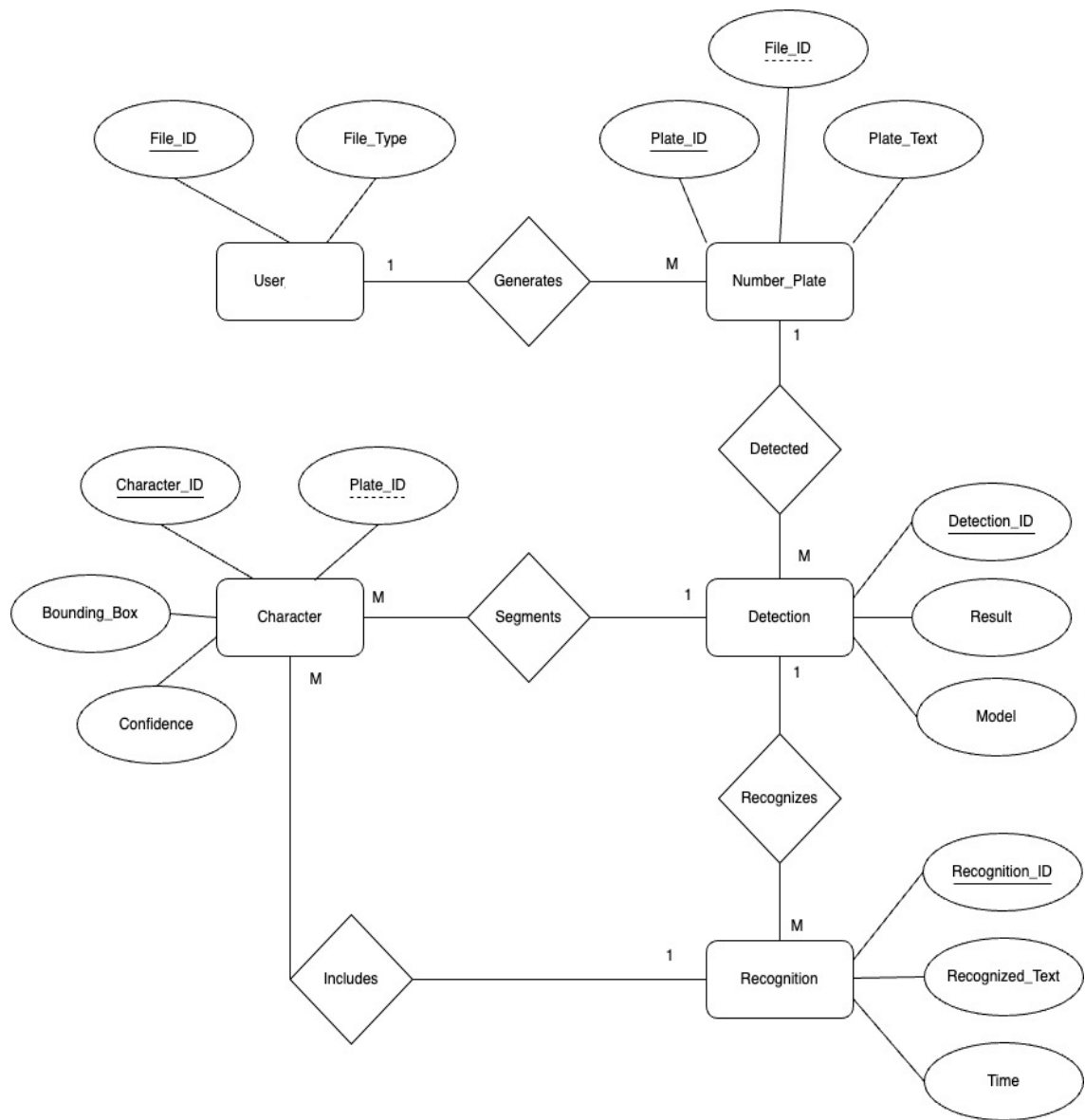
This phase combines all components—number plate detection, character recognition, and the web application—to test the end-to-end functionality of the system and ensure seamless operation.

- **Optimization and Fine-Tuning:**

Once integrated, the system undergoes optimization to enhance performance and address any minor issues, ensuring the final product runs efficiently.

3.1.3 Analysis – Structured Approach

I. Data Modelling using ER diagram



The Result entity, with a 'Result_ID', holds the compiled data of recognized characters. It's linked to a specific user and includes the 'Compiled_Data' (the final recognized value) and 'Processed_Time' (when the result was created). This result is then stored in the Recognized Plate Database.

The Recognized Plate Database is like a treasure chest for all processed data. It has some cool attributes like 'Database_ID' (a unique ID), 'Plate_Data' (the final compiled plate info), and 'Storage_Date' (when the data was stored). This way, we can easily find and retrieve the data later.

The Web Application is like the bridge between the system and the user. It pulls results from the database and shows them to the user. The application also keeps in touch with the user, so they can access their processed data and interact with the system without any hassle.

In a nutshell, this entire system shows how data flows from the user's input to the final display of results. We keep everything organized and connected so that processing and data storage are smooth and efficient.

II. Process modelling using DFD

- **DFD Level 0**

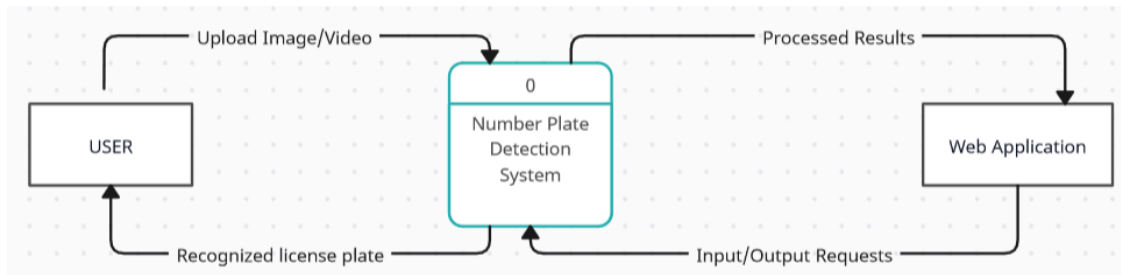


Figure 3.4 DFD Level 0

- **DFD Level 1**

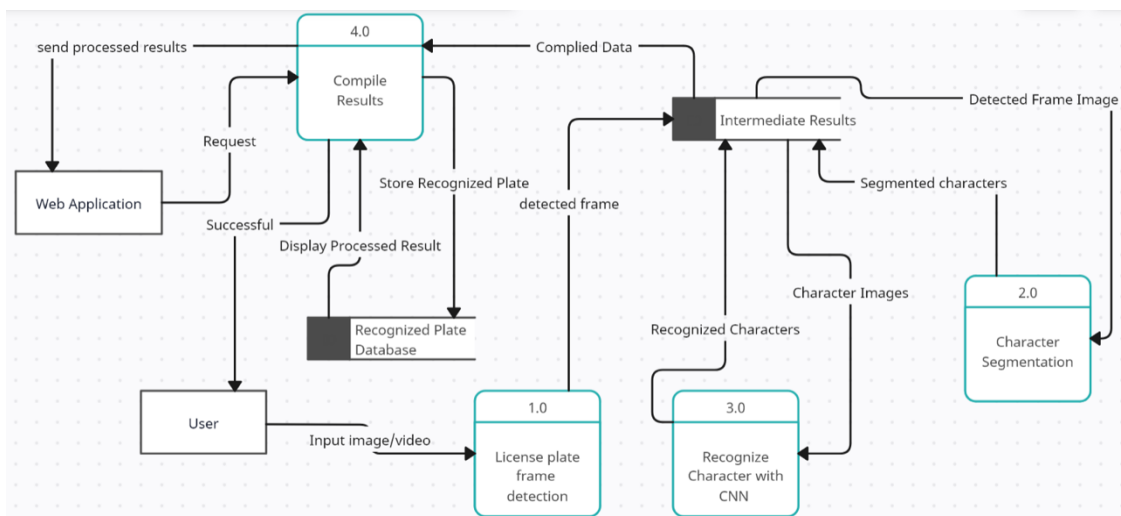


Figure 3.5 Process Modelling using DFD

- **User Input**

The process starts with the 'User', who provides an 'Input image or video' containing a vehicle's license plate. This input is then sent to the system for processing.

- **License Plate Frame Detection (1.0)**

The License Plate Frame Detection module processes the input to detect and isolate the license plate within each frame of the image or video. This step outputs a 'detected frame', focusing on the license plate for further analysis.

- **Character Segmentation (2.0)**

In this stage, the 'Character Segmentation' module takes the detected license plate frame and breaks it down into individual 'character images.' This segmentation is essential for separating each character, preparing them for accurate recognition in the next stage. The segmented characters are saved as intermediate results.

- **Recognize Character with CNN (3.0)**

The segmented character images are fed into the 'Character Recognition' module powered by a CNN. This module analyzes each character image to identify the specific character. The 'recognized characters' are saved as intermediate results, and the 'character data' is passed to the next stage.

- **Compile Results (4.0)**

The 'Compile Results' module gathers the recognized characters and compiles them into a coherent 'character data' string representing the entire license plate number. This compiled data is then stored in the 'Recognized Plate Database' for record-keeping and future use.

- **Recognized Plate Database**

The 'Recognized Plate Database' maintains a record of previously recognized license plates, allowing for easy data retrieval. It serves as a backup or historical repository of license plate recognition data.

- **Web Application and User Output**

The final 'compiled results' are sent to the 'Web Application', where they are displayed to the 'User'. The user can view the processed license plate data on the application interface, completing the workflow.

CHAPTER 4

SYSTEM DESIGN

4.1 Design - Structured approach

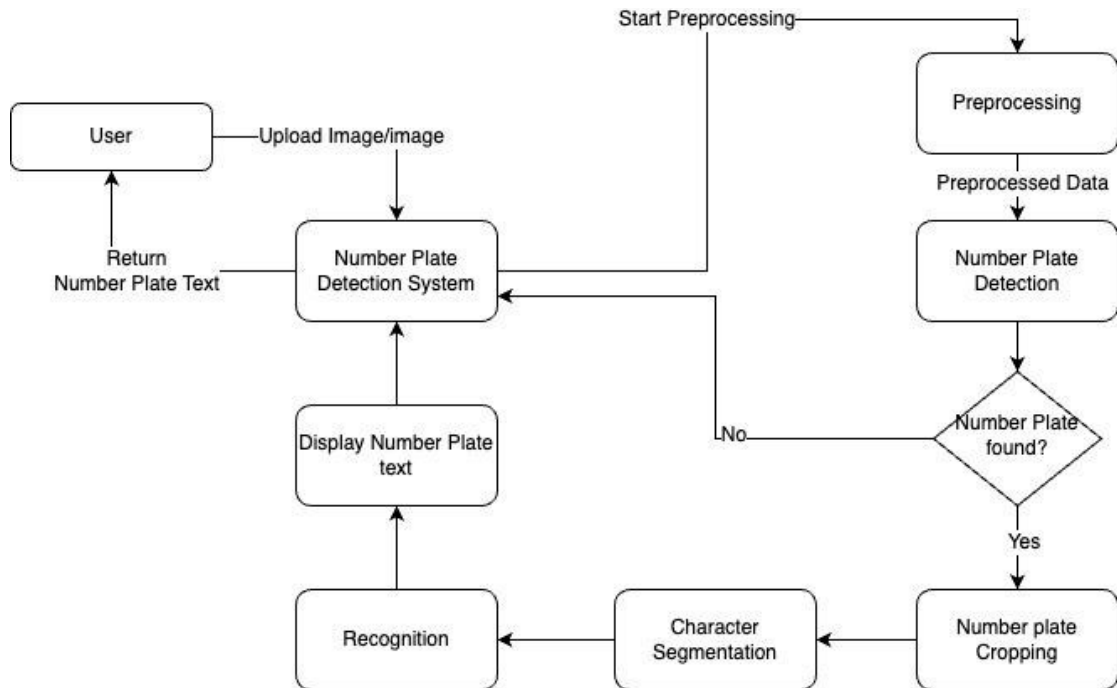


Figure 4.1 System Design

This system design figure represents an ANPR system, using a YOLO model for number plate detection and segmentation, and a CNN model for character recognition. Let's break it down step by step:

1. Input:

The system begins with the User uploading an image or a video frame containing a vehicle.

2. Preprocessing:

The uploaded image undergoes preprocessing to enhance its quality and make it suitable for further processing. Steps may include resizing, noise reduction, grayscale conversion, or histogram equalization.

3. Number Plate Detection:

The preprocessed data is fed into the Number Plate Detection System. YOLO, a deep learning model, is used here to detect the location of the number plate in the image. YOLO's efficiency allows for real-time object detection, making it ideal for this task.

Decision Point: If a number plate is detected, the system proceeds to the next step. If no number plate is detected, the process terminates, returning a "Number Plate Not Found" status.

4. Number Plate Cropping:

When a number plate is detected, the bounding box coordinates provided by YOLO are used to crop the region containing the number plate from the image.

5. Character Segmentation:

The cropped image (containing the number plate) is processed for character segmentation. Techniques like thresholding or contour detection isolate individual characters from the plate for recognition.

6. Character Recognition:

Each segmented character is fed into a CNN model for recognition. The CNN model is trained to classify alphanumeric characters typically found on number plates.

7. Output:

The recognized characters are combined to reconstruct the number plate text. The extracted text is then displayed to the user and returned as the output.

4.2 Algorithm Details

The system employs a comprehensive algorithm for detecting number plates and recognizing their characters from images or videos. The workflow comprises preprocessing, detection utilizing YOLOv8, character segmentation, and classification employing CNN. The following provides a mathematically detailed description of the system's workflow:

4.2.1 Number Plate Detection Using YOLOv8

YOLOv8 predicts bounding boxes for detected objects (number plates). The bounding box parameters are represented as

$$B = \{(x, y, w, h, c)\} \quad \dots(1)$$

Where:

x, y : Center coordinates of the bounding box.

w, h : Width and height of the bounding box.

c : Confidence score of the prediction.

a) Bounding Box Filtering: The confidence score measures the likelihood that the detected region contains a number plate. By applying a threshold T_c we eliminate low-confidence predictions:

$$B_{\text{filtered}} = \{B_i \mid c_i \geq T_c\} \quad \dots(2)$$

Where T_c is the confidence threshold ($T_c = 0.6$).

For each detected number plate ($B_i = (x, y, w, h)$), the cropping region in the image is defined as:

$$\text{Cropped Region} = I[y_{\min} : y_{\max}, x_{\min} : x_{\max}] \quad \dots(3)$$

Where

$$x_{\min} = x - \frac{w}{2}, x_{\max} = x + \frac{w}{2}, y_{\min} = y - \frac{h}{2}, y_{\max} = y + \frac{h}{2}$$

This ensures that subsequent processing focuses on the number plate, reducing computational overhead.

4.2.2 Character Segmentation Using YOLOv8

Once the number plate is cropped, another YOLOv8 model identifies characters within the region. Similar to number plate detection, bounding boxes (C) are predicted for individual characters.

$$C = \{(x_c, y_c, w_c, h_c, c_c)\} \quad \dots(4)$$

Where

x_c, y_c : Center coordinates of character boxes.

w_c, h_c : Width and height of character boxes.

c_c : Confidence score of detection.

a) Confidence Filtering: Only character detections with confidence c_c above T_{char} are considered. This reduces noise and prevents incorrect predictions:

$$C_{\text{filtered}} = \{C_j \mid c_{cj} \geq T_{\text{char}}\} \quad \dots(5)$$

Where T_{char} is the character confidence threshold ($T_{\text{char}} = 70$).

4.2.3 Character Sorting

Characters are sorted into rows and columns based on their (y_c) and (x_c) coordinates:

- Define a row threshold Δy .
- Characters are grouped into rows:

$$Row_i = \{C_k \mid |y_c - \mu_{yi}| \leq \Delta y\} \quad \dots(6)$$

Where μ_{yi} is the mean y_c of characters in row i .

Within each row, characters are sorted by x_c :

$$Row_i = \text{sort}(Row_i, \text{key} = x_c) \quad \dots(7)$$

The sorted characters form the final text.

4.2.4 Character Recognition Using CNN

Each detected character is passed through a CNN for classification.

- a) **Feature Extraction:** Convolutional layers extract hierarchical features from the input image. For a character image X , the feature map f is computed as:

$$\text{Feature Map}, f = \sigma(W_f \cdot X + b_f) \quad \dots(8)$$

Where:

X : Input image tensor.

W_f, b_f : Weights and biases of the convolutional layer.

σ : Activation function (ReLU).

- b) **Classification:** The extracted features are passed through dense layers to predict the character class:

$$y = \text{Softmax}(W_c \cdot f + b_c) \quad \dots(9)$$

Where W_c, b_c are the weights and biases for the dense (fully connected) layer.

c) **Prediction:** The predicted class \hat{y} and confidence score are:

$$\hat{y} = \text{argmax}(y) \quad \dots(10)$$

The confidence is:

$$\text{Confidence} = \max(y) \quad \dots(11)$$

4.2.5 Text Reconstruction

Once the characters are recognized, they are concatenated to form the detected text:

$$\text{Detected Text} = \text{Concatenate} (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) \quad \dots(12)$$

Where \hat{y}_i are the predicted characters in sorted order.

4.2.6 Video Frame Processing

For video inputs, the system processes each frame independently and aggregates the results. If F_t is the t -th frame, the detected texts are:

$$\text{Texts}_{\text{video}} = \bigcup_{t=1}^T (\text{Detected Text} F_t) \quad \dots(13)$$

This allows for robust recognition across frames, compensating for variations in lighting, motion blur, or occlusions.

4.2.7 Activation Function:

- **ReLU Activation Function:**

ReLU activation function is pivotal in deep learning models due to its simplicity and efficiency.

The equation for ReLU is as follows:

Equation:

$$f(x) = \max(0, x) \quad \dots(14)$$

Range: $[0, \infty)$

In the above code, although ReLU is not directly implemented, the underlying principles are utilized when working with neural networks, such as TensorFlow models for classification tasks. ReLU helps in introducing non-linearity, enabling the neural network to learn complex patterns. This is evident in the `classification_model`, where ReLU enhances the ability of the CNN to recognize distinct characters on the number plate accurately.

- **Softmax Activation Function:**

The Softmax activation function is essential for multi-class classification tasks, converting the output layer's logits into probabilities.

Equation:

Here,

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \dots(15)$$

In the code above, the TensorFlow Softmax function is used to predict the likelihood of each class during character recognition. By utilizing Softmax, the `predict_character` function converts raw predictions into interpretable probabilities, enabling the selection of the most likely character class.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Tools Used

I. Programming Language:

Python was the perfect choice because of its many useful tools for machine learning, computer vision, and building web apps. It made it easy to combine different parts, like detection models, recognition models, and the web interface.

II. Frameworks:

- **FastAPI:**

A modern, high-performance web framework used to develop the API endpoints for the project. It ensures smooth interaction between the user and the backend, handling file uploads, processing requests, and displaying results dynamically.

- **TensorFlow:**

TensorFlow is a deep learning framework that helps you train and use the CNN model for character recognition. It has tools for preprocessing data, building models, and making predictions quickly.

- **OpenCV:**

A computer vision library designed for image processing tasks, including resizing, cropping, and annotating detected number plates and characters.

III. Database: MySQL

A relational database was employed to store and retrieve metadata pertaining to detections. This database ensures efficient data organization and facilitates straightforward retrieval of past results.

5.1.2 Implementation Details

Number Plate Detection Using YOLOv8

The YOLOv8 model, a state-of-the-art object detection framework, was utilized to detect vehicle number plates in uploaded images or video frames.

Steps:

- **Model Training:** Pre-trained YOLOv8 weights were fine-tuned on a custom dataset consisting of annotated images of vehicle number plates.
- **Detection Process:** The model processes the uploaded media and identifies regions containing number plates, generating bounding boxes around detected plates.
- **Confidence Filtering:** Only bounding boxes with a confidence score above a set threshold (e.g., 0.6) are retained, reducing false positives.
- **Cropping:** Detected number plate regions are cropped from the input image for subsequent processing.

Character Segmentation Using YOLOv8

A separate YOLOv8 model was employed to segment individual characters from the cropped number plate images.

Steps:

- **Segmentation Model:** The cropped number plate images are passed through a character segmentation YOLOv8 model.
- **Bounding Box Generation:** Bounding boxes for each character on the number plate are identified.
- **Confidence Thresholding:** Characters with a detection confidence below 0.7 are filtered out.
- **Character Extraction:** The bounding boxes are used to extract individual characters, preparing them for recognition.

Character Recognition Using a CNN Model

A CNN was developed to classify alphanumeric characters extracted from segmented number plate regions.

Steps:

- **Image Resizing:** Each character image is resized to a standard dimension (e.g., 32x32 pixels) to ensure consistent input for the CNN model.
- **Classification:** The CNN processes the resized character images, predicting the class (0–9, A–Z) along with a confidence score.
- **Sorting:** Recognized characters are spatially sorted to reconstruct the correct order of the number plate text.

Web Integration Using FastAPI

The FastAPI framework was used to develop a web interface that facilitates user interaction with the system.

Features:

- **Image Upload Endpoint:** Users can upload images or video frames for processing.
- **Detection Endpoint:** This endpoint manages the complete workflow—detecting number plates, segmenting characters, recognizing text, and returning annotated images with results.
- **Result History:** A dedicated endpoint displays previously stored detection results in a tabular format, including links to processed images.

Database Integration

A MySQL database was implemented to store detection results and metadata securely.

Key Features:

- **Storage:** Results, including recognized text, image paths, and timestamps, are saved in a relational database.
- **Retrieval:** The database facilitates efficient retrieval of historical data for display on the web interface.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit testing is a process where individual software components or modules are tested to ensure they function as intended. The primary goal is to verify that each piece of code performs as expected. During the coding phase, each module was tested independently to confirm its proper functionality. Any bugs or issues identified during unit testing were resolved through debugging. Below is a summary of some of the test cases conducted during this phase.

Table 5.1: Unit Test Case: URL

S.N.	Test Inputs	Expected Outputs	Actual Outputs	Result
1.	To verify the Home Page's URL response	200 response	200 response	PASS
2.	To verify the Upload Page's URL response	200 response	200 response	PASS
3.	To verify the Result Page's URL response	200 response	200 response	PASS
4.	To verify the About Page's URL response	200 response	200 response	PASS


Table 5.2: Unit Test Case for Uploading, Detecting, Segmenting, Saving

S.N.	Test Inputs	Expected Outputs	Actual Outputs	Result
1.	To confirm that the picture is uploaded.	Successfully uploaded.	Successfully uploaded.	PASS
2.	To confirm that the number plate is detected.	Successfully detected.	Successfully detected.	PASS
3.	To confirm that the number is segmented.	Successfully segmented.	Successfully segmented.	PASS
4.	To confirm that the segmented characters are saved.	Successfully saved.	Successfully saved.	PASS

Table 5.3: Unit Test Case for System

S.N.	Test Inputs	Expected Outputs	Actual Outputs	Result
1.	To verify the system can manage errors.	Successfully Handle errors.	Successfully Handle errors.	PASS
2.	To verify system processes images	Processed images.	Processed images.	PASS
3.	To verify result storage in MySQL database	Successfully stored.	Successfully stored.	PASS

Table 5.4: Unit Test Case for Output

S.N.	Test Inputs	Expected Outputs	Actual Outputs	Result
1.		BAB0147	BAB0147	PASS
2.		BAD5877	BAD5877	PASS
3.		BAD7466	BAD7466	PASS

5.3 Result Analysis

Based on the performance metrics and results obtained from training your CNN model for character recognition at both 20 epochs and 40 epochs, I have created a detailed result analysis. This analysis is modeled after the referenced bird species identification report's structured approach.

5.3.1 Model Performance Overview

The character recognition CNN was trained for two configurations:

- **20 Epochs:** Focused on faster training with moderate tuning to evaluate baseline performance.
- **40 Epochs:** Utilized additional training to evaluate model performance with extended optimization.

The primary evaluation metrics used include:

- **Accuracy:** Training and validation accuracy progression.
- **Loss:** Training and validation loss trends.
- **Hyperparameter Analysis:** Analysis of learning rate, regularization, and network design.

5.3.2 Performance Metrics Comparison

- **Training for 20 Epochs**
 - **Training Accuracy:** The model achieved a steady growth in training accuracy, reaching above 90% by the end of 20 epochs.
 - **Validation Accuracy:** Validation accuracy fluctuated but stabilized around 85–90%, showing signs of overfitting as the validation loss was relatively higher than the training loss.
 - **Loss Trend:** The training loss reduced significantly, but the validation loss curve suggested a potential mismatch between model capacity and data generalization.

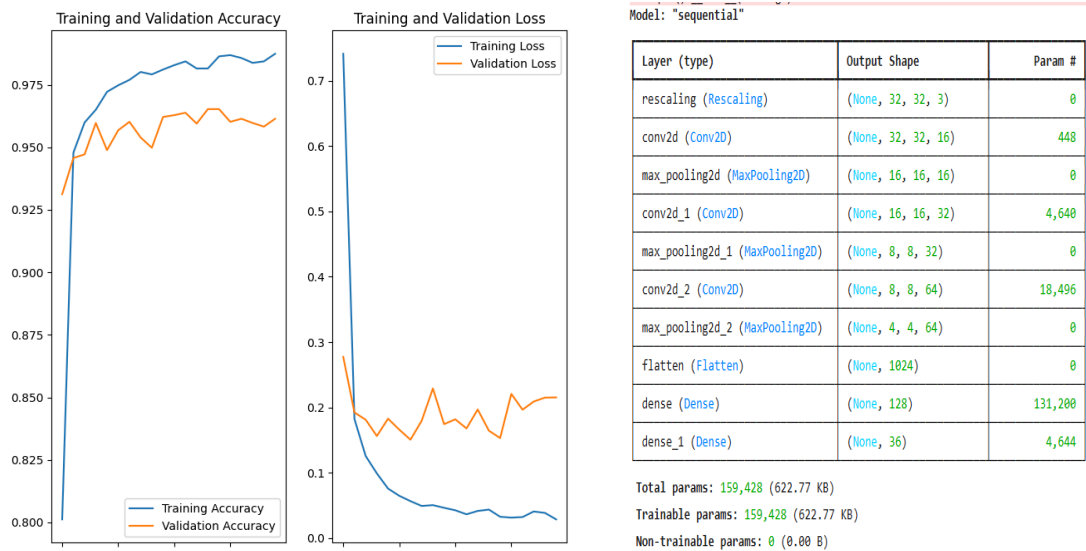


Figure 5.1 Graphical Representation of training for 20 epoch

- **Training for 40 Epochs**

- **Training Accuracy:** Further improvement was observed, with training accuracy exceeding 92% by epoch 40.
- **Validation Accuracy:** Validation accuracy improved significantly, stabilizing between 90–92%, indicating better generalization than the 20-epoch model.
- **Loss Trend:** Both training and validation losses showed a decreasing trend, with the gap between them narrowing, reflecting better model optimization and regularization.

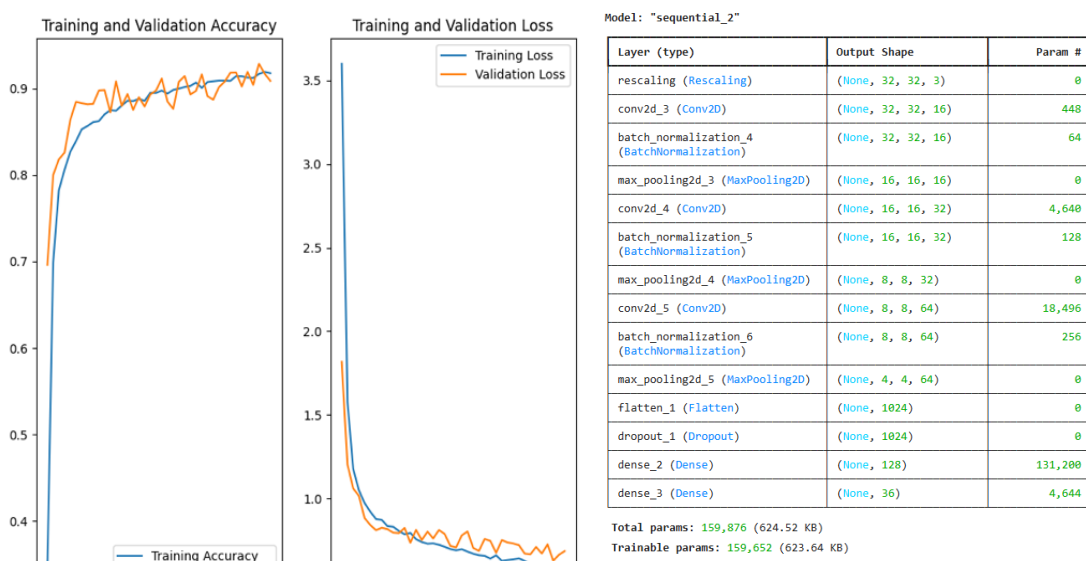


Figure 5.2 Graphical Representation of training for 40 epoch

5.3.3 Hyperparameter Tuning Analysis

Hyperparameter adjustments played a significant role in improving model performance:

- **Learning Rate:** A learning rate of 0.005 provided the best results in balancing convergence speed and accuracy.
- **Regularization Strength:** An L2 regularization of 0.001 effectively mitigated overfitting while preserving model accuracy.
- **Padding and Stride:** The use of valid padding and a stride of 2 resulted in optimal feature extraction, as demonstrated by improved validation accuracy and reduced loss.

Table 5.5: Result and Analysis

Metric	Value	Remarks
Confusion Matrix	High precision and recall across most characters.	Confusion mainly observed with characters like “O” and “0”
Hyperparameter Tuning	Learning Rate: 0.0005, Regularization: 0.001.	Provided optimal results for accuracy and generalization.
Classification Report	Accuracy: 92% Precision: 91% Recall: 90%	Indicates robust and reliable character recognition.
System Accuracy	90%	Reflects strong end-to-end system performance

5.3.4 Data Augmentation and Preprocessing

The inclusion of data augmentation techniques, such as random flipping, rotation, and zoom, improved model robustness by simulating varied data distributions. This led to:

- Reduced overfitting, as reflected in better alignment between training and validation accuracy.
- Enhanced ability to generalize to unseen data, as evident from the validation metrics.

5.3.5 Model Architecture Refinements

The improved architecture incorporated:

- **Batch Normalization:** Stabilized learning and improved convergence speed.
- **Three Convolutional Layers:** Enhanced character-specific trait extraction.

5.3.6 Confusion Matrix

The confusion matrix evaluates the performance of our character segmentation model, normalized to show proportions rather than absolute counts. It considers two classes: character (foreground) and background. The model achieves a true positive rate of 0.99 for characters and 1.00 for background, with only 1% of character pixels misclassified as background. While the performance is excellent, especially for background classification, minor misclassifications in character pixels suggest potential challenges with edges or shapes. Addressing these could involve refining preprocessing, enhancing data diversity, or fine-tuning the model to further improve accuracy.

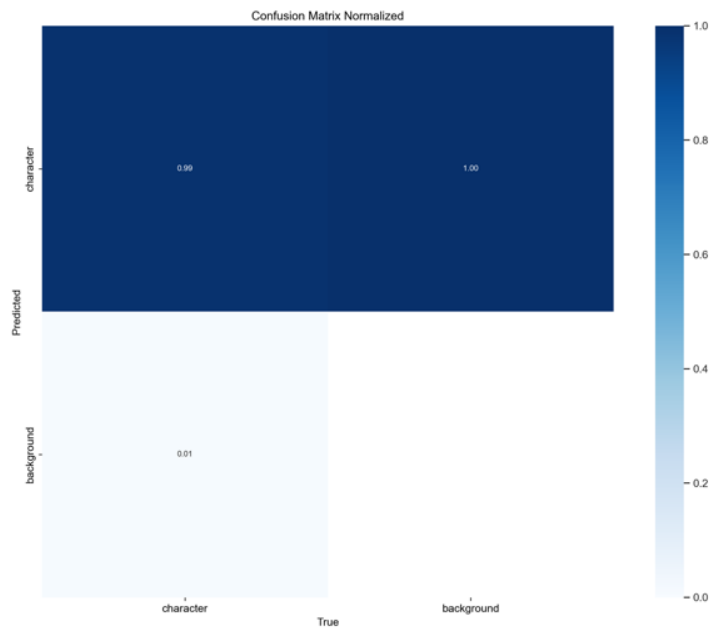


Figure 5.3. Confusion Matrix of Segmentation

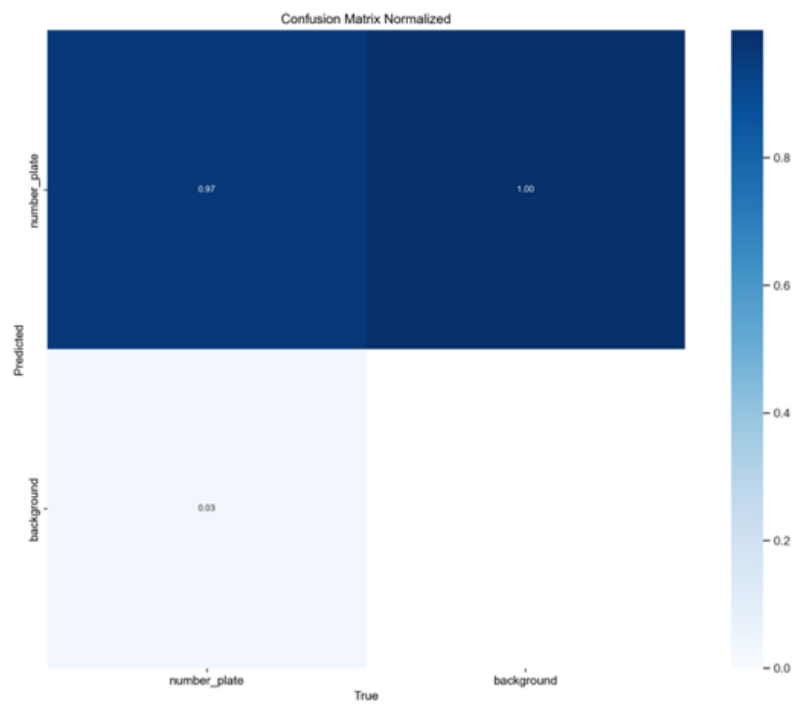


Figure 5.4 Confusion Matrix of Number Plate

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion

The Vehicle Number Plate Detection and Character Recognition system is a practical and efficient tool for automating license plate recognition. By using YOLO for accurate plate detection and a CNN for reliable character recognition, the system delivers impressive results. In ideal conditions—such as good lighting, clear visibility, and standard angles—it achieves an accuracy of 92%. While tougher situations like dim lighting, odd angles, or unusual plate designs may slightly affect its performance, the system still proves dependable. With its easy-to-use web interface, built using FastAPI, users can quickly upload images or videos and get real-time results. The system also securely stores all detected data in a MySQL database, making it simple to access whenever needed. Thorough testing shows the system performs exceptionally well, with high precision and recall in recognizing characters when conditions are favorable. Looking ahead, enhancements like real-time video processing, support for different regional plate styles, and better handling of difficult inputs will make it even more versatile. This project marks a big step forward in using AI to make transportation systems smarter and safer, offering valuable support for traffic management, parking solutions, and law enforcement.

6.2 Future Recommendation

- **Video Processing for Real-Time Recognition**

The system currently supports only still image processing. Adding functionality for video streams will enable real-time frame-by-frame detection and recognition, ensuring smooth operation for continuous input.

- **Error Handling and Confidence Thresholding**

The system requires enhanced error-handling mechanisms for situations like unclear plates, poor angles, or challenging lighting. Adding confidence thresholds and fallback options will help manage such cases better.

REFERENCE

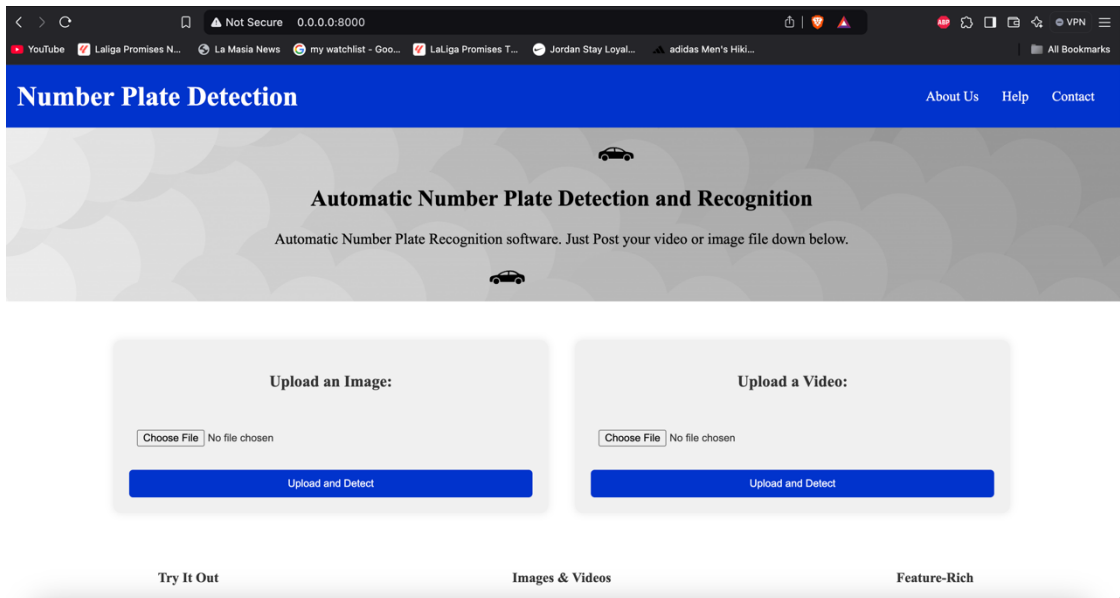
- [1] C. Patel, "Automatic number plate recognition system (anpr): A survey," *International Journal of Computer Applications*, vol. 69, 2013.
- [2] O. Salim, "Study for Food Recognition System Using Deep Learning," *Journal of Physics: Conference Series*, 2021.
- [3] R. Ibrahim, "Embedded System for Eye Blink Detection Using Machine Learning Technique," *1st Babylon International Conference on Information Technology an*, pp. 58-62, 2021.
- [4] S. Al-Shemarry, S. Abdulla, "An efficient texture descriptor for the detection of license plates from vehicle images in difficult conditions," *IEEE transactions on intelligent transportation systems*, vol. 21, pp. 553-564, 2019.
- [5] Badr A, Thabet A.M, "Automatic number plate recognition system," vol. 38, pp. 62-71, 2011.
- [6] D. Bouchain, "Character recognition using convolutional neural networks," *Inst. Neural Inf. Process.*, 2007.
- [7] J. P. Harvey, "Gpu acceleration of object classification algorithms using nvidia cuda," 2009.
- [8] Prof. A.B.Gadicha, "A Review paper on Vehicle Number Plate Recognition(VNPR) Using Improved Character Segmentation Method," *International Journal of Scientific and Research Publications*, vol. 3, no. 12, pp. 2250-3153, 2013.
- [9] M. M. Shidore, "Number Plate Recognition for Indian Vehicles," *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 2, 2011.

- [10] Prof.Pradnya Randive, "AUTOMATIC LICENSE PLATE RECOGNITION [ALPR]-A REVIEW PAPER," *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 1, 2016.
- [11] Y. Yuan, "A robust and efficient approach to license plate detection," *IEEE Transactions on Image Processing*, vol. 26, pp. 1102-1114, 2016.
- [12] M. Ibrahim, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Transactions on circuits and systems for video technology*, vol. 23, pp. 311-325, 2012.
- [13] Shen, "Reading car license plates using deep convolutional neural networks and LSTMs," *arXiv preprint arXiv:1601.05610*, 2016.
- [14] A. Hasan, "Machine Learning-based Diabetic Retinopathy Early Detection and Classification Systems-A Survey," *1st Babylon International Conference on IT and Science*, no. 16-21, 2021.

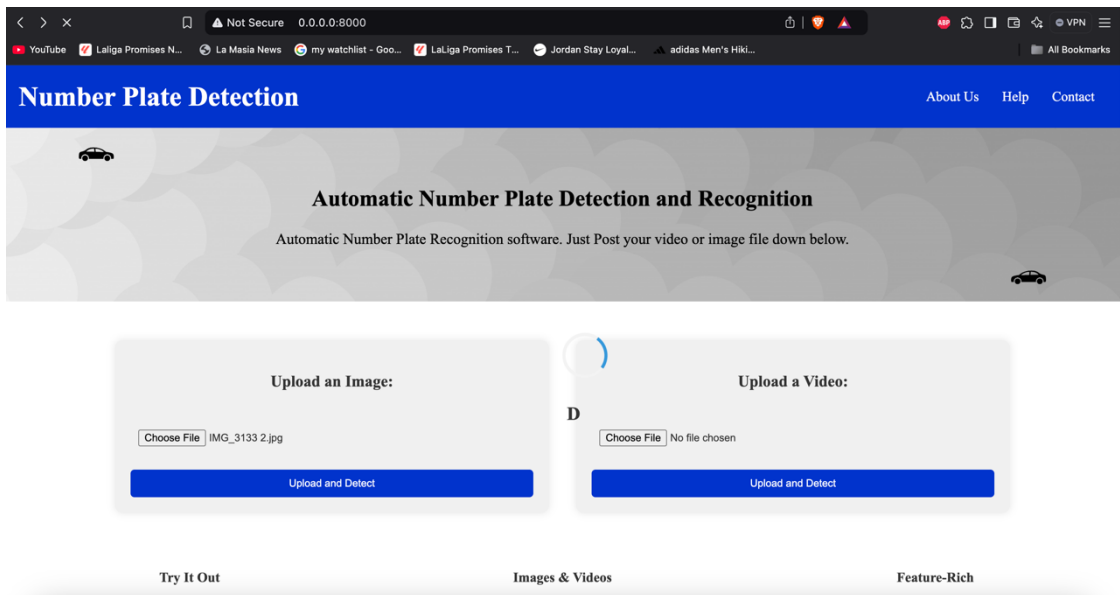
APPENDICES

Screenshots

a. Home Page



b. Image Detection loading screen



c. Output of character Recognition.

Number Plate Detection

HomeAbout usHelpContactDownload

Detection Results

Processed Image:



Recognized Characters:

3AB0147

d. Stored Detection Data

<>🔍⚠️ Not Secure0.0.0.0:8000/user🔒🛡️🚫🔒🛡️🚫VPN☰

📺 YouTube📌 LaLiga Promises N...🌐 La Masia News🕒 my watchlist - Goo...📌 LaLiga Promises T...🗣️ Jordan Stay Loyal...👤 adidas Men's Hiki...

• [Home](#)

Stored Detection Data:

ID	File Type	File Name	Recognized Texts	Detected Characters	Detection Time
1	image	1c29172c-05cb-4f00-968f-0b84c4ab0a8b.jpg	BAB0147	BAB0147	2024-12-07 19:57:06
2	image	a6d7f38a-bd1b-4b09-9167-3634ed9a7428.jpg	BAB0147	BAB0147	2024-12-07 19:58:35