AUTOMATIC NUMBER PLATE RECOGNITION FOR VEHICLE ACCESS CONTROL

REAL TIME PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the Degree

of

Bachelor of Technology

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

By

SHLESHITHA CHEEKATI	**22AG1A6676**
NAGASMITHA GARINE	**22AG1A6686**
NITHISHA SAMA	**22AG1A66C3**

Under the Esteemed Guidance of

Ms. Divya Jyothi

Assistant professor

Department of Computer Science and Engineering (AI&ML)

ACE ENGINEERING COLLEGE

An AUTONOMOUS Institution

NBA Accredited B. Tech Courses, Accorded NAAC "A" Grade &

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telanagana)

Ankushapur (V), Ghatkesar (M), Medchal-Malkajgiri Dist -501 301.

JULY 2024.

ACE

ENGINEERING COLLEGE

AN AUTONOMOUS INSTITUTION

Website: www.aceec.ac.in E-mail: info@aceec.ac.in

CERTIFICATE

This is to certify that the Real time project work entitled **“AUTOMATIC NUMBER PLATE RECOGNITION FOR VEHICLE ACCESS CONTROL”** is being submitted by **CH. Shleshitha (22AG1A6676), G. Nagasmitha(22AG1A6686), Nithisha(22AG1A66C3),** in partial fulfilment for the award of Degree of **BACHELOR OF THECHNOLOGY** in **COMPUTER SCIENCEAND ENGINEERING(AI&ML)** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2023-24 is a record of Bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Ms. Divya Jyothi	Dr S. Kavitha
Assistant professor	Assoc. Professor and
	Head Dept. of CSE(AI&ML)

ACKNOWLEDGEMENT

We would like to express our gratitude to all the people behind the screen who have helped us transform an idea into a real time application.

We would like to express our heart-felt gratitude to our parents without whom we would not have been privileged to achieve and fulfil our dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy**, for having founded such an esteemed institution. We are also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

We profoundly thank **Dr S. Kavitha**, Assoc Professor and Head of the Department of Computer Science and Engineering (AI&ML), who has been an excellent guide and also a great source of inspiration to our work.

We extremely thank **Mrs. J Bhargavi**, Assistant Professor, Project coordinator, who helped us in all the way in fulfilling of all aspects in completion of our Real Time Project.

We are very thankful to our internal guide**, Ms. Divya Jyothi,** Assistant Professor who has been an excellent and also given continuous support for the completion of our project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, we would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

CH. Shleshitha	**(22AG1A6676)**
G. Nagasmitha	**(22AG1A6686)**
S. Nithisha	**(22AG1A66C3)**

DECLARATION

This is to certify that the work reported in the present project titled "**AUTOMATIC NUMBER PLATE RECOGNITION FOR VEHICLE ACCESS CONTROL**" is a record work done by us in the Department of CSE (Artificial Intelligence & Machine Learning), ACE Engineering College. No part of the this is copied from books/journals/internet and whenever the portion is taken, the same has been duly referred in the text; the reported are based on the project work done entirely by us not copied from any other source.

CH. Shleshitha	**(22AG1A6676)**
G. Nagasmitha	**(22AG1A6686)**
S. Nithisha	**(22AG1A66C3)**

ABSTRACT

Automatic Number Plate Recognition (ANPR) is a powerful technology used for vehicle access control. It enables the automated detection and recognition of vehicle number plates to determine whether a vehicle is allowed to enter a secured area. This project utilizes ANPR to enhance security and streamline the vehicle entry process.

Our system employs a Flask web application to handle the number plate detection and recognition process. Images of vehicle number plates are captured and processed using Optical Character Recognition (OCR) to extract text. This text is then compared against a preloaded dataset to verify access permissions. The project integrates machine learning models and image processing techniques to ensure high accuracy and reliability.

The system's implementation involves several steps, including image acquisition, preprocessing, number plate detection, character segmentation, and recognition. The recognized number plates are cross-verified with an existing database to grant or deny access. This project addresses the need for efficient and accurate vehicle access control in various security-sensitive environments.

INDEX

LIST OF FIGURES

LIST OF NOTATIONS / ABBREVIATIONS

Abbreviation	Full Form
ANPR	Automatic Number Plate Recognition
OCR	Optical Character Recognition
ML	Machine Learning
DB	Database
API	Application Programming Interface
RAM	Random Access Memory
SSD	Solid State Drive

CHAPTER 1
INTRODUCTION

1.1 Introduction

Automatic Number Plate Recognition (ANPR) is a technology that uses optical character recognition to automatically read vehicle registration plates. It plays a crucial role in various applications such as law enforcement, toll collection, and access control. ANPR systems capture images of vehicle plates and use image processing and machine learning techniques to accurately identify the characters on the plate.

Vehicle access control is vital for securing premises and managing the flow of vehicles. It ensures that only authorized vehicles are granted entry, enhancing security and efficiency. ANPR-based access control systems automate this process, reducing the need for manual checks and improving the accuracy and speed of vehicle identification.

1.2 Existing system

Automatic Number Plate Recognition (ANPR) systems have been in use for several decades. These systems are employed in various applications, such as traffic management, toll collection, parking management, and security enforcement. ANPR systems work by capturing images of vehicle license plates, processing these images to extract the plate numbers, and then using this information for various purposes.

1.3 Proposed System

The proposed system aims to enhance vehicle access control by integrating ANPR with a database of authorized vehicles. This system is designed to improve accuracy and reliability, addressing some of the limitations of existing systems. Our system employs a Flask web application to handle the number plate detection and recognition process. Images of vehicle number plates are captured and processed using Optical Character Recognition (OCR) to extract text. This text is then compared against a preloaded dataset to verify access permissions. The project integrates machine learning models and image processing techniques to ensure high accuracy and reliability.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction to ANPR

Automatic Number Plate Recognition (ANPR) systems have become an essential tool in modern traffic management and security systems. ANPR technology uses optical character recognition (OCR) on images to read vehicle registration plates. It was first developed in the 1970s and has since evolved with advancements in image processing and machine learning.

2.2 Evolution of ANPR Systems

Early ANPR systems relied heavily on basic image processing techniques. With the advent of more sophisticated algorithms and computing power, the accuracy and speed of ANPR systems have improved significantly.

- **1980s-1990s:** Initial implementations using edge detection and morphological operations.
- **2000s:** Integration with digital cameras and improvements in OCR algorithms.
- **2010s:** Use of machine learning and deep learning to improve detection rates and handle various plate formats and conditions.

2.3 Components of ANPR Systems

An ANPR system typically consists of several components:

- **Image Acquisition:** Cameras capture images of vehicle plates.
- **Preprocessing:** Techniques like grayscale conversion, noise reduction, and edge detection prepare the image for plate detection.
- **Plate Detection:** Locates the number plate within the image using methods like contour detection and morphological operations.
- **Character Segmentation:** Isolates individual characters on the plate.
- **Character Recognition:** Uses OCR to read and convert characters into text.
- **Post-Processing:** Refines the detected text and removes any errors.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 **Hardware Requirements**

- Camera for capturing number plate images
- Server with sufficient RAM and SSD for processing
- Network infrastructure for data transmission

3.2 **Software Requirements**

- Python for development
- Flask for web application
- OpenCV for image processing
- Tesseract for OCR
- Database for storing authorized number plates

CHAPTER 4
SYSTEM ARCHITECTURE

4.1 Overview of System Components

The system architecture can be broadly divided into the following modules:

- ✓ Image Acquisition Module
- ✓ Image Preprocessing Module
- ✓ Number Plate Detection Module
- ✓ Character Segmentation Module
- ✓ Character Recognition Module
- ✓ Access Validation Module
- ✓ Web Application Interface

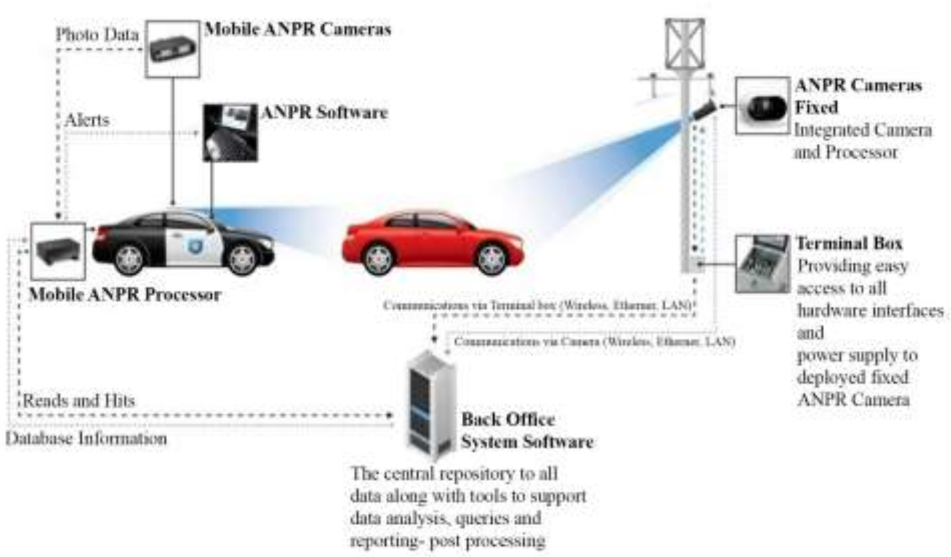


Photo Data
Mobile ANPR Cameras
Alerts
ANPR Software
Mobile ANPR Processor
ANPR Cameras Fixed
Integrated Camera and Processor
Terminal Box
Providing easy access to all hardware interfaces and power supply to deployed fixed ANPR Camera
Communications via Terminal box (Wireless, Ethernet, LAN)
Communications via Camera (Wireless, Ethernet, LAN)
Reads and Hits
Database Information
Back Office System Software
The central repository to all data along with tools to support data analysis, queries and reporting- post processing

Fig 4.1 Workflow for Vehicle Access Control

4.2 Modules

4.2.1 Image Acquisition Module

- **Camera Setup**: High-resolution cameras are placed at strategic points to capture images of vehicle number plates. The cameras should be capable of capturing images in various lighting conditions.

- **Image Capture**: Images are captured as vehicles approach the entry point. The captured images are then forwarded to the preprocessing module for further processing.

4.2.2 Image Preprocessing Module

- **Image Enhancement**: This step involves improving the quality of the captured images using techniques such as histogram equalization, noise reduction, and contrast adjustment.
- **Gray-Scale Conversion**: The colour images are converted to gray-scale to simplify the processing steps and reduce computational complexity.
- **Gaussian Blur**: A Gaussian blur is applied to the gray-scale images to reduce noise and enhance the edges of the number plates.

4.2.3 Number Plate Detection Module

- **Edge Detection**: The preprocessed images are subjected to edge detection using techniques such as the Sobel operator to highlight the edges of the number plates.
- **Thresholding**: Thresholding is applied to convert the gray-scale images into binary images, making it easier to identify the number plates.
- **Morphological Operations**: Morphological operations, including dilation and erosion, are used to enhance the detected edges and fill in any gaps.
- **Contour Detection**: Contours are detected in the binary images to locate the potential number plate regions. The system then identifies the contour that likely represents the number plate based on aspect ratio and size constraints.

4.2.4 Character Segmentation Module

- **Plate Region Extraction**: The detected number plate region is extracted from the image.
- **Character Isolation**: The extracted number plate region is further processed to isolate individual characters. This involves segmenting the characters based on spacing and connected components analysis.

4.2.5 Character Recognition Module

- **Optical Character Recognition (OCR)**: The segmented characters are passed through an OCR engine, such as Tesseract, to recognize and convert them into text.

- **Text Post-Processing**: The recognized text is post-processed to correct any recognition errors. This may include filtering out non-alphanumeric characters and converting all text to a standard format (e.g., uppercase).

4.2.6 Access Validation Module

- **Dataset Comparison**: The recognized number plate text is compared against a preloaded dataset of authorized vehicles. This dataset is sorted for efficient searching.
- **Search Algorithm**: A binary search algorithm is used to quickly determine if the recognized number plate is present in the authorized vehicle list.
- **Access Decision**: Based on the search result, an access decision is made. If the number plate is found in the dataset, access is granted; otherwise, access is denied.

4.2.7 Web Application Interface

- **Flask Web Application**: The entire system is integrated into a Flask web application that provides a user-friendly interface for interacting with the system.
- **File Upload**: Users can upload images of number plates through the web interface.
- **Result Display**: The application displays the recognition results, including the recognized number plate text and access decision.
- **Logging and Reporting**: The system logs all access attempts and can generate reports for further analysis.

4.3 Data Flow

- **Image Capture**: The camera captures an image of an approaching vehicle's number plate.
- **Preprocessing**: The captured image undergoes preprocessing to enhance its quality.
- **Detection**: The preprocessed image is analysed to detect the number plate region.
- **Segmentation**: The number plate region is segmented to isolate individual characters.
- **Recognition**: The segmented characters are recognized using OCR to extract the number plate text.
- **Validation**: The recognized text is validated against the authorized vehicle dataset.
- **Decision**: The system makes an access decision based on the validation result.
- **Output**: The result is displayed on the web interface and logged for future reference.

4.4 System Workflow

The following flowchart illustrates the detailed workflow of the ANPR system:

1. **Start**
2. **Capture Image**: The system captures an image of the vehicle's number plate.
3. **Preprocess Image**: The captured image is preprocessed to enhance quality.
4. **Detect Number Plate**: The system detects the number plate region in the preprocessed image.
5. **Segment Characters**: The detected number plate region is segmented to isolate characters.
6. **Recognize Characters**: The isolated characters are recognized and converted into text.
7. **Validate Number Plate**: The recognized text is validated against the authorized vehicle dataset.
8. **Access Decision**: The system makes an access decision based on the validation result.
    - **If authorized**: Grant access.
    - **If unauthorized**: Deny access.
9. **Display Result**: The result is displayed on the web interface.
10. **Log and Report**: The access attempt is logged for future reference.
11. **End**

4.5 Algorithms

□ Number Plate Detection:

- **Gaussian Blur:**
    - **Purpose:** Reduce noise in the image, making edge detection more accurate.
    - **Implementation:** cv2.GaussianBlur(img, (5, 5), 0)
- **Grayscale Conversion:**
    - **Purpose:** Convert the image to grayscale for easier processing.
    - **Implementation:** cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    - **Implementation:** cv2.threshold(img2, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
- **Morphological Transformations:**
    - **Purpose:** Enhance the binary image by closing small gaps and connecting disjointed parts of the number plate.

- ◦ **Implementation:** cv2.morphologyEx(src=img2, op=cv2.MORPH_CLOSE, kernel=element, dst=morph_img_threshold)

- **Contour Detection:**
    - **Purpose:** Identify potential regions of interest (number plates) in the binary image.
- **Plate Validation:**
    - **Purpose:** Validate the potential number plate based on its aspect ratio, area, and rotation angle.
    - **Functions:** ratioCheck, isMaxWhite, ratio_and_rotation
- **Character Recognition (OCR):**
    - **Purpose:** Recognize text from the detected number plate using Optical Character Recognition (OCR).
    - **Implementation:** pytesseract.image_to_string(plate_im, config=custom_config, lang='eng')

Sorting:

- **Quick Sort Algorithm:**
    - **Purpose:** Efficiently sort the list of allowed vehicles.
    - **Implementation:** The recursive quickSort function with partition function to rearrange elements based on a pivot.
    - **Key Functions:**
        - partition(arr, low, high): Partitions the array into two halves.
        - quickSort(arr, low, high): Recursively sorts the elements before and after the partition.

Searching:

- **Binary Search Algorithm:**
    - **Purpose:** Quickly search for a vehicle number in the sorted list of allowed vehicles.
    - **Implementation:** The recursive binarySearch function to find the index of the element.
    - **Key Function:**
        - binarySearch(arr, l, r, x): Recursively searches for the element x in the array arr between indices l and r.

CHAPTER 5
SOFTWARE DESIGN

5.1 Flowchart

A flowchart is a graphical representation of a process, showing the sequence of steps involved. For your ANPR-based vehicle access control project, the flowchart illustrates the steps from image upload to the final decision on whether the vehicle is allowed to enter

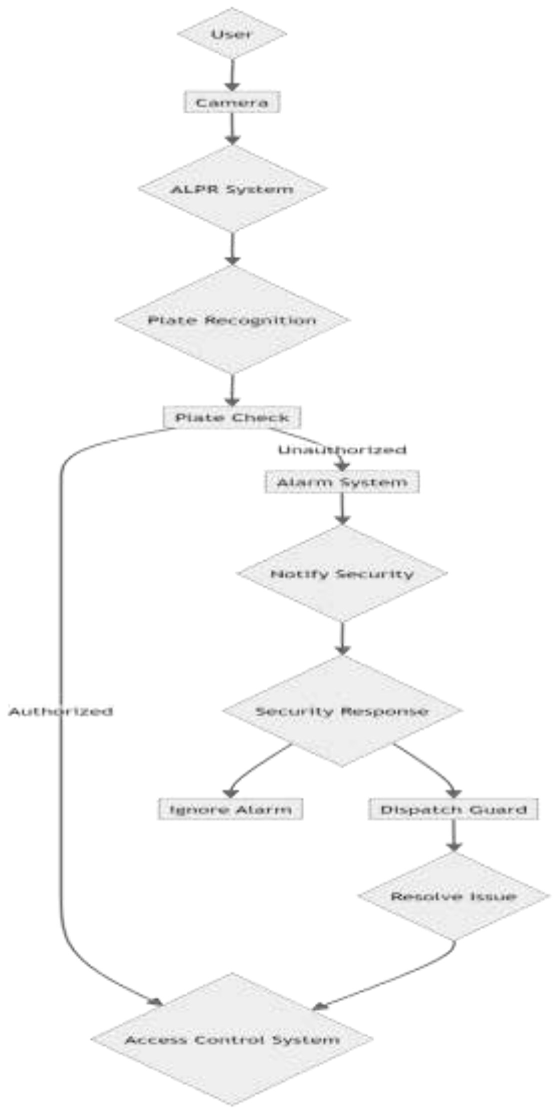


Fig 5.1 Flowchart

5.2 UML Diagrams

5.2.1 Class Diagram

A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that describes the structure of a system by showing its classes, their attributes, methods (or operations), and the relationships among objects. It is a fundamental part of object-oriented modeling and serves to represent the static view of an application.

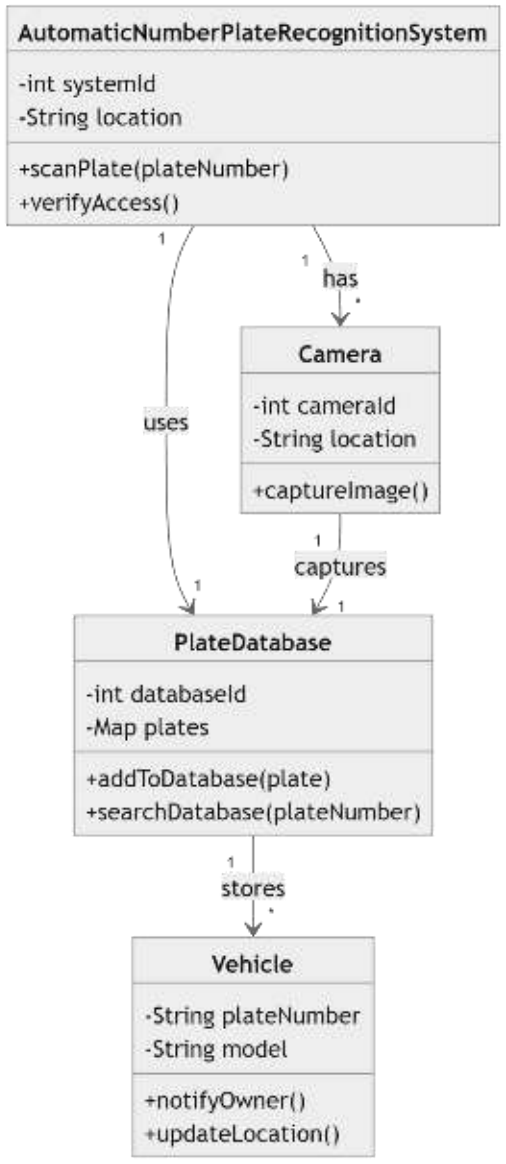


Fig 5.2 Class Diagram

5.2.2 Sequence Diagram

A sequence diagram is a type of interaction diagram in UML (Unified Modeling Language) that shows how objects interact in a particular sequence of events. It is used to represent the flow of messages, events, and actions between objects or components over time.

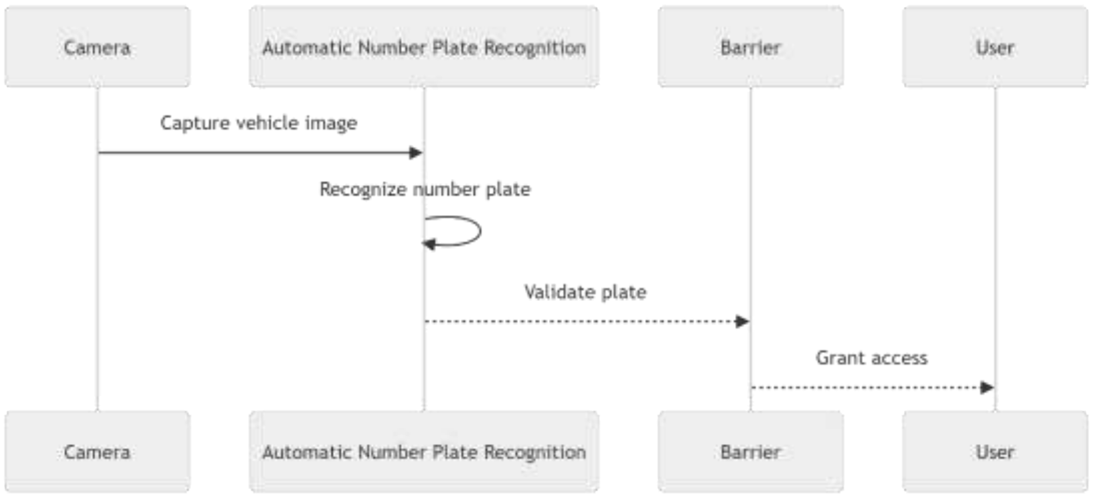


Fig. 5.3 Sequence Diagram

5.2.3 Use Case Diagram

A use case diagram is a type of behavioural diagram in UML (Unified Modeling Language) that captures the functional requirements of a system. It visually depicts the interactions between users (actors) and the system to achieve a specific goal (use cases). Use case diagrams are essential for understanding the functionality of a system from the user's perspective.

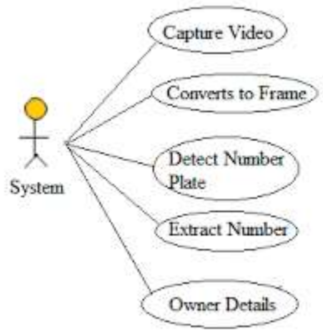


Fig. 5.4 Use Case Diagram

5.3 Database Design

5.3.1 E-R Diagrams

An Entity-Relationship (ER) diagram is a type of diagram used in database design to represent the structure of data. It illustrates the entities in a database and the relationships between those entities. ER diagrams help in understanding the data requirements and structuring the database logically before its physical implementation.

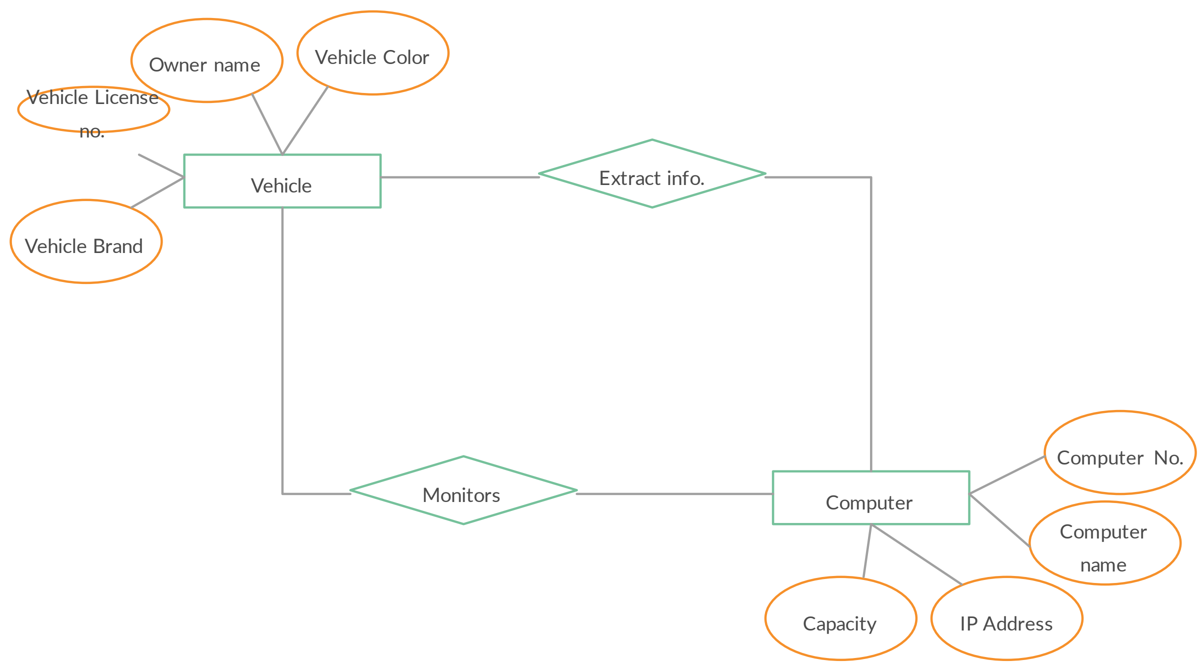


Fig. 5.5 E-R Diagram

CHAPTER 6
IMPLEMENTATION

6.1 FRONT END CODE

```
from flask import Flask, render_template, request, redirect, url_for
import os
import numpy as np
import cv2
from PIL import Image
import pytesseract
import re

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'static/uploads'
if not os.path.exists('static/uploads'):
    os.makedirs('static/uploads')

# Detecting number plate function
def number_plate_detection(img):
    def clean2_plate(plate):
        gray_img = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
        _, thresh = cv2.threshold(gray_img, 110, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        if contours:
            contour_area = [cv2.contourArea(c) for c in contours]
            max_cntr_index = np.argmax(contour_area)
            max_cnt = contours[max_cntr_index]
            max_cntArea = contour_area[max_cntr_index]
            x, y, w, h = cv2.boundingRect(max_cnt)
            if not ratioCheck(max_cntArea, w, h):
                return plate, None
            final_img = thresh[y:y + h, x:x + w]
            return final_img, [x, y, w, h]
        else:
            return plate, None
    def ratioCheck(area, width, height):
        ratio = float(width) / float(height)
        if ratio < 1:
            ratio = 1 / ratio
        if (area < 1063.62 or area > 73862.5) or (ratio < 2 or ratio > 6):
            return False
        return True

    def isMaxWhite(plate):
        avg = np.mean(plate)
        return avg >= 115

    def ratio_and_rotation(rect):
        (x, y), (width, height), rect_angle = rect
        if width > height:
            angle = -rect_angle
        else:
            angle = 90 + rect_angle
        if angle > 15:
            return False
        if height == 0 or width == 0:
            return False
        area = height * width
        return ratioCheck(area, width, height)

    img2 = cv2.GaussianBlur(img, (5, 5), 0)
    img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    img2 = cv2.Sobel(img2, cv2.CV_8U, 1, 0, ksize=3)
    _, img2 = cv2.threshold(img2, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    element = cv2.getStructuringElement(shape=cv2.MORPH_RECT, ksize=(17, 3))
    morph_img_threshold = img2.copy()
    cv2.morphologyEx(src=img2, op=cv2.MORPH_CLOSE, kernel=element,
dst=morph_img_threshold)
    contours, _ = cv2.findContours(morph_img_threshold,
mode=cv2.RETR_EXTERNAL, method=cv2.CHAIN_APPROX_NONE)
    for i, cnt in enumerate(contours):
        min_rect = cv2.minAreaRect(cnt)
        if ratio_and_rotation(min_rect):
            x, y, w, h = cv2.boundingRect(cnt)
            plate_img = img[y:y + h, x:x + w]
            if isMaxWhite(plate_img):
                clean_plate, rect = clean2_plate(plate_img)
                if rect:
                    x1, y1, w1, h1 = rect
                    x, y, w, h = x + x1, y + y1, w1, h1
                    plate_im = Image.fromarray(clean_plate)
                    # Configure Tesseract
                    custom_config = r'--oem 3 --psm 6'
                    text = pytesseract.image_to_string(plate_im,
config=custom_config, lang='eng')
                    return text.strip()
    return ""

# Sorting and searching functions
def partition(arr, low, high):
    i = (low - 1)
    pivot = arr[high]
    for j in range(low, high):
        if arr[j] < pivot:
            i = i + 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return (i + 1)

def quickSort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi - 1)
        quickSort(arr, pi + 1, high)
    return arr

def binarySearch(arr, l, r, x):
    if r >= l:
        mid = l + (r - l) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binarySearch(arr, l, mid - 1, x)
        else:
            return binarySearch(arr, mid + 1, r, x)
    else:
        return -1

# Preloaded dataset
allowed_vehicles = ["ABC1234", "XYZ5678", "DEF9012", "MH20EE7598",
"MH20EJ0365"]
allowed_vehicles = quickSort(allowed_vehicles, 0, len(allowed_vehicles) - 1)

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)
        if file:
            file_path = os.path.join(app.config['UPLOAD_FOLDER'],
file.filename)
            file.save(file_path)
            img = cv2.imread(file_path)
            number_plate = number_plate_detection(img)
            res2 = str("".join(re.split("[^a-zA-Z0-9]*", number_plate)))
            res2 = res2.upper()
            result = binarySearch(allowed_vehicles, 0, len(allowed_vehicles) -
1, res2)
            if result != -1:
                message = "The Vehicle is allowed to visit."
            else:
                message = "The Vehicle is not allowed to visit."
            return render_template('index.html', message=message,
number_plate=res2, image_file=file.filename)
    return render_template('index.html')
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

6.2 BACK-END CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Number Plate Detection</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            background-color: #f0f0f0;
        }

        h1 {
            margin: 0;
            padding: 60px 0;
            background-color: #0a3666;
            color: #cfe2f6;
            font-size: 2.5em;
            text-align: center;
            width: 100%;
        }

        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            margin-top: 20px;
```

```
}

input[type="file"] {
    margin-bottom: 10px;
}

.button-container {
    display: flex;
    justify-content: center;
}

button {
    margin: 5px;
    padding: 10px 20px;
    background-color: #0a3666;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1em;
}

button:hover {
    background-color: #072d4f;
}

.result {
    margin-top: 20px;
    padding: 20px;
    width: 80%;
    text-align: center;
}

.result p {
    margin: 10px 0;
    font-size: 1.2em;
    color: #0a3666; /* Highlight output */
    font-weight: bold; /* Highlight output */
```

```
        }

        .number-plate {
            color: #ff0000; /* Red color to highlight the number plate */
            font-size: 1.5em;
        }

        .image {
            max-width: 80%;
            height: auto;
            margin-top: 20px;
            border-radius: 4px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>
    <h1>Automatic Number Plate Recognition For Vehicle Access Control</h1>
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="file" accept=".jpg, .jpeg, .png">
        <div class="button-container">
            <button type="submit">Upload</button>
            <button type="button" onclick="clearForm()">Clear</button>
        </div>
    </form>
    {% if message %}
        <div class="result">
            <p>{{ message }}</p>
            <p>Detected Number Plate: <span class="number-plate">{{
number_plate }}</span></p>
        </div>
    {% endif %}
    {% if image_file %}
        <div id="imageContainer">
            <img src="{{ url_for('static', filename='uploads/' + image_file)
}}" class="image" alt="Uploaded Image">
        </div>
    {% endif %}
```

```
        <script>
            function clearForm() {
                var fileInput = document.querySelector('input[type="file"]');
                fileInput.value = '';  // Clear uploaded file
                var resultDiv = document.querySelector('.result');
                if (resultDiv) {
                    resultDiv.innerHTML = '';  // Clear result message
                    resultDiv.style.display = 'none';  // Hide result box
                }
                var imageContainer = document.getElementById('imageContainer');
                if (imageContainer) {
                    imageContainer.innerHTML = '';  // Clear image container
                }
            }
        </script>
    </body>
    </html>
```

CHAPTER 7
TESTING

7.1 Testing Methodologies

Testing involves both functional and non-functional testing to ensure the system meets its requirements.

7.2 Model Evaluation

Model evaluation metrics include accuracy, precision, recall, and F1 score to assess the performance of the OCR model.

7.3Testing Procedures

Testing procedures involve:

- Unit testing of individual modules.
- Integration testing to ensure modules work together.
- System testing to validate the entire system's functionality.

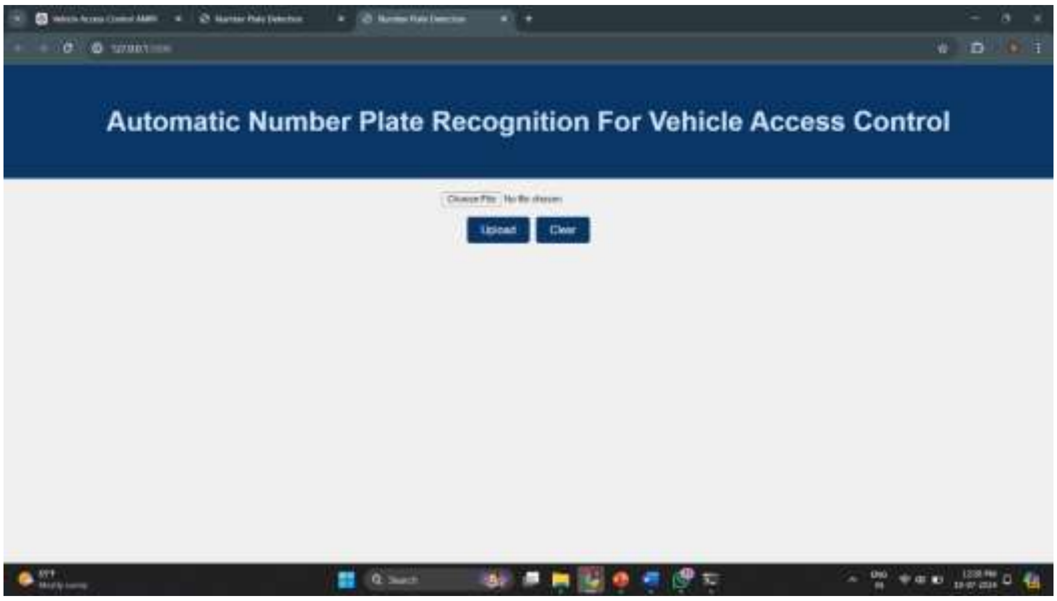
7.4 Testing Tools

Tools used for testing include:

- JUnit for unit testing.
- Selenium for automated UI testing.
- Custom scripts for performance testing.

CHAPTER 8
OUTPUT SCREENS

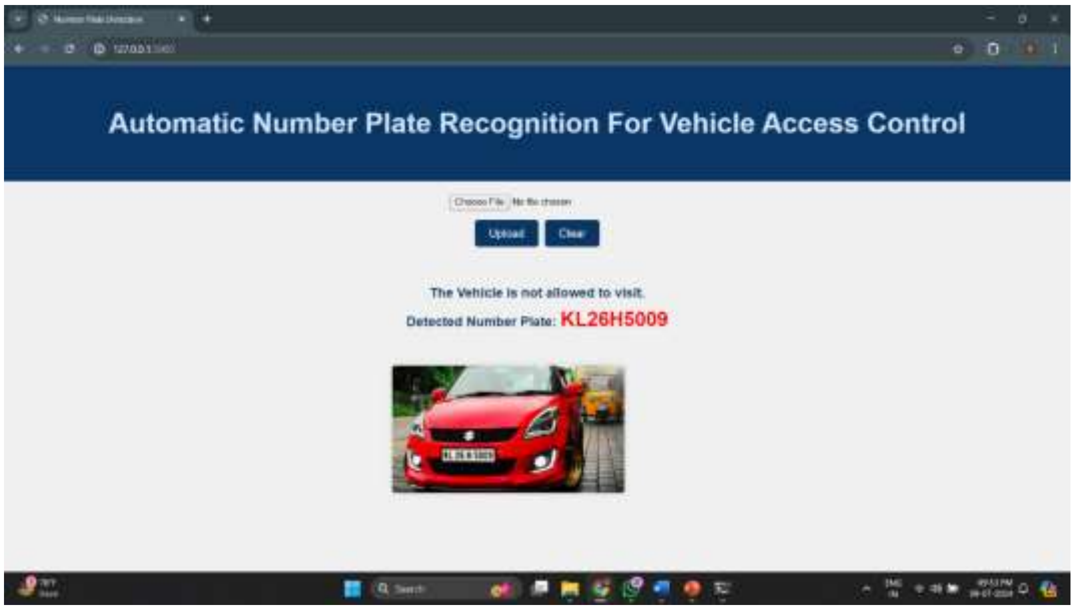
8.1 Home Screen



Automatic Number Plate Recognition For Vehicle Access Control

Fig 8.1 Home Screen

8.2 Output Screen



Automatic Number Plate Recognition For Vehicle Access Control

The Vehicle is not allowed to visit.

Detected Number Plate: KL26H5009

Fig 8.2 Output Screen

CHAPTER 9
CONCLUSION

9.1 Future Enhancements

Future enhancements could include:

- Incorporating advanced machine learning techniques for better accuracy.
- Enhancing the system to handle multiple languages and formats of number plates.
- Integrating with other security systems such as facial recognition.

9.2 Conclusion

The Vehicle Access Control System through ANPR provides a robust and efficient solution for automated vehicle entry management. By leveraging advanced machine learning and OCR techniques, the system ensures high accuracy and reliability, significantly enhancing security and operational efficiency.

CHAPTER 10
REFERENCES

- Website: https://docs.opencv.org
- Website: https://flask.palletsprojects.com
- Website: https://github.com/tesseract-ocr/tesseract
- Website: https://pillow.readthedocs.io
- Website: https://numpy.org/doc
- Z. Zheng, W. Yang, Q. Li, and H. Wang, "Automatic License Plate Recognition System," in Proceedings of the International Conference on Intelligent Transportation Systems, 2012.
- Book: Adrian Rosebrock, "Machine Learning for OpenCV: Intelligent Image Processing with Python," 2017.
- Book: David Beazley and Brian K. Jones, "Python Cookbook," 3rd Edition, O'Reilly Media, 2013.

CHAPTER 11
APPENDICES

A. Technical Specifications

A. Framework and Libraries Used:

- Flask: Web framework for backend development.
- OpenCV: Computer vision library for image processing tasks.
- NumPy: Numerical computing library for efficient array operations.
- Py-tesseract: Python wrapper for Google's Tesseract-OCR Engine.
- PIL (Pillow): Python Imaging Library for handling images.

B. Algorithm Overview:

- **Number Plate Detection:** Utilizes image processing techniques such as Gaussian Blur, Sobel edge detection, thresholding, and morphological operations to detect number plates within images.
- **Text Extraction:** Uses Tesseract OCR with custom configurations to extract alphanumeric characters from detected number plates.
- **Data Processing:** Implements quicksort for sorting the preloaded dataset of allowed vehicle number plates and binary search for efficient lookup.

B. System Architecture

- **Server-side Processing:** Flask application handles HTTP requests, processes uploaded images using OpenCV for number plate detection and text extraction.
- **Data Handling:** Preloaded dataset of allowed vehicle number plates is stored and processed in-memory.

C. Sample Input and Output

- Sample Input: Uploaded image containing a vehicle with a visible number plate.
- Output: Confirmation message indicating whether the vehicle is allowed based on dataset lookup.