Number Plate Recognition

A MINI PROJECT-I REPORT

Submitted by

MITHUNRAJ V	**621322106060**
BHARAT S	**621322106013**
GUNA D	**621322106032**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

ANNA UNIVERSITY :: CHENNAI 600 025

SEPTEMBER 2024

KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Project Report **"Number Plate Recognition"** is the bonafide work of **"MITHUNRAJ V (621322106060), BHARAT S (621322106013) GUNA D (621322106032)"** who carried out the 20EC605L Mini Project-I work under my supervision.

SIGNATURE
Dr.M.DHARMALINGAM, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Professor,
Department of Electronics and Communication Engineering,
Kongunadu College of Engineering and Technology (Autonomous),
Thottiam, Trichy-621215

SIGNATURE
Dr.K.AMUDHA, M.E., Ph.D.,
SUPERVISOR
Professor,
Department of Electronics and Communication Engineering,
Kongunadu College of Engineering and Technology (Autonomous),
Thottiam, Trichy-621215

Submitted for the project viva-voce examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

Namakkal-Trichy Main Road, Tholurpatti, Thottiam- Taluk, Trichy- Dist, Tamil Nadu -621 215

COLLEGE VISION & MISSION STATEMENT

VISION

"To become an Internationally Renowned Institution in Technical Education, Research and Development, by transforming the Students into Competent Professionals with Leadership Skills and Ethical Values"

MISSION

- Providing the Best Resources and Infrastructure
- Creating Learner - Centric Environment and continuous Learning
- Promoting Effective Links with Intellectuals and Industries
- Enriching Employability and Entrepreneurial Skills
- Adapting to Changes for Sustainable Development

KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

Namakkal-Trichy Main Road, Tholurpatti, Thottiam- Taluk, Trichy- Dist,

Tamil Nadu -621 215

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION

To create highly skilled, proficient and excellent Electronics and Communication Engineers having professional ethics, passion and competence to adapt to the latest transformations in technology.

MISSION

- Promoting quality teaching and effective learning to face the global challenges
- Enriching professionals of high caliber to excel in their careers through students' overall development
- Promoting education that imparts multidisciplinary design approaches, innovation and creativity

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Graduates shall exhibit the skills and knowledge required to design, develop and implement solutions for real life problems.

PEO2: Graduates shall excel in professional career, higher education and research.

PEO3: Graduates shall demonstrate professionalism, entrepreneurship, ethical behavior, communication skills and collaborative team work to adapt the emerging trends by engaging in lifelong learning.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Students shall have skills and knowledge to work and design on PCB, analog and digital systems, adhoc and sensor networks, embedded and communication systems.

PSO2: Students are able to perform and design in the simulation tools and IoT related modules.

PROGRAM OUTCOMES (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project involves creating an automated system for detecting vehicle number plates, using Optical Character Recognition (OCR) and integrating it with a web interface. The system enables users to upload images of vehicles through a Flask-powered web page. Once an image is uploaded, the system identifies the number plate and retrieves corresponding owner and fine information.

The system is built using Python, OpenCV, and Tesseract OCR to effectively detect and extract characters from the number plates in the images. These details are then cross-referenced with a local JSON file that holds the relevant owner information and any fines linked to the vehicle. Through the user-friendly Flask interface, users can easily view the owner's name, contact information, and a breakdown of any fines.

Designed for simplicity and efficiency, this project offers an effective way to retrieve vehicle and fine details from uploaded images. It also has potential for future upgrades, such as adding real-time functionality or integrating more extensive datasets.

ACKNOWLEDGEMENT

The satisfaction that ones on completing a project cannot be fully enjoyed without mentioning the people who made it possible. We express our sincere thanks and our grateful acknowledgement to our chairman **Dr.PSK.R.PERIASWAMY**, Kongunadu Educational Institutions for providing ample facilities in the college for undertaking our project.

We wish to express our gratitude and thanks to our beloved Principal **Dr. R. ASOKAN, M.S.,M.Tech.,Ph.D.,** for forwarding us to do our project and offering adequate duration in completing our project.

We would like to thank **Dr.J.YOGAPRIYA, M.E.,Ph.D.,** Dean(R&D) Kongunadu college of Engineering and Technology for her encouragement during the course of study.

We have immense pleasure in expressing my sincere gratitude to our Head of the Department **Dr.M.DHARMALINGAM, M.E.,Ph.D.,** and my project supervisor **Dr.K.AMUDHA, M.E.,Ph.D.,** for their meticulous guidance which was an inspiration to us.

We also express our gratefulness to my **PARENTS** and my **FRIENDS** for their affectionate blessings and loving co-operation at all stages of this academic venture.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

OCR	Optical Character Recognition
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
OpenCV	Open-Source Computer Vision Library
ITS	Intelligent-Transportation-Systems
ANPR	Automatic-Number-Plate-Recognition
JPEG	Joint Photographic Experts Group
AMD	Advanced Micro Devices
SSD	Solid-State Drive
DBMS	Database Management System
SQLite	Structured Query Language Lite
API	Application Programming Interface
GUI	Graphical User Interface

CHAPTER 1
INTRODUCTION

1.1. GENERAL

Technology has become a key solution for addressing daily challenges, and one of the areas it has greatly improved is vehicle number plate detection. This kind of system is commonly used for things like managing traffic, monitoring parking spaces, and enhancing security. The process involves using image processing and Optical Character Recognition (OCR) to pull details from images of vehicles, such as their number plates, which are then checked against stored data. This technology saves time, increases accuracy, and reduces the need for manual work.

With improvements in computer vision and more powerful computing, detecting number plates in real-time has become more practical and precise. These systems are used by authorities to track vehicles, identify violations, and issue fines where necessary. Their automated nature also reduces human error and helps manage data more efficiently.

This project involves building a web-based system that automates the detection of number plates from vehicle images. After uploading an image, the system identifies the plate and provides the relevant information, such as the owner's details and any associated fines. The system uses tools like Python, OpenCV, and Tesseract OCR to carry out these tasks.

The system is designed for personal use, making it simpler and more accessible than commercial solutions typically used by government or large organizations. It offers a basic yet functional platform for users who want to manage vehicle data for personal or small-scale purposes. The project also leaves room for future updates, such as real-time detection or integration with larger datasets.

1.2. OBJECTIVE

The goal of this project is to create an automated system that detects vehicle number plates and retrieves related information, like the vehicle owner's details and any fines. The system is easy to use and designed for personal purposes. It works through a web page where users can upload vehicle images, and the system will automatically detect the plate and provide the relevant details.

1.3. IMPLEMENTATION

This project is built using a combination of technologies that work together to detect number plates from images. Python serves as the main programming language due to its rich set of libraries for handling image processing and OCR. For detecting the number plate from an image, OpenCV is used to handle image manipulation and find contours, which helps locate the plate on the vehicle. The characters on the plate are then extracted using Tesseract OCR, which converts the image into readable text.

Here’s how the system works: A user uploads an image of a vehicle through a web interface that is built using Flask, a simple and flexible web framework. The system processes the image to identify the number plate. Once detected, it compares the plate's alphanumeric data with a JSON file containing stored information about vehicle owners and any fines they owe. The information is displayed on the web page after the process is complete.

To improve the accuracy of plate detection, the system applies a few image processing techniques such as blurring and thresholding, which help reduce noise and make it easier to detect the plate area. The dimensions of the plate are verified to ensure it fits common size standards, making the detection process more reliable.

The user interface is designed to be straightforward. Flask helps in serving the HTML pages where users upload the vehicle image and get the

results. The front end uses basic styling, keeping the platform easy to use for non-technical users.

1.4. MOTIVATION

The motivation for this project comes from the need to manage vehicle data more efficiently. Keeping track of vehicles and their associated information can be a challenging and time-consuming task if done manually. Automating this process not only saves time but also reduces errors and simplifies managing vehicle-related details. This project provides a tool that offers a quick and easy way to retrieve information by simply uploading an image.

The system also addresses the growing need for tools that can help manage traffic violations, such as unpaid fines. By integrating fine details into the platform, the system provides not just the vehicle owner's information, but also whether the vehicle has any outstanding fines, making it useful in a variety of personal situations.

Another inspiration for the project is the rapid development of tools like OpenCV and Tesseract, which make it easier for developers to work on image processing tasks. With these tools becoming more accessible, the project takes advantage of them to create a reliable and easy-to-use system for number plate detection.

CHAPTER 2

LITERATURE REVIEW

[1] Lubna, Mufti N, Shah SAA. Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorithms. Sensors (Basel). 2021 Apr 26;21(9):3028. doi: 10.3390/s21093028. PMID: 33925845; PMCID: PMC8123416.

Technologies and services towards smart-vehicles and Intelligent-Transportation-Systems (ITS), continues to revolutionize many aspects of human life. This paper presents a detailed survey of current techniques and advancements in Automatic-Number-Plate-Recognition (ANPR) systems, with a comprehensive performance comparison of various real-time tested and simulated algorithms, including those involving computer vision (CV). ANPR technology has the ability to detect and recognize vehicles by their number-plates using recognition techniques. Even with the best algorithms, a successful ANPR system deployment may require additional hardware to maximize its accuracy. The number plate condition, non-standardized formats, complex scenes, camera quality, camera mount position, tolerance to distortion, motion-blur, contrast problems, reflections, processing and memory limitations, environmental conditions, indoor/outdoor or day/night shots, software-tools or other hardware-based constraint may undermine its performance. This inconsistency, challenging environments and other complexities make ANPR an interesting field for researchers. The Internet-of-Things is beginning to shape future of many industries and is paving new ways for ITS. ANPR can be well utilized by integrating with RFID-systems, GPS, Android platforms and other similar technologies. Deep-Learning techniques are widely utilized in CV field for better detection rates. This research aims to advance the state-of-knowledge in ITS (ANPR) built on CV algorithms; by citing relevant prior work, analysing

and presenting a survey of extraction, segmentation and recognition techniques whilst providing guidelines on future trends in this area.

[2] K. Yogheedha, A. S. A. Nasir, H. Jaafar and S. M. Mamduh, "Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach," 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia, 2018, pp. 1-8, doi: 10.1109/ICASSDA.2018.8477639.

A vehicle license plate recognition system is an important proficiency that could be used for identification of engine vehicle all over the earth. It is valuable in numerous applications such as entrance admission, security, parking control, road traffic control, and speed control. However, the system only manages to identify the license number and needs an operator to control the collected data. Therefore, this paper proposes an automatic license plate recognition system by using the image processing and template matching approach. The current study aims to increase the efficiency of license plate recognition system for University Malaysia Perlis (UniMAP) smart university. This venture comprises of simulation program to recognize license plate characters where a captured image of vehicles will be the input. Then, these images will be processed using several image processing techniques and optical character recognition method in order to recognize the segmented number plate. The image processing techniques consist of colour conversion, image segmentation using Otsu's thresholding, noise removal, image subtraction, image cropping and bounding box feature. The optical character recognition based on template matching approach is used to analyse the printed characters on the segmented license plate image and to produce an output data consisting of characters. Overall, the proposed automatic vehicle license plate recognition system is capable to perform the recognition

process by successfully recognizing license plate of 13 cars, from a total of 14 cars.

[3] G. G. Desai and P. P. Bartakke, "Real-Time Implementation of Indian License Plate Recognition System," *2018 IEEE Punecon*, Pune, India, 2018, pp. 1-5, doi: 10.1109/PUNECON.2018.8745419.

A simple and fast technique is presented in this paper for Indian license plate recognition system. Using OpenALPR's software framework and RaspberryPi, a real-time Indian license plate recognition system could be implemented for some application specific purposes. HAAR and LBP features are extracted from the acquired vehicle images and subjected to training of cascade classifiers in order to localize license number plates. The validation process is used to optimally select number of stages of cascade classifiers. The extracted number plates are then utilized for character recognition. Cascade classifier with LBP features is suitable for localization of license plates with accuracy more than 98%. Whereas, the average number plate recognition accuracy is above 96% for images captured from front side. The proposed system has been prototyped using C++ and RaspberryPi 3 and experimental results have been shown for recognition of Indian license plates.

[4] Tang, Y., Zhang, C., Gu, R. *et al.* Vehicle detection and recognition for intelligent traffic surveillance system. *Multimed Tools Appl* 76, 5817–5832 (2017).

Vehicle detection and type recognition based on static images is highly practical and directly applicable for various operations in a traffic surveillance system. This paper will introduce the processing of automatic vehicle detection and recognition. First, Haar-like features and AdaBoost algorithms are applied for feature extracting and constructing classifiers, which are used to locate the

vehicle over the input image. Then, the Gabor wavelet transform and a local binary pattern operator is used to extract multi-scale and multi-orientation vehicle features, according to the outside interference on the image and the random position of the vehicle. Finally, the image is divided into small regions, from which histograms sequences are extracted and concentrated to represent the vehicle features. Principal component analysis is adopted to reach a low dimensional histogram feature, which is used to measure the similarity of different vehicles in euler space and the nearest neighborhood is exploited for final classification. The typed experiment shows that our detection rate is over 97 %, with a false rate of only 3 %, and that the vehicle recognition rate is over 91 %, while maintaining a fast processing time. This exhibits promising potential for implementation with real-world applications.

[5] R. Mukherjee, A. Pundir, D. Mahato, G. Bhandari and G. J. Saxena, "A robust algorithm for morphological, spatial image-filtering and character feature extraction and mapping employed for vehicle number plate recognition," ***2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*****, Chennai, India, 2017, pp. 864-869, doi: 10.1109/WiSPNET.2017.8299884.**

This paper illustrates the implementation of a character feature extraction, classifier and an effective mapping algorithm applied on images of vehicle license plates, using nonlinear feature extraction and spatial mapping. A novel algorithm for low-level visual feature extraction has been presented, highlighting non-linear morphological operations with structuring, filtering, erosion and dilation techniques. Filtering and enhancement of raw image for key feature extraction and recognition by edges by sobel, canny and Fuzzy logic, convolution, median filtering and scaling image for boosting visual spatial features has been performed. Template matching using thresholding by

effective feature mapping and semantic feature matching methods with pre-fed datasets has been accomplished as an application to image computing. The results obtained are tabulated and an accuracy of 79.30% using the above-mentioned algorithm is reported. The images henceforth have been exposed to standard channel noises and thereafter compared for loss of information and overall structure. Comparison methodologies used are MSE and PSNR values and Structured Similarity Index (SSIM). The accuracy of results thus obtained are reported, suggesting the robustness and effectiveness of the algorithm. Image acquisition is accomplished by a remotely controlled android platform-based device integrated with the MATLAB platform in real-time. The acquired number-plate data is transferred into a text file for creating a vehicle index log, critical for security systems.

CHAPTER 3

PROPOSED SYSTEM

3.1. INTRODUCTION

This project presents a smart, automated number plate detection system, primarily designed to simplify and enhance vehicle data retrieval through optical character recognition (OCR) and an easy-to-use web interface. Traditional methods of manually recording or verifying vehicle information are time-consuming and susceptible to errors, especially in situations requiring fast processing, such as traffic management, security checks, and fine enforcement. Existing systems are often limited in scope, sometimes reliant on hardware-intensive solutions or lacking efficient user interfaces for individual use.

This proposed system leverages OCR technology alongside the OpenCV and Tesseract libraries to detect and read alphanumeric characters on number plates with high accuracy. By using a Flask-based web interface, users can upload vehicle images from any device with internet access, making the solution accessible and user-friendly. Additionally, the system stores and retrieves vehicle owner details and any associated fines from a local JSON file, reducing reliance on extensive databases while maintaining essential information securely and efficiently.

One key improvement over earlier method is the integration of a web-based interface, which allows for seamless interaction without the need for specialized software installations. The local JSON data structure also supports quick updates and searches, enabling users to add or modify records without complex database management. As a result, this system offers a streamlined approach to number plate recognition, meeting both functional and accessibility requirements in a way that surpasses more traditional, labour-intensive solutions.

3.2. SYSTEM OVERVIEW

The automated number plate detection system streamlines the process of recognizing vehicle license plates and retrieving associated information. The system workflow is designed to be user-friendly and efficient, following a sequential process from image upload to result display. Below is an overview of each stage in the system:

1. **Image Upload**: Users begin by accessing a Flask-based web interface where they can upload an image containing a vehicle's number plate. This interface supports JPEG image format.

2. **Image Pre-processing**: After the image is uploaded, it undergoes pre-processing to improve detection accuracy. OpenCV is utilized to adjust image parameters, including converting the image to grayscale, applying noise reduction, and enhancing contrast to make characters on the number plate more distinguishable from the background.

3. **Number Plate Detection**: Using OpenCV, the system identifies the region within the image that contains the number plate. This involves edge detection and contouring methods to locate rectangular shapes, followed by filtering based on standard size and aspect ratio parameters typical of vehicle plates.

4. **Character Recognition with OCR**: Once the number plate is isolated, Tesseract OCR processes the segmented region to extract alphanumeric text from the plate. The OCR engine reads each character on the number plate, translating the visual data into readable text for further analysis.

5. **Data Retrieval and Matching**: The extracted number plate information is then matched against records stored in a local JSON file. This file contains details of registered vehicle owners, including the owner's name, contact details, and any relevant fines or penalties associated with the vehicle. The system searches the JSON file for a matching entry, retrieving and displaying the related information.

6. **Display of Results**: Upon a successful match, the system displays the vehicle and owner details on the web interface. Information such as the owner's name, contact details, and any fines is shown alongside the processed image, allowing users to verify and review the result quickly. If no match is found, the

system informs the user that no relevant records exist for the detected number plate.

7. **Error Handling and Notifications**: Throughout the process, the system checks for potential issues, such as poor image quality or failure to detect a number plate. If these issues arise, the user is notified, and they are encouraged to re-upload a clearer image to ensure successful detection.

This automated, step-by-step process significantly reduces the time and effort needed for manual verification, providing accurate results through an accessible and intuitive web interface. The system also allows for easy data updates in the JSON file, enabling seamless scalability for future applications.

3.3. BLOCK DIAGRAM

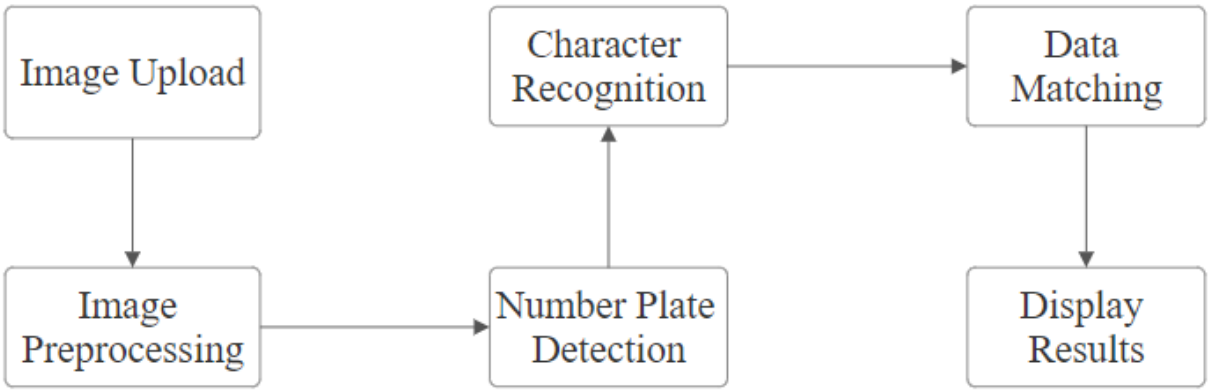

Figure 3.1 Block Diagram

1. **Image Pre-processing**:

- When an image is uploaded, it undergoes pre-processing to enhance quality for optimal detection accuracy.
- This step includes resizing the image, adjusting brightness and contrast, and reducing noise.

- Various filters are applied to sharpen features, helping the system to identify number plate regions more accurately.

2. **Plate Detection**:

- Using computer vision techniques, the system identifies the region of the image containing the number plate.
- Edge detection and contour-finding algorithms are applied to locate rectangular areas resembling a number plate.
- After detection, the system isolates this region, cropping it from the original image for further analysis.

3. **Character Recognition**:

- The cropped number plate region is passed to an Optical Character Recognition (OCR) system, such as Tesseract OCR.
- The OCR engine processes the image to identify and extract alphanumeric characters on the plate.
- It then converts these characters into a text format, making them ready for data matching in the next step.

4. **Data Retrieval**:

- The extracted text is used to search a local JSON database containing registered owner information and any outstanding fines.
- When a match is found, the system retrieves details such as the owner's name, contact information, and fine history (if applicable).
- Finally, these results are displayed on the web interface, providing users with a comprehensive view of the vehicle and owner data based on the uploaded image.

This flow ensures an efficient and streamlined process from image input to result output, achieving a practical solution for quick access to vehicle and owner information.

3.4. FLOWCHART

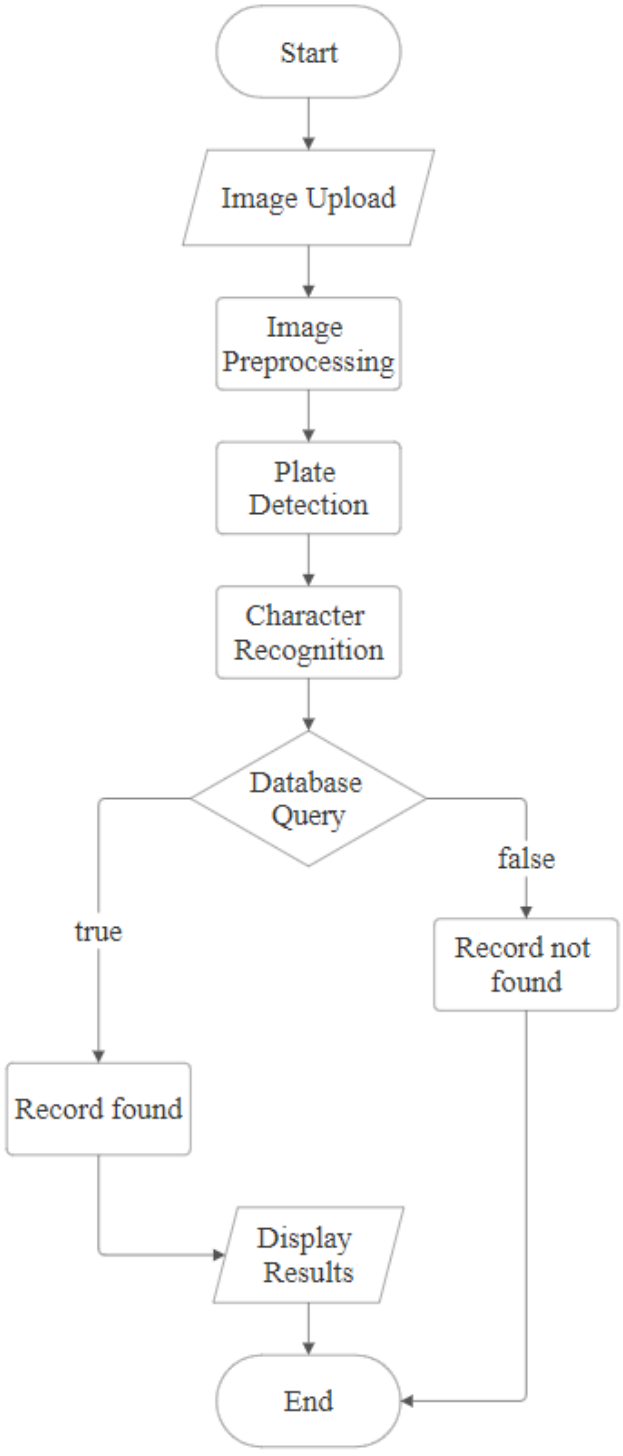

Figure 3.2 Flow Chart

3.5. EXPECTED OUTPUT

The end-user will experience a simple and clear interface that displays relevant information after uploading an image of a vehicle's number plate. Here's a breakdown of what they will see:

1. **Image Upload Section**:
    - At the top of the page, there is an option to upload a vehicle image containing a visible number plate.
    - Upon selecting an image, the user can click "Upload" to begin processing.
2. **Output Section**:

    Once processing is complete, the system shows:

    - **Detected Number Plate**: The recognized alphanumeric characters from the number plate are displayed clearly.
    - **Owner Details**: Information about the registered owner, such as name, address, and contact details.
    - **Fines and Violations**: Any outstanding fines associated with the vehicle, including violation type, fine amount, and date issued.
3. **Example Output**:
    - **Detected Number Plate**: TN87A3980
    - **Owner Name**: Suresh Kumar
    - **Contact Information**: 9988112233
    - **Address**: T Nagar, Chennai, Tamil Nadu
    - **Fine Details**:

- **Violation**: No Helmet
- **Location**: Mount Road
- **Landmark**: Near Spencer Plaza
- **Date**: 05/09/2024
- **Time**: 11:45 AM
- **Fine Amount**: ₹300
- **Status**: Paid

This setup provides a straightforward and efficient way for users to access essential details about a vehicle and its associated fines, directly from an uploaded image.

CHAPTER 4

HARDWARE DESCRIPTION

4.1. INTRODUCTION

This project is configured to run in a local environment, leveraging the processing power of a computer rather than a remote server or cloud infrastructure. Operating locally enables complete control over data and resources without reliance on internet connectivity, ensuring both privacy and real-time processing capabilities. However, to manage resource-intensive tasks like image preprocessing, Optical Character Recognition (OCR), and data handling efficiently, specific hardware specifications are recommended. These requirements are critical to support smooth functioning of Python-based libraries like OpenCV for image processing and Tesseract OCR for character recognition, as well as the Flask framework for the web interface.

4.2. HARDWARE COMPONENTS

Since this project operates locally, the primary hardware component is a standard computer or laptop. Below are the specifications required to maintain optimal performance for image processing, OCR, and web interface management:

- **Computer/Laptop Requirements:**
    - **Processor**: A multi-core processor, preferably Intel i5 or AMD Ryzen 5 and above, to handle the computational load required by Python libraries.
    - **RAM**: At least 8 GB of RAM is recommended, as OCR tasks and image processing can be memory-intensive. More memory will improve efficiency, especially if the application processes high-resolution images.

- **Storage**: At least 256 GB of SSD storage is recommended for faster data access and system boot time. While the storage requirement is relatively low, having an SSD can enhance performance during multiple tasks.
- **Operating System**: Compatible with Windows, macOS, or Linux, allowing flexibility based on the user's system. Linux is preferred for its efficiency with Python packages and libraries, though Windows and macOS are also fully compatible.

This setup ensures that the local environment is capable of handling computational demands while remaining responsive and efficient. No additional computing devices or peripherals are required, as all software dependencies run directly on the host machine.

4.3. SYSTEM REQUIREMENTS

For deploying and running the Flask application that serves as the user interface, along with the OCR functionalities, the following system requirements and software dependencies are necessary:

- **System Specifications:**
  - **Operating System**: Windows 10/11, macOS, or Linux, supporting Python 3.8 or higher. Linux is often recommended for ease of handling Python-based environments and dependencies.
  - **Python Environment**: Python 3.8 or higher is required for compatibility with OpenCV, Tesseract OCR, and Flask. A virtual environment such as venv or conda can be used to manage project dependencies separately.
  - **Flask Web Server**: Flask serves as the backend framework to handle image upload, user interactions, and data display. Running this

application locally requires the Flask development server, which can be initiated from the command line.

- **OpenCV Library**: This library is essential for image preprocessing and handling, such as resizing, grayscale conversion, and edge detection, enabling the number plate detection process.
- **Tesseract OCR**: The OCR engine, which is installed separately, is required for accurate character recognition from processed images. This component converts the detected number plate images into machine-readable text.
- **JSON Data Storage**: A local JSON file is used for storing vehicle and owner details, eliminating the need for a separate database management system (DBMS). However, for larger datasets, an SQLite database may be configured as an alternative for structured data retrieval.

This setup ensures that the system runs efficiently, with resources sufficient to support the necessary tasks involved in image processing, OCR, and data matching. Running the system locally provides a compact and secure environment, particularly suitable for personal and development use cases.

CHAPTER 5

SOFTWARE DESCRIPTION

In this chapter, we discuss the software components required to build, deploy, and run the automated number plate detection system. The software environment comprises Python as the primary programming language, several key libraries for image processing and optical character recognition, and Flask for creating the web-based interface. Each software element plays an essential role in delivering the desired functionality and performance of the system.

5.1. PYTHON AND LIBRARIES

Python is the foundational programming language used in this project due to its extensive support for data processing, machine learning, and web frameworks, as well as its ease of use and community support. Below, we outline each key Python library used in this system, describing their specific roles and benefits.

1. **Python**:
    - **Description**: Python is a versatile, high-level programming language renowned for its readability and broad application in fields ranging from data science to web development. Its large ecosystem of libraries and frameworks makes it an ideal choice for projects involving image processing and machine learning.
    - **Role in Project**: Python serves as the backbone of this system, connecting all other components and enabling seamless interactions between libraries. Its built-in functions and support for external libraries simplify the development and integration of various functionalities required for number plate detection and information retrieval.

2. **OpenCV (Open Source Computer Vision Library)**:
    - **Description**: OpenCV is an open-source computer vision and machine learning software library. It provides extensive tools for image processing, including techniques for handling grayscale transformations, edge detection, and shape recognition.
    - **Role in Project**: OpenCV handles image preprocessing in this system, including resizing, converting to grayscale, and enhancing contrast to facilitate more accurate number plate detection. By enabling accurate segmentation and extraction of number plates from the uploaded images, OpenCV ensures a high level of precision, which is critical for the OCR stage.
3. **Tesseract OCR**:
    - **Description**: Tesseract is a popular open-source OCR engine developed by Google. It transforms images of text into machine-readable text, allowing the extraction of characters from images.
    - **Role in Project**: Tesseract OCR performs character recognition on the preprocessed number plate images extracted by OpenCV. It converts the alphanumeric characters on the detected number plate into text that can then be used to match records in the local database. Tesseract's accuracy in text recognition is crucial for the reliability of the system in identifying vehicle information accurately.
4. **Flask**:
    - **Description**: Flask is a lightweight web framework for Python that is widely used for building web applications and APIs. It is known for its flexibility and simplicity, making it well-suited for small to medium-sized applications.

- **Role in Project**: Flask serves as the web interface for this system, providing a user-friendly platform where users can upload images and receive results. It manages image uploads, directs processing tasks, and displays the output in a structured and accessible manner. Flask's built-in development server also allows the system to run locally without the need for complex server configurations.

5.2. WEB APPLICATION FRAMEWORK

Flask was chosen as the web application framework for this project due to its simplicity, flexibility, and suitability for developing lightweight applications. Flask is a micro-framework in Python, meaning it provides the essential tools needed for web development without excessive features, making it ideal for projects that don't require complex functionality.

Why Flask?

Flask is designed for projects that require fast development cycles and straightforward deployment, making it a preferred choice for small to medium-sized applications, particularly those involving real-time data processing. For our number plate detection system, Flask's advantages include:

1. **Lightweight and Flexible**: Flask is unopinionated, meaning it doesn't impose specific project structures or dependencies. This flexibility allowed us to build a custom application tailored precisely to our requirements, handling image uploads, processing, and displaying results without unnecessary overhead.
2. **Ease of Integration**: Flask seamlessly integrates with various Python libraries, including OpenCV and Tesseract OCR, which are critical for image processing and character recognition. This integration allows us to

coordinate image uploads, detection, and character extraction processes within a single web framework, streamlining the user experience.

3. **Efficient Local Hosting**: Flask comes with a built-in development server, allowing for easy local hosting of the project. For this locally run application, Flask handles all incoming requests, processes image data, and serves results in real-time, eliminating the need for additional server configurations.

How Flask Facilitates User Interaction

Flask's simplicity allows us to provide a clean, accessible interface where users can interact with the system easily. The web interface is structured to accept image uploads directly from the user, which are then processed in real-time. The steps involved in facilitating user interaction through Flask are as follows:

1. **Image Upload Interface**: Flask handles the front-end of the application by rendering HTML forms that allow users to upload vehicle images. This interface provides a clear, simple input option for the user, making it intuitive even for those without technical experience.
2. **Backend Processing**: Once an image is uploaded, Flask passes it to the backend for preprocessing, number plate detection, and character recognition. Flask's ability to manage asynchronous tasks ensures that image processing is handled promptly, and results are sent back to the interface seamlessly.
3. **Output Display**: Flask generates dynamic responses based on the processed data. After detection and character recognition are complete, Flask's templating engine renders the results, displaying vehicle information such as the owner's name, address, and any associated fines.

The clear display structure in Flask enhances the user experience, making results easy to read and understand.

5.3. DATABASE AND DATA STORAGE

In this project, the db.json file acts as a simple database to store owner and vehicle information. It is designed to hold structured data for each vehicle's number plate, including owner details, fine information, and any related infractions. JSON (JavaScript Object Notation) is chosen for its easy readability, compatibility with Python, and flexibility in storing nested data.

Structure of db.json

The structure of the db.json file is straightforward, storing vehicle data as key-value pairs, where each key represents a unique vehicle number plate, and the associated value is a dictionary containing relevant information. Here's a sample structure:


```
{
  "KL26H5009": {
    "Name": "Amit Sharma",
    "Address": "Jayanagar, Bangalore, Karnataka",
    "Phone": "9876543210",
    "Fine Details": "Signal Jump",
    "Location": "MG Road",
    "Near": "Metro Station",
    "Date": "12/09/2024",
    "Time": "09:30AM",
    "Fine Amount": "500",
    "Paid or Unpaid": "Unpaid"
  },
  "HR26DK8337": {
    "Name": "Priya Verma",
    "Address": "Connaught Place, New Delhi",
    "Phone": "8765432109",
    "Fine Details": "Wrong Parking",
    "Location": "Karol Bagh Market",
    "Near": "McDonald's",
    "Date": "25/08/2024",
    "Time": "02:00PM",
    "Fine Amount": "1000",
```

```
        "Paid or Unpaid": "Paid"
    }
}
```


Each entry within db.json consists of:

Vehicle Number Plate: This serves as the unique key for each entry, allowing for quick lookup.

Owner Details: Contains the name, address, and phone number of the vehicle owner.

Fine Details: Lists the type of infraction (e.g., "No Helmet," "Speeding").

Location: Includes specific location details, helping to identify where the infraction occurred.

Date and Time: Specifies the exact date and time of the infraction.

Fine Amount: The monetary penalty associated with the infraction.

Paid Status: Indicates whether the fine is "Paid" or "Unpaid."

Storage and Retrieval Process

Storage: The db.json file is saved locally and updated manually as new vehicle records and infraction details are added. JSON's format allows for adding, removing, or modifying entries easily by editing the file directly.

Data Retrieval: In the Flask application, the db.json file is loaded into Python dictionaries, enabling rapid lookup. When a user uploads an image of a vehicle, the number plate is extracted using OCR, and the system performs a search in db.json for a matching entry.

Display: Once the relevant data is retrieved, it's passed to the web interface to display the owner's details and fine information in an organized, user-friendly manner.

Using JSON allows us to maintain data integrity while keeping the system lightweight and easy to manage. This format provides a practical solution for local storage, making data access quick and simple for this project's needs.

CHAPTER 6

RESULTS AND DISCUSSION

6.1. SYSTEM TESTING

Testing involved various scenarios to evaluate the system's accuracy and reliability under different conditions. Each test was designed to assess how effectively the OCR (Optical Character Recognition) component and the web interface handled real-world image variations and external factors.

- **Lighting Conditions**: The system was tested with images taken in different lighting environments, such as bright daylight, indoor lighting, and low light. Bright daylight images typically yielded higher accuracy, while low-light conditions resulted in some challenges due to shadowing on the number plate.

- **Image Angles**: Images were uploaded at varying angles to simulate real-life scenarios where the vehicle may not be photographed head-on. Moderate angles were generally handled well by the OCR system, but very steep angles caused distortions, reducing accuracy.

- **Plate Designs**: Different number plate designs, including variations in font style, color, and character spacing, were tested. While standard plates were accurately detected, custom fonts or designs sometimes decreased recognition accuracy, especially if the text size was non-standard.

- **Blurred or Low-Resolution Images**: Test images with intentional blur or lower resolutions were used to assess the system’s limits. Blurred images affected OCR performance considerably, but mild blurring could be processed if the characters were still distinguishable.

6.2. APPLICATION INTERFACE

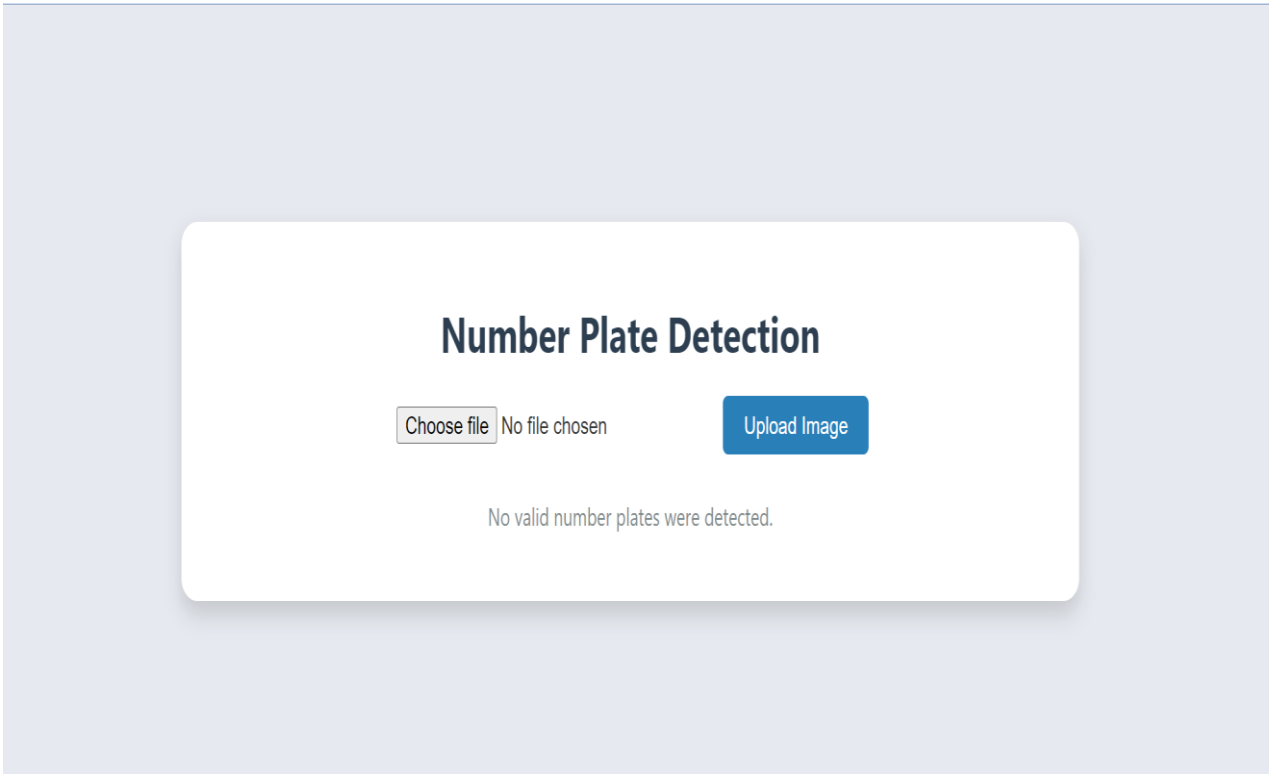

Figure 6.1 APPLICATION INTERFACE - I

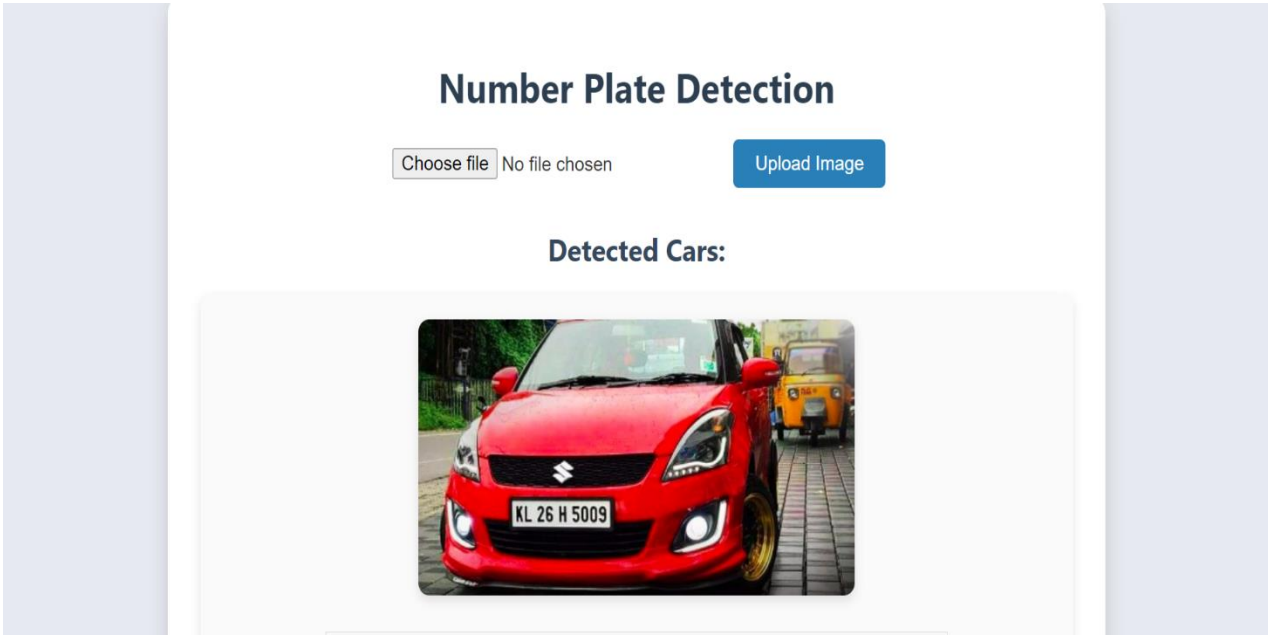

Figure 6.2 APPLICATION INTERFACE - II

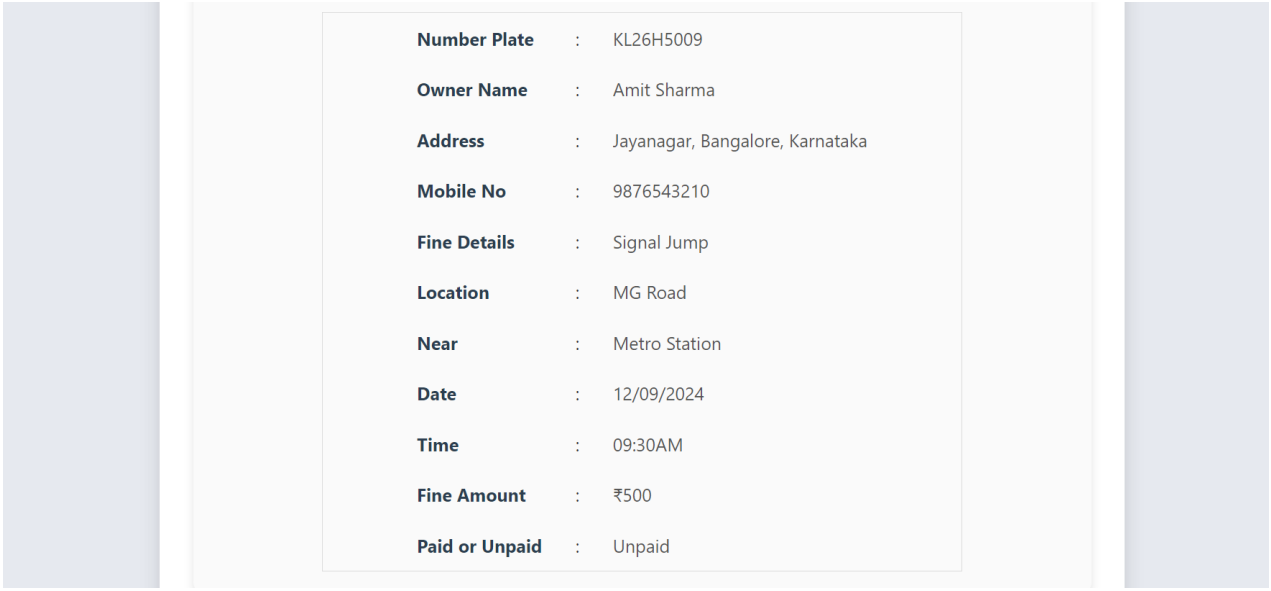

Figure 6.3 APPLICATION INTERFACE - III

6.3. ANALYSIS OF RESULTS

Based on testing, the system demonstrated high accuracy with standard, clear images taken in adequate lighting. For instance, well-lit images with standard font plates were recognized with over 90% accuracy. In low light, accuracy dropped by around 20% due to increased noise and shadows in the images. However, by applying adaptive thresholding, the impact of lighting was somewhat mitigated.

In terms of speed, the OCR and data-matching process averaged around 2-3 seconds per image, making the system responsive enough for real-time applications. For certain image types, preprocessing steps, such as noise reduction and contrast enhancement, helped improve accuracy without significantly affecting processing time.

6.4. CHALLENGES AND SOLUTIONS

Throughout the development and testing phases, several issues arose. Here's a summary of key challenges and the corresponding solutions:

- **OCR Accuracy**: OCR accuracy was impacted by non-standard fonts, angled images, and low resolution. To counter this, preprocessing steps like adaptive thresholding and contrast adjustment were added to enhance text visibility. These steps improved OCR performance, especially in varying light conditions.

- **Image Quality**: Blurred or low-quality images reduced recognition accuracy. To address this, the system was adjusted to prompt users if image quality was too low, encouraging them to upload clearer images.

- **Lighting Variability**: Shadowing and reflections in certain lighting conditions caused some detection issues. To minimize this, grayscale conversion and brightness adjustments were applied, and testing was repeated to confirm improvement. This approach helped standardize images and reduced the impact of harsh lighting.

- **Data Retrieval Speed**: As the db.json file grows, retrieval time may increase. Although not currently a major issue, indexing or migrating to a more robust database solution could be considered for scaling the system.

CHAPTER 7

CONCLUSION

This project successfully developed an automated system for detecting vehicle number plates and retrieving relevant owner and fine details through a simple, user-friendly web interface. Leveraging Python, OpenCV, Tesseract OCR, and Flask, the system processes images uploaded by users, performs Optical Character Recognition (OCR) on detected number plates, and retrieves matching owner details from a locally stored JSON file. Practical testing confirmed the system's effectiveness in various scenarios, including different lighting conditions and plate designs, with standard images showing high recognition accuracy.

Accomplishments

The main accomplishments of the project include:

- **Accurate OCR Processing**: The system effectively detects and reads number plates in images under standard conditions, with a high level of accuracy in text extraction.
- **Data Retrieval**: Integration with a JSON file enabled reliable retrieval of relevant vehicle owner and fine details, making the system functional without a complex database.
- **User-Friendly Interface**: The Flask-based web application allows easy interaction, enabling users to upload images and receive results efficiently.

Limitations

Despite its achievements, the system has certain limitations:

- **Image Quality Dependence**: The system’s accuracy can be affected by low-quality images, blurred text, or excessive angle variations.

- **Lighting Sensitivity**: Performance in poor lighting is reduced, with shadows and reflections occasionally affecting OCR accuracy.

- **Local File Storage**: Using a JSON file limits the project's scalability, as managing larger datasets or frequent updates would require transitioning to a database.

Future Improvements

There are several promising directions for enhancing this project:

- **Real-Time Video Processing**: Expanding to handle video input would allow for real-time detection in surveillance or traffic-monitoring applications, capturing moving vehicles and storing detection logs dynamically.

- **Advanced Database Integration**: Migrating from a JSON file to a scalable database solution, such as SQLite or MongoDB, would enable handling a larger volume of data and facilitate quicker searches.

- **Enhanced Preprocessing Techniques**: Further refining image preprocessing, such as implementing machine learning-based image enhancement or noise reduction, could improve accuracy in diverse lighting conditions.

- **Extended Device Compatibility**: Optimizing the system for deployment on devices with lower computing power, like embedded systems, could enable its use in edge devices, providing versatile deployment options.

In conclusion, this project provides an efficient approach for automated number plate detection with a potential foundation for scaling into broader applications, such as security and traffic management systems. By addressing current limitations and incorporating additional features, the system can be further optimized and extended for real-time, large-scale applications.

APPENDIX

PROGRAM.PY

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


# Load owner details
with open("db.json") as f:
    owner_details = json.load(f)

app = Flask(__name__)


# Detecting number plate
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)
        num_contours, _ = cv2.findContours(
            thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE
```

```
    )

    if num_contours:
        contour_area = [cv2.contourArea(c) for c in num_contours]
        max_cntr_index = np.argmax(contour_area)
        max_cnt = num_contours[max_cntr_index]
        x, y, w, h = cv2.boundingRect(max_cnt)

        if not ratioCheck(cv2.contourArea(max_cnt), 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
    return (1063.62 < area < 73862.5) and (3 < ratio < 6)

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

def ratio_and_rotation(rect):
    (x, y), (width, height), rect_angle = rect
```

```
        angle = -rect_angle if width > height else 90 + rect_angle
        return (
            abs(angle) <= 15
            and width > 0
            and height > 0
            and ratioCheck(width * height, 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 = cv2.morphologyEx(img2, cv2.MORPH_CLOSE,
kernel=element)

    num_contours, _ = cv2.findContours(
        morph_img_threshold, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE
    )

    for cnt in num_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:
                    plate_im = Image.fromarray(clean_plate)
                    text = pytesseract.image_to_string(plate_im, lang="eng")
                    return text
    return None


@app.route("/")
def index():
    detected = False
    dir_path = os.path.dirname(__file__)
    car_images = []

    # Detect number plates in the images
    for img_file in glob.glob(os.path.join(dir_path, "Images", "*.jpeg")):
        img = cv2.imread(img_file)
        img2 = cv2.resize(img, (600, 600))
        number_plate = number_plate_detection(img)

        if number_plate is not None:
            res2 = str("".join(re.split("[^a-zA-Z0-9]*", number_plate))).upper()

            if res2 in owner_details:
                owner_info = owner_details[res2]
                car_images.append(
```

```
                {
                    "plate": res2,
                    "name": owner_info["Name"],
                    "phone": owner_info["Phone"],
                    "address": owner_info["Address"],
                    "fine_details": owner_info["Fine Details"],
                    "location": owner_info["Location"],
                    "near": owner_info["Near"],
                    "date": owner_info["Date"],
                    "time": owner_info["Time"],
                    "fine_amount": owner_info["Fine Amount"],
                    "paid_or_unpaid": owner_info["Paid or Unpaid"],
                    "img_file": img_file,
                }
            )
            detected = True

    return render_template("index.html", car_images=car_images,
detected=detected)


@app.route("/image/<path:filename>")
def get_image(filename):
    return send_file(filename, mimetype='image/jpeg')
    # return send_file(os.path.join("Images", filename), mimetype="image/jpeg")


@app.route("/upload", methods=["POST"])
def upload_image():
```

```
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']

    if file.filename == '':
        return "No selected file"

    if file and file.filename.endswith(".jpeg"):
        # Clear the 'Images' folder before saving the new image
        dir_path = os.path.join("Images")
        if os.path.exists(dir_path):
            shutil.rmtree(dir_path)  # Remove all previous images
        os.makedirs(dir_path)  # Recreate the folder after deletion

        file_path = os.path.join(dir_path, file.filename)
        file.save(file_path)

        return redirect(url_for("index"))  # After upload, redirect to the index to
display results

    return "Invalid file format, only JPEG is allowed"


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

INDEX.HTML

```
<!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: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  min-height: 100vh;
  margin: 0;
  background-color: #e6e9ef;
  color: #333;
 }

 h1 {
  font-size: 2em;
  color: #2c3e50;
  margin-bottom: 20px;
 }

 .container {
  text-align: center;
```

```
  width: 90%;
  max-width: 800px;
  background-color: #ffffff;
  border-radius: 15px;
  padding: 30px;
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.15);
  transition: transform 0.3s ease;
}

button {
  background-color: #2980b9;
  color: white;
  border: none;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1em;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #3498db;
}

input[type="file"] {
  margin-bottom: 15px;
  padding: 5px;
  font-size: 1em;
```

```
}

.car-info {
  padding: 20px;
  margin: 20px 0;
  background-color: #fafafa;
  border-radius: 10px;
  text-align: center;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.car-info img {
  max-width: 400px;
  max-height: 300px;
  width: 100%;
  height: auto;
  margin-bottom: 15px;
  border-radius: 10px;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
}

.car-info table {
  display: flex;
  align-items: center;
  justify-content: center;
  border: 1px solid #ddd;
  width: 75%;
  margin: 0 12.5%;
```

```
  border-collapse: collapse;
  margin-top: 10px;
}

.car-info table td {
  padding: 12px;
  text-align: left;
}

.car-info table td:first-child {
  font-weight: bold;
  color: #2c3e50;
}

.car-info table td:nth-child(2) {
  width: 15px;
  text-align: center;
}

.car-info table td:last-child {
  font-weight: normal;
  color: #555;
}

h2 {
  color: #34495e;
  margin-bottom: 15px;
}
```

```
p {
  color: #7f8c8d;
}


  </style>
</head>
<body>
  <div class="container">
    <h1>Number Plate Detection</h1>
    <form action="/upload" method="POST" enctype="multipart/form-data">
      <input type="file" name="file" accept="image/jpeg" required />
      <button type="submit">Upload Image</button>
    </form>
    {% if detected %}
    <h2>Detected Cars:</h2>
    <div>
      {% for car in car_images %}
      <div class="car-info">
        <img src="/image/{{ car.img_file }}" alt="Car Image" />
        <table>
          <tr>
            <td>Number Plate</td>
            <td>:</td>
            <td>{{ car.plate }}</td>
          </tr>
          <tr>
```

```
  <td>Owner Name</td>
  <td>:</td>
  <td>{{ car.name }}</td>
</tr>
<tr>
  <td>Address</td>
  <td>:</td>
  <td>{{ car.address }}</td>
</tr>
<tr>
  <td>Mobile No</td>
  <td>:</td>
  <td>{{ car.phone }}</td>
</tr>
<tr>
  <td>Fine Details</td>
  <td>:</td>
  <td>{{ car.fine_details }}</td>
</tr>
<tr>
  <td>Location</td>
  <td>:</td>
  <td>{{ car.location }}</td>
</tr>
<tr>
  <td>Near</td>
  <td>:</td>
  <td>{{ car.near }}</td>
```

```
      </tr>
      <tr>
        <td>Date</td>
        <td>:</td>
        <td>{{ car.date }}</td>
      </tr>
      <tr>
        <td>Time</td>
        <td>:</td>
        <td>{{ car.time }}</td>
      </tr>
      <tr>
        <td>Fine Amount</td>
        <td>:</td>
        <td>₹{{ car.fine_amount }}</td>
      </tr>
      <tr>
        <td>Paid or Unpaid</td>
        <td>:</td>
        <td>{{ car.paid_or_unpaid }}</td>
      </tr>
    </table>
  </div>
  {% endfor %}
</div>
{% else %}
<p>No valid number plates were detected.</p>
{% endif %}
```

```
    </div>
  </body>
</html>
```

DB.JSON


```
{
  "KL26H5009": {
    "Name": "Amit Sharma",
    "Address": "Jayanagar, Bangalore, Karnataka",
    "Phone": "9876543210",
    "Fine Details": "Signal Jump",
    "Location": "MG Road",
    "Near": "Metro Station",
    "Date": "12/09/2024",
    "Time": "09:30AM",
    "Fine Amount": "500",
    "Paid or Unpaid": "Unpaid"
  },
  "HR26DK8337": {
    "Name": "Priya Verma",
    "Address": "Connaught Place, New Delhi",
    "Phone": "8765432109",
    "Fine Details": "Wrong Parking",
    "Location": "Karol Bagh Market",
    "Near": "McDonald's",
    "Date": "25/08/2024",
    "Time": "02:00PM",
    "Fine Amount": "1000",
```

```
    "Paid or Unpaid": "Paid"
},
"MH20EE7598": {
    "Name": "Rohit Kulkarni",
    "Address": "Shivaji Nagar, Pune, Maharashtra",
    "Phone": "9988776655",
    "Fine Details": "Over Speeding",
    "Location": "Pune-Mumbai Expressway",
    "Near": "Food Court",
    "Date": "01/11/2024",
    "Time": "04:15PM",
    "Fine Amount": "2000",
    "Paid or Unpaid": "Unpaid"
},
"TN87A3980": {
    "Name": "Suresh Kumar",
    "Address": "T Nagar, Chennai, Tamil Nadu",
    "Phone": "9988112233",
    "Fine Details": "No Seatbelt",
    "Location": "Mount Road",
    "Near": "Spencer Plaza",
    "Date": "05/09/2024",
    "Time": "11:45AM",
    "Fine Amount": "300",
    "Paid or Unpaid": "Paid"
},
"RJ14XZ9907": {
    "Name": "Neha Gupta",
```

```
    "Address": "Malviya Nagar, Jaipur, Rajasthan",
    "Phone": "9191919191",
    "Fine Details": "Red Light Violation",
    "Location": "Tonk Road",
    "Near": "Gaurav Tower",
    "Date": "30/09/2024",
    "Time": "06:30PM",
    "Fine Amount": "1500",
    "Paid or Unpaid": "Unpaid"
},
"GJ01BC7621": {
    "Name": "Rahul Desai",
    "Address": "Satellite, Ahmedabad, Gujarat",
    "Phone": "9292929292",
    "Fine Details": "Drunk Driving",
    "Location": "SG Highway",
    "Near": "Zydus Hospital",
    "Date": "10/10/2024",
    "Time": "10:45PM",
    "Fine Amount": "5000",
    "Paid or Unpaid": "Unpaid"
},
"UP32LK4312": {
    "Name": "Arun Pratap",
    "Address": "Hazratganj, Lucknow, Uttar Pradesh",
    "Phone": "9111111111",
    "Fine Details": "Overloading",
    "Location": "Ring Road",
```

```
    "Near": "Phoenix Mall",
    "Date": "18/10/2024",
    "Time": "03:20PM",
    "Fine Amount": "3500",
    "Paid or Unpaid": "Paid"
},
"WB20FR9090": {
    "Name": "Anjali Sen",
    "Address": "Salt Lake, Kolkata, West Bengal",
    "Phone": "9323232323",
    "Fine Details": "No Seatbelt",
    "Location": "EM Bypass",
    "Near": "Ruby Hospital",
    "Date": "08/07/2024",
    "Time": "05:00PM",
    "Fine Amount": "400",
    "Paid or Unpaid": "Unpaid"
},
"MH20EJ0365": {
    "Name": "Rajeev Mishra",
    "Address": "Civil Lines, Raipur, Chhattisgarh",
    "Phone": "9343434343",
    "Fine Details": "Signal Violation",
    "Location": "G.E. Road",
    "Near": "City Center Mall",
    "Date": "19/06/2024",
    "Time": "01:30PM",
    "Fine Amount": "600",
```

```
    "Paid or Unpaid": "Paid"
  },
  "MP09LM6587": {
    "Name": "Vikram Singh",
    "Address": "New Palasia, Indore, Madhya Pradesh",
    "Phone": "9454545454",
    "Fine Details": "No Insurance",
    "Location": "AB Road",
    "Near": "C21 Mall",
    "Date": "02/08/2024",
    "Time": "08:45AM",
    "Fine Amount": "1200",
    "Paid or Unpaid": "Unpaid"
  }
}
```

REFERENCE

1. Flask Documentation: https://flask.palletsprojects.com/

2. OpenCV Documentation: https://docs.opencv.org/

3. Tesseract OCR Documentation: https://github.com/tesseract-ocr/tesseract

4. Lubna, Mufti N, Shah SAA. Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorithms. Sensors (Basel). 2021 Apr 26;21(9):3028. doi: 10.3390/s21093028. PMID: 33925845; PMCID: PMC8123416.

5. K. Yogheedha, A. S. A. Nasir, H. Jaafar and S. M. Mamduh, "Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach," 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia, 2018, pp. 1-8, doi: 10.1109/ICASSDA.2018.8477639.

6. G. G. Desai and P. P. Bartakke, "Real-Time Implementation of Indian License Plate Recognition System," *2018 IEEE Punecon*, Pune, India, 2018, pp. 1-5, doi: 10.1109/PUNECON.2018.8745419.

7. Tang, Y., Zhang, C., Gu, R. *et al.* Vehicle detection and recognition for intelligent traffic surveillance system. *Multimed Tools Appl* 76, 5817–5832 (2017).

8. R. Mukherjee, A. Pundir, D. Mahato, G. Bhandari and G. J. Saxena, "A robust algorithm for morphological, spatial image-filtering and character feature extraction and mapping employed for vehicle number plate recognition," *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, 2017, pp. 864-869, doi: 10.1109/WiSPNET.2017.8299884.