977-302 Digital Engineering Project I

Semester 2/2024

License Plate Recognition

Project Report

Danny Roberts 6530613033

Advisor: **AJ. Kullawat Chaowanawatee**

Asst. Prof. Dr. Kittasil Silanon

Asst. Prof. Dr. Komsan Kanjanasit

College of Computing

Prince of Songkla University, Phuket Campus

Project Title	License Plate Recognition
Author	DannyRoberts
Major	Digital Engineering
Academic Year	2024

Abstract

The content of this project is to remember the letters of the license plate for the convenience of entering and exiting each place that must be safe. So we use (Raspberry Pi) as a board to be attached to the camera and is small in size that does not take up space. And that during the entry and exit of the car, it will be able to display whether this car is in the data or not, so we will use the data storage as (Firebase) to store data. And we will be able to know whether the car that enters is in the database or not. And we have made a website that can enter data. We use (Flask) for making a website. Make it ready to enter the database for the convenience of members in that place.

Keywords: *License Plate* Recognition

Acknowledgements

I would like to express my heartfelt gratitude to Professor Kullawat Chaowanawatee for his invaluable guidance in teaching me about Raspberry Pi, a platform I had never used before. He provided insightful advice on using SSH to remotely access the Raspberry Pi, eliminating the need for an HDMI connection and making the workflow much more convenient.Additionally, he offered guidance on database management, particularly with Firebase, which was crucial for our project. His support extended beyond technical knowledge—he helped us with OCR implementation and database storage while also giving us a realistic perspective on the feasibility of our project.I truly appreciate all the knowledge, advice, and time he dedicated to mentoring us. His teachings have been incredibly valuable, and I am sincerely grateful for everything he has shared with us.

Danny Roberts

26 Apr 2025

Table of Contents

ABSTRACT II
ACKNOWLEDGEMENTS III

CHAPTER 1 INTRODUCTION 1

1.1 OVERVIEW 1
1.2 BACKGROUND 2
1.3 OBJECTIVES 2
1.4 SCOPES 3
1.5 PROCEDURES 3
1.6 EXPECTED PROJECT RESULTS 3
1.7 LOCATION 3
1.8 DEVELOPMENT TOOLS 4

CHAPTER 2 LITERATURE REVIEW 5

2.1 LICENSE PLATE RECOGNITION USING RASPBERRY PI 5
2.2 VEHICLE NUMBER PLATE RECOGNITION AND OCR PERFORMANCE 6
2.3 REAL-TIME LICENSE PLATE RECOGNITION WITH FIREBASE INTEGRATION 7
2.4 CHAURAH: A SMART RASPBERRY PI-BASED PARKING SYSTEM 8

CHAPTER 3 DETAIL OF SYSTEM 10

3.1 SOFTWARE SPECIFICATION 11
3.1.1 REAL-TIME DETECTION MODULE 11
3.1.2 WEB INTERFACE & CONTROL 12
3.1.3 BACKEND PROCESSING AND DATA LOGGING 12
3.2 HARDWARE REQUIREMENTS 13
3.2.1 EASY SETUP USING A LAPTOP OR DESKTOP 13
THE ENTIRE SYSTEM CAN RUN ON A STANDARD PERSONAL COMPUTER WITHOUT NEEDING TO INSTALL OR CONFIGURE EXTERNAL EMBEDDED DEVICES. USERS ONLY NEED TO CONNECT A WEBCAM OR USE AN IP STREAM FROM A PHONE CAMERA, WHICH SIMPLIFIES BOTH DEVELOPMENT AND DEPLOYMENT 13
3.2.2 BETTER AUTOFOCUS AND LIGHTING COMPENSATION 13
MODERN WEBCAMS AND SMARTPHONE CAMERAS INCLUDE BUILT-IN AUTO-EXPOSURE AND AUTOFOCUS MECHANISMS. THIS SIGNIFICANTLY IMPROVES IMAGE CLARITY, ESPECIALLY IN VARIABLE LIGHTING CONDITIONS (E.G., DAYLIGHT VS. EVENING), WHICH IS ESSENTIAL FOR ENSURING HIGH OCR ACCURACY. 13
3.2.3 PLUG-AND-PLAY COMPATIBILITY VIA USB OR IP CAMERA FEED 14
3.3 SOFTWARE STACK 15
3.3.1 BACKEND FRAMEWORK 15
TECHNOLOGY USED: FLASK (PYTHON WEB FRAMEWORK) 15
FLASK IS A LIGHTWEIGHT AND FLEXIBLE WEB FRAMEWORK USED TO MANAGE ALL ROUTING, BACKEND LOGIC, AND TEMPLATE RENDERING IN THIS SYSTEM. IT IS RESPONSIBLE FOR: 15
- HANDLING HTTP REQUESTS (GET, POST) 15
- MANAGING USER SESSIONS AND AUTHENTICATION 15

- Rendering HTML pages using Jinja2 15
- Connecting to and interacting with the database via SQLAlchemy 15
- Routing between different parts of the application such as login, dashboard, and camera pages 15

Flask's minimal structure and extensibility make it suitable for rapid development of web-based prototypes. 15

3.3.2 OCR and AI Inference Libraries 16

Technologies Used: 16

- EasyOCR (for license plate text detection) 16
- OpenVINO Toolkit (for vehicle type and color classification) 16

EasyOCR is a deep learning-based OCR engine that supports multi-language recognition. It is used to detect and decode characters from license plate regions cropped from each frame. .. 16

3.3.3 Computer Vision and Video Processing 16

3.4 Database Design 18

3.5 Frontend Interface 19

The system's frontend is served by Flask using static HTML templates rendered through the Jinja2 engine. The user interface is designed to separate functionalities between general users and administrators, ensuring role-based access control while maintaining a simple, responsive, and intuitive user experience. 19

3.5.1 Login and Registration Pages 19

The system provides a simple and user-friendly interface for authentication through dedicated login and registration pages. These are implemented using the login.html and register.html templates. New users can create an account with a unique username and password, while existing users can log in with their credentials. Authentication is managed via Flask session variables, which maintain the user's login state throughout their interaction with the platform. 19

Upon incorrect login or successful registration, the system displays flash messages using Bootstrap alert components. The forms are styled with consistent Bootstrap form classes to ensure clarity, accessibility, and responsiveness across devices. 19

3.5.2 User Dashboard 19

After logging in, non-admin users are redirected to their personal dashboard (dashboard.html). Here, they are provided with a form to manually submit vehicle data. The form includes fields such as license plate number, province, brand, model, and vehicle category (e.g., Employee, Visitor, Customer). An image of the vehicle or plate can also be uploaded through this interface. 19

Once submitted, the data is recorded into the MySQL database and the uploaded image is saved under the static/uploads/ directory. Beneath the submission form, users can view a personal log of their entries, organized in a table format with timestamps, plate numbers, and thumbnail previews of the uploaded images. This provides a clear record of their interactions with the system. 19

3.5.3 Admin Dashboard 19

3.5.4 Admin Dashboard 20

3.5.5 Styling and Layout 20

3.6 Backend Logic and Data Flow 21

3.7 Real-Time Detection and Camera Loop 22

3.7.1 Camera Frame Capture 22

3.7.2 License Plate Detection and Cropping 22

3.8 Project Plan 23

CHAPTER 4 RESULTS AND CONCLUSION 24

4.1 RESULTS 24

REFERENCES 29

LIST OF FIGURES

Figure 1Vehicle Number Plate Recognition and OCR 6
Figure 2Real-Time License Plate Recognition with Firebase Integration 7
Figure 3A Smart Raspberry Pi-based Parking System 8
Figure 4Detail of System 11
Figure 5Software Stack 15
Figure 6Car 25
Figure 7Live License Plate 26
Figure 8Real-Time table 26

LIST OF TABLES

Table 1Project Plan.....23

LIST OF ABBREVIATIONS

LPR	License Plate Recognition
OCR	Optical Character Recognition
API	Application Programming Interface
UI	User Interface
DB	Database
SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
CV	Computer Vision
AI	Artificial Intelligence
USB	Universal Serial Bus

Chapter 1 Introduction

1.1 Overview

This project introduces a smart License Plate Recognition (LPR) system developed using open-source tools and consumer-grade hardware. The system is designed to automatically detect, recognize, and classify vehicle license plates in real time using a webcam or mobile phone camera, and to store and display the results through a web-based interface.

The core architecture leverages a combination of computer vision and deep learning models. EasyOCR, a pre-trained optical character recognition library, is used to extract alphanumeric text from detected license plate regions. OpenVINO is integrated for vehicle attribute classification such as type (car, bus, truck) and color (red, white, black, etc.). OpenCV handles video input and preprocessing, while Flask serves as the lightweight web server that connects the frontend UI with backend data processing and MySQL database storage.

The system supports both manual and automatic data collection. Authenticated users can upload plate information and vehicle images via a user dashboard, while the administrator can monitor live camera input, receive OCR results, and browse detailed records via the admin dashboard.

1.2 Background

With the rise in vehicular traffic and urban development, efficient traffic monitoring and vehicle identification systems have become an essential requirement in cities worldwide. License Plate Recognition (LPR) systems, often powered by OCR technology, are key components in smart city frameworks. They are used for law enforcement, entrance security, tolling systems, and parking automation.

This project proposes a smart LPR system using a general-purpose computer and a webcam or mobile camera as input. The system applies EasyOCR, a deep learning-based text detection library, to extract license plate numbers from vehicles in real time. Coupled with OpenVINO, a computer vision acceleration toolkit, the system further classifies vehicle types and colors using pre-trained models.

Unlike systems that rely on specific hardware like Raspberry Pi or cloud services like Firebase, this project employs Flask, a lightweight Python-based web framework. This ensures platform independence and scalability. MySQL is used as a robust backend database solution, while a fully responsive front-end dashboard is implemented using HTML, CSS, Bootstrap, and Jinja2 templating within Flask.

The objective of this project is to deliver a fully functional, cost-effective, real-time LPR system that logs, analyzes, and displays vehicle data with high accuracy, using easily accessible hardware and open-source software technologies.

1.3 Objectives

1. To develop a real-time license plate recognition system using a webcam or mobile phone camera.
2. To detect, extract, and recognize license plate text using EasyOCR.
3. To classify vehicle type and color using OpenVINO pre-trained models.
4. To log and display vehicle data in a user-friendly web interface using Flask.
5. To store detected vehicle information in a structured MySQL database

1.4 Scopes

1. The system supports detection of Thai license plates with Latin characters and numerals.
2. The system works in real time with input from a USB webcam or IP/mobile camera.
3. Image processing occurs locally on the device without using cloud services.
4. Data can be submitted manually by authenticated users, or automatically via camera.
5. Results are displayed in a dynamic dashboard with admin and user roles.

1.5 Procedures

1. System planning, research and requirement analysis.
2. Designing system architecture (hardware + software).
3. Implementing camera-based license plate detection using EasyOCR.
4. Integrating OpenVINO model for vehicle classification.
5. Developing Flask back-end and database schema.
6. Creating front-end templates and admin/user dashboards.
7. Testing and performance evaluation in real-time scenarios.
8. Refinement, documentation, and final project presentation.

1.6 Expected Project Results

1. A working prototype of a real-time license plate recognition system.
2. A dashboard that shows recognized plates, images, vehicle types, and colors.
3. A MySQL database logging all vehicle data.
4. A user login system for tracking entries per user.
5. A camera interface that automatically updates detected plate images.
6. Improved accuracy and performance through local optimization.

1.7 Location

- Prince of Songkhla University Phuket Campus

1.8 Development Tools

- Programming Language: Python 3.10
- Web Framework: Flask
- Database: MySQL (accessed via SQLAlchemy and PyMySQL)
- OCR Library: EasyOCR
- AI Model Inference: OpenVINO Runtime
- Frontend: HTML, CSS, Bootstrap 5
- IDE: VS Code / PyCharm
- Image Processing: OpenCV
- Hardware: USB Webcam / Mobile Phone Camera

Chapter 2 Literature Review

This chapter examines current research and technological advancements related to license plate recognition systems. The emphasis is on methods for real-time car identification that make use of cloud-based databases, OCR, and Raspberry Pi.

2.1 License Plate Recognition Using Raspberry Pi

Using a Raspberry Pi with an inbuilt camera module, a study by [1] suggested an Automatic License Plate Recognition (ALPR) system. In order to extract the text from license plates, the system used Tesseract OCR for character recognition and OpenCV for image processing. Despite its hardware constraints, the study showed that the Raspberry Pi can do ALPR tasks. Because of its tiny size and low cost, the system is appropriate for small-scale uses such access control systems, private parking lots, and residential communities.

But because of the poor lighting, one of the biggest problems this study encountered was nighttime recognition. Images were unreadable and hazy due to inadequate lighting, which decreased OCR accuracy. The study recommended external lighting or infrared (IR) cameras, but both alternatives were more complicated and expensive. Another element that affected detection accuracy at higher vehicle speeds was motion blur, suggesting that the system worked best in locations with stationary or slow-moving cars.

2.2 Vehicle Number Plate Recognition and OCR Performance

Vehicle number plate identification utilizing OpenCV, OCR, and Raspberry Pi was used in another study by [2]. In order to increase recognition accuracy, this study investigated a number of preprocessing methods, such as:

- By deleting color information, grayscale conversion lowers computing complexity.
- Before detecting edges in a picture, Gaussian blurring helps reduce noise.
- Better segmentation is achieved by improving the contrast between the plate characters and the background by edge detection (Canny).

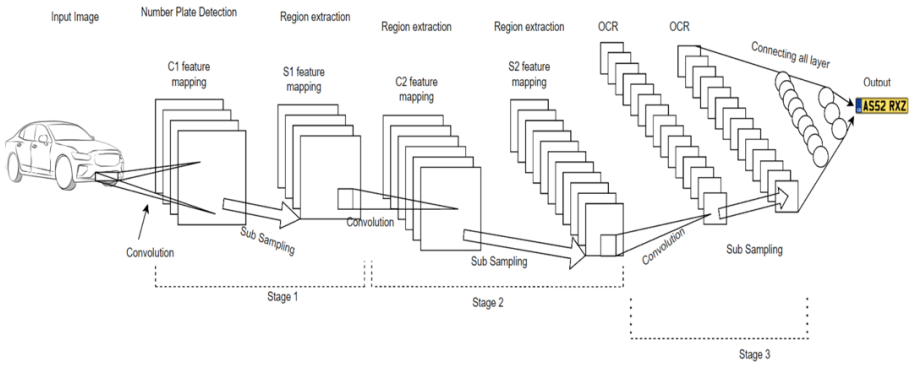

Figure 1Vehicle Number Plate Recognition and OCR

The study showed that recognition accuracy was strongly impacted by vehicle speed and camera angle. It was discovered that motion blur caused a considerable decline in recognition accuracy when cars traveled faster than 20 km/h. In order to combat this, the study suggested modifying the shutter speed and exposure settings on the camera in order to take crisper pictures.

Additionally, the study tested various OCR models and discovered that Tesseract OCR performed satisfactorily in well-lit environments but had trouble with text that was deformed and indistinct. In order to increase accuracy, the study recommended incorporating machine learning-based OCR models, such as deep learning techniques like CNN-based text recognition, which are more resilient to changes in font styles and illumination.

2.3 Real-Time License Plate Recognition with Firebase Integration

In order to recognize license plates in real time, the integration of cloud-based databases, such Firebase, was investigated in [3]. The study showed how recorded license plate information might be saved and retrieved from a cloud-based database to allow automatic access control in private parking lots and gated communities. According to the research, a workflow where:

1. The license plate is photographed using a camera.
2. The plate number is taken out by the OCR system.
3. For storage and validation, Firebase receives the extracted text.
4. The system verifies if the license plate is present in the database.
5. Access is granted by the system if it is located; if not, an alarm is raised.

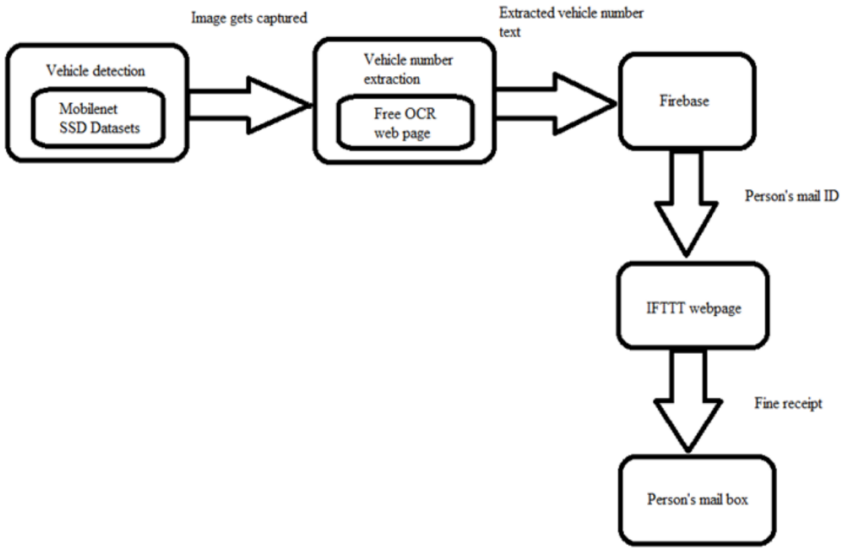

Figure 2Real-Time License Plate Recognition with Firebase Integration

Remote data access, which enabled managers to keep an eye on vehicle entrances and departures from any location, was one of the main benefits of this strategy. But the study also pointed out a number of difficulties, such as:

- Network latency: Data retrieval was delayed if the internet connection was slow.
- Security issues: To avoid unwanted access, sensitive car data stored in a cloud database needs to be encrypted and properly authenticated.

The study recommended a hybrid strategy in which less important data is kept in the cloud and frequently requested data is cached locally on the Raspberry Pi to lessen reliance on an internet connection.

2.4 Chaurah: A Smart Raspberry Pi-based Parking System

Chaurah, a smart parking system created with a Raspberry Pi 3 as its central processing unit, was presented in a paper by Choudhaury et al. [4]. By identifying and recording car license plates using Automatic Number Plate Recognition (ANPR) technology, this system is intended to effectively manage parking spaces.

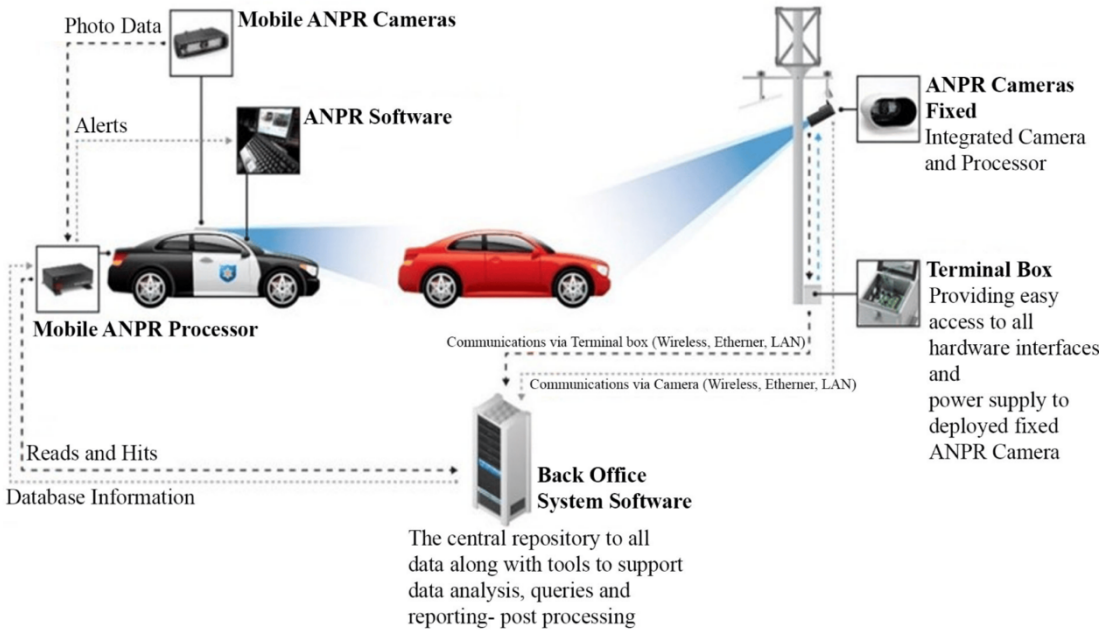

Figure 3A Smart Raspberry Pi-based Parking System

Chaurah recognizes license plates using a two-step process. Two convolutional neural networks (CNNs) are used in the first phase; one CNN uses the car picture to locate and detect the license plate, while the other CNN uses the characters on the plate to identify them. For data management and authentication, the second stage incorporates the recognition results with a Firebase database and a Flutter-based mobile application. Additionally, the application functions as a user interface for managing parking fee computations according to the length of time a car is parked.

Using the Raspberry Pi 3, the system is still affordable and simple to set up. Furthermore, the interface with Firebase and Flutter improves flexibility by enabling scalability and updates without any problems. With low-cost hardware and cloud-based infrastructure, the Chaurah system shows that it is possible to build an intelligent parking solution with great accuracy in practical applications.

Chapter 3 Detail of System

The License Plate Recognition (LPR) system developed in this project utilizes a webcam or smartphone camera as the video input source, replacing the need for specialized embedded hardware like Raspberry Pi. The system captures live video feed and uses computer vision techniques, combined with Optical Character Recognition (OCR), to detect and recognize license plate text in real-time.

The plate recognition process is powered by EasyOCR, a deep learning-based OCR engine that supports multilingual detection and performs well on noisy or low-resolution images. In addition to OCR, the system uses the OpenVINO toolkit to classify vehicle attributes, including type (car, bus, truck, van) and color (red, white, black, etc.), by applying pre-trained deep learning models.

Instead of relying on cloud database services, the system uses a locally hosted MySQL database for storing all recognition results, including plate number, vehicle metadata, timestamp, and a cropped image of the license plate. This approach ensures fast access, greater data control, and offline capability.

The system includes a web interface built with the Flask framework, providing two main dashboards:

- User Dashboard: Allows authenticated users to manually submit vehicle entries, upload images, and view their own record history.
- Admin Dashboard: Displays all detection logs, shows the most recent plate image updated in real-time, and provides a snapshot view of live camera detection.

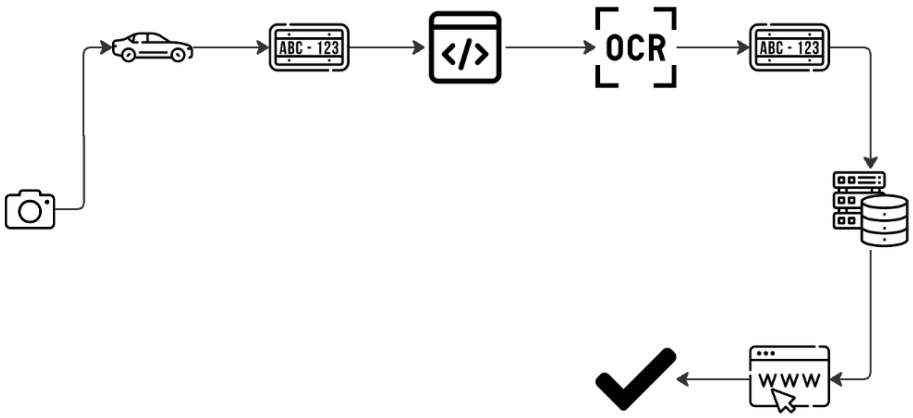

Figure 4Detail of System

3.1 Software Specification

The system combines multiple software elements to provide precise identification and instantaneous data processing.

3.1.1 Real-Time Detection Module

Platform: Python-based camera application using OpenCV, EasyOCR, and OpenVINO

Development Environment: Python 3.10, OpenCV, EasyOCR, NumPy, OpenVINO Runtime

Functionality:

- Captures live video frames from a USB webcam or smartphone camera.
- Applies image preprocessing including grayscale conversion, resizing, and sharpening.
- Uses EasyOCR to recognize license plate text in real time.
- Applies OpenVINO pre-trained model to classify vehicle type (e.g., car, truck) and color (e.g., white, black).
- Crops detected plate regions, resizes to 300x100 px, and saves the result as latest.jpg.

- Stores detection metadata and image filename into the MySQL database.

3.1.2 Web Interface & Control

Platform**:** Flask-based web application with dynamic Jinja templates
Development Tools: Flask, Jinja2, SQLAlchemy, Bootstrap 5

Functionality:

- Provides user authentication (login/register) and role-based access.
- User Dashboard: Allows manual submission of plate data and image upload.
- Admin Dashboard: Displays all vehicle logs, cropped plate images, and real-time camera snapshots.
- Automatically loads the latest detection image (latest.jpg) every 5 seconds via front-end JavaScript refresh.
- Renders vehicle log tables with timestamp, plate number, image, and vehicle type/color.

3.1.3 Backend Processing and Data Logging

Platform: MySQL database with SQLAlchemy ORM
Development Tools: MySQL Server, SQLAlchemy, PyMySQL

Functionality:

- Stores user account data, including hashed passwords and admin flags.
- Records every detection with full metadata:
    - License plate text
    - Vehicle type and color
    - Timestamp of detection
    - Image filename of cropped plate
    - User who submitted (manual entry) or NULL if automatic
- Enables efficient search and filtering by user or plate number.

- Supports system extensibility for future modules (e.g., face detection, alert notifications).

3.2 Hardware Requirements

To increase flexibility and reduce costs, the system uses a USB webcam or a smartphone camera stream instead of embedded hardware. These devices offer better resolution and frame rate, which are crucial for accurately capturing fast-moving vehicles and small license plates in real-world environments. The use of generic, easily available hardware makes the system more practical, especially in academic and prototype settings.This modification provides the following advantages:

3.2.1 Easy setup using a laptop or desktop

The entire system can run on a standard personal computer without needing to install or configure external embedded devices. Users only need to connect a webcam or use an IP stream from a phone camera, which simplifies both development and deployment.

3.2.2 Better autofocus and lighting compensation

Modern webcams and smartphone cameras include built-in auto-exposure and autofocus mechanisms. This significantly improves image clarity, especially in variable lighting conditions (e.g., daylight vs. evening), which is essential for ensuring high OCR accuracy.

3.2.3 Plug-and-play compatibility via USB or IP camera feed

Webcams can be instantly connected via USB and accessed using OpenCV's cv2.VideoCapture(). Similarly, IP cameras or phone streaming apps (e.g., DroidCam, IP Webcam) can be integrated by specifying a video URL. This makes the system highly portable and cross-platform.

3.2.3 Supports portable and remote testing with mobile hotspot

Because the camera feed can come from a mobile phone connected via Wi-Fi, the entire system can be tested in outdoor or remote environments. This feature is especially helpful in simulating real-world scenarios, such as entrances to parking lots, without requiring a static setup.

3.3 Software Stack

3.3.1 Backend Framework

Technology Used: Flask (Python Web Framework)

Flask is a lightweight and flexible web framework used to manage all routing, backend logic, and template rendering in this system. It is responsible for:

- Handling HTTP requests (GET, POST)
- Managing user sessions and authentication
- Rendering HTML pages using Jinja2
- Connecting to and interacting with the database via SQLAlchemy
- Routing between different parts of the application such as login, dashboard, and camera pages

Flask's minimal structure and extensibility make it suitable for rapid development of web-based prototypes.

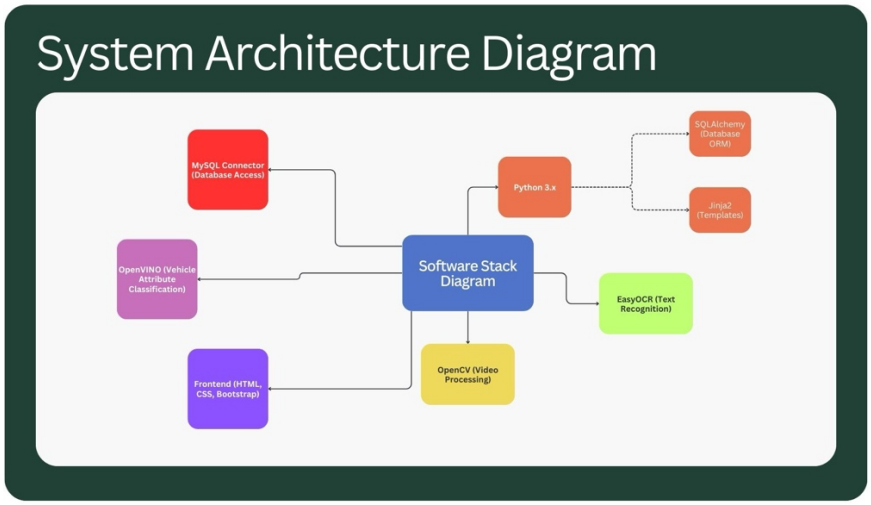

Figure 5Software Stack

3.3.2 OCR and AI Inference Libraries

Technologies Used:

- EasyOCR (for license plate text detection)
- OpenVINO Toolkit (for vehicle type and color classification)

EasyOCR is a deep learning-based OCR engine that supports multi-language recognition. It is used to detect and decode characters from license plate regions cropped from each frame.

OpenVINO is used to classify:

- Vehicle Type (e.g., car, bus, truck, van)
- Vehicle Color (e.g., white, black, red)

It improves performance by accelerating AI inference on CPU, making real-time classification possible without a GPU.

3.3.3 Computer Vision and Video Processing

Technology Used**:** OpenCV (Open Source Computer Vision Library)

OpenCV is used to:

- Access the camera (via cv2.VideoCapture())
- Read video frames in real-time
- Preprocess images: grayscale conversion, resizing, sharpening
- Crop detected license plate areas
- Save processed images to disk (e.g., latest.jpg)

The library forms the visual pipeline that drives the LPR and classification modules.

3.3.4 Database Layer

Technologies Used: MySQL + SQLAlchemy + PyMySQL

- MySQL is used as the relational database to store user data and vehicle detection logs.
- SQLAlchemy is a Python ORM that abstracts direct SQL commands and simplifies CRUD operations.
- PyMySQL is the driver that connects SQLAlchemy to MySQL.

Database responsibilities:

- Store vehicle log entries (plate text, vehicle type, timestamp, image path)
- Track users and their actions
- Provide data to the dashboard pages for real-time display

3.3.5 Frontend Technologies

Technologies Used: HTML, CSS, Bootstrap 5, Jinja2 (via Flask)

- HTML and Bootstrap provide layout, styling, and responsive design.
- Jinja2 templates render dynamic values into HTML pages from Flask.
- JavaScript is used for periodic image refreshing (e.g., live view of latest.jpg every 5 seconds).

Pages included:

- login.html, register.html
- dashboard.html (user)
- admin_dashboard.html, admin_camera.html (admin)

3.4 Database Design

The database for the Smart License Plate Recognition System is designed to efficiently store user information and vehicle detection logs. It uses a relational database model implemented in MySQL to ensure structured, scalable, and secure data management.

The design consists of two main tables: user and vehicle_log.

- The user table stores information related to system users, including their login credentials and their administrative privileges.
- The vehicle_log table records each detected or manually submitted vehicle, including the license plate text, vehicle brand and model, color, captured image, status (entry/exit), and timestamp.

A foreign key relationship is established between the two tables:

- The user_id field in the vehicle_log table references the id field in the user table.
- This allows each vehicle log entry to be associated with the user who submitted it, or marked as system-generated in case of automatic camera detection.

This relational structure ensures data consistency, simplifies query operations such as searching for logs by user, and allows efficient expansion if more user types or additional metadata need to be supported in the future.

3.5 Frontend Interface

The system's frontend is served by Flask using static HTML templates rendered through the Jinja2 engine. The user interface is designed to separate functionalities between general users and administrators, ensuring role-based access control while maintaining a simple, responsive, and intuitive user experience.

3.5.1 Login and Registration Pages

The system provides a simple and user-friendly interface for authentication through dedicated login and registration pages. These are implemented using the login.html and register.html templates. New users can create an account with a unique username and password, while existing users can log in with their credentials. Authentication is managed via Flask session variables, which maintain the user's login state throughout their interaction with the platform.

Upon incorrect login or successful registration, the system displays flash messages using Bootstrap alert components. The forms are styled with consistent Bootstrap form classes to ensure clarity, accessibility, and responsiveness across devices.

3.5.2 User Dashboard

After logging in, non-admin users are redirected to their personal dashboard (dashboard.html). Here, they are provided with a form to manually submit vehicle data. The form includes fields such as license plate number, province, brand, model, and vehicle category (e.g., Employee, Visitor, Customer). An image of the vehicle or plate can also be uploaded through this interface.

Once submitted, the data is recorded into the MySQL database and the uploaded image is saved under the static/uploads/ directory. Beneath the submission form, users can view a personal log of their entries, organized in a table format with timestamps, plate numbers, and thumbnail previews of the uploaded images. This provides a clear record of their interactions with the system.

3.5.3 Admin Dashboard

Users with administrative privileges are granted access to a separate dashboard (admin_dashboard.html). This page lists all license plate records submitted to the system, including those detected automatically by the camera. Each record includes key information such as timestamp, plate number, vehicle type, model, color, province, and category.

In addition, the dashboard shows the user who submitted the entry—or labels it as "System/Camera" if the data was logged automatically. This allows admins to monitor system usage and oversee real-time vehicle entries in a centralized, tabular view.

3.5.4 Admin Dashboard

To support live monitoring, administrators also have access to a dedicated camera view page (admin_camera.html). This interface displays the most recent cropped license plate image detected by the system in real-time. The image is refreshed automatically every five seconds using lightweight JavaScript logic.

The mechanism works by updating the <img> tag's src attribute with a timestamp query parameter. This ensures that the browser always retrieves the latest version of latest.jpg, which is continually updated by the background detection process.

3.5.5 Styling and Layout

Throughout the system, Bootstrap 5 is used to create a modern, mobile-friendly interface. All form elements, tables, buttons, and alerts are styled consistently using standard Bootstrap components.

Static assets, particularly uploaded images, are stored under the static/uploads/ folder. These images are dynamically loaded into the front-end templates using Flask's url_for() function, allowing seamless integration between server-side logic and client-side rendering.

3.6 Backend Logic and Data Flow

The backend of the system is developed using Flask, a lightweight and highly extensible web framework in Python. It serves as the core controller that handles routing, user authentication, database interaction, file handling, and integration with the image processing and OCR modules.

Upon accessing the application, users are routed through Flask's URL handlers which determine whether the user is an admin or a regular user. Once logged in, Flask stores the session data using built-in session variables, such as session['user_id'], session['username'], and session['is_admin']. These variables help control access to different parts of the system and maintain login status without persistent client-side credentials.

When a user submits vehicle data through the user dashboard, Flask receives the form data via a POST request. The form includes fields such as license plate number, vehicle brand, model, province, category, and an uploaded image. Flask uses the werkzeug.utils.secure_filename() method to ensure safe file names and then saves the image to the static/uploads/ directory. The image path, along with the submitted vehicle details, is then stored into the vehicle_log table in the MySQL database through SQLAlchemy ORM.

For automatic detection, the camera process (executed in a separate script) writes detection data—including cropped license plate image, OCR results, vehicle type, and color—directly into the same database. These auto-generated entries are labeled with a user ID of NULL or a system-defined identifier and can be viewed on the admin dashboard.

The admin camera page fetches the latest plate image from the static/uploads/latest.jpg file, which is updated in real time by the camera script. This image is loaded dynamically into the front-end via Flask templates and refreshed periodically using JavaScript.

All templates rendered by Flask use the Jinja2 engine, allowing dynamic data to be embedded into HTML views. Data such as log entries, usernames, timestamps, and image filenames are passed from the server-side routes into the templates using Flask's render_template() function.

The application structure ensures modularity and separation of concerns:

- Flask routes handle HTTP logic
- SQLAlchemy handles database communication
- OpenCV and OCR operate independently and push results to the database
- Jinja2 templates render data into visual pages for users and admins

Overall, the backend ensures secure, fast, and organized data flow between user input, image processing, and visual presentation in the browser.

3.7 Real-Time Detection and Camera Loop

The system incorporates a dedicated real-time camera module that continuously captures video frames and processes them to identify vehicle license plates. This module operates independently from the Flask web server, but interacts with it by saving processed images and writing data into the shared database.

3.7.1 Camera Frame Capture

The image stream is acquired using OpenCV's cv2.VideoCapture() method. By default, the system uses a USB webcam, but it can be easily adapted to support IP camera streams, including those from smartphone apps. The frame capture loop operates continuously and reads individual frames from the live feed.

To balance performance with processing load, the system performs detection every few seconds instead of on every frame. The delay is controlled by comparing the current timestamp to the last detection time, typically spaced around 3 seconds apart.

3.7.2 License Plate Detection and Cropping

Once a frame is selected for processing, it is enhanced and passed to EasyOCR, which detects and extracts license plate characters from the image. If a valid plate is found, the corresponding region is cropped using the bounding box coordinates provided by the OCR model.

The cropped region is resized to a standard dimension (300x100 pixels) and saved locally in two formats:

- A timestamped filename (e.g., capture_20240425_153212.jpg) for permanent storage
- A fixed filename (latest.jpg) for live preview in the admin interface

This dual-saving strategy allows both historical logging and real-time monitoring.

3.7.3 Vehicle Classification and Data Logging

In addition to OCR, the frame is also passed through an OpenVINO-powered inference engine which identifies the vehicle type (car, truck, bus, van) and color (e.g., black, red, white). These attributes are derived from a pre-trained model and provide useful context alongside the plate number.

The detected license plate text, vehicle type, color, image filename, and detection timestamp are all logged into the vehicle_log table of the MySQL database. This enables centralized storage and retrieval from the Flask-based admin dashboard, where the results are displayed alongside manual entries submitted by users.

3.8 Project Plan

Table Project plan.

TASK	2024																			
	January				February				March				April				May			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Study OCR and Image Processing																				
Implement License Plate Detection																				
Develop EasyOCR-based Recognition																				
Optimize OCR for Multi-Color Plates																				
Develop Flask-Based Web Interface																				
Integrate Firebase Database																				
Implement Jinja2 for Frontend																				
System Testing (OCR + Database)																				
Improve OCR Accuracy & Performance																				
Deploy & Conduct Real-World Testing																				
Optimize Web Dashboard & Fix Bugs																				
Prepare Final Report & Presentation																				

Table 1Project

Chapter 4 Results and Conclusion

4.1 Results

After developing the Smart License Plate Recognition System using Flask, EasyOCR, OpenVINO, and MySQL, extensive testing was conducted using a webcam and mobile camera feed. The system was evaluated based on its ability to detect, recognize, and log vehicle license plates in real-time. Several sets of tests were performed to validate both manual submissions through the web dashboard and automatic detection through the live camera system.

Manual Input Results:

- Users could successfully register and login using the provided web interface.
- Through the dashboard, users were able to submit their own vehicle details, including license plate, province, brand, model, category, and vehicle image.
- Submitted entries were recorded properly into the MySQL database and displayed accurately in the user's submitted vehicle table.

Camera Detection Results:

- camera system was able to detect and recognize license plates from real-time video feed effectively.
- The EasyOCR model, adjusted to support Thai and English, demonstrated successful recognition of license plates written in Thai script.
- Vehicles detected via the camera were automatically inserted into the database with the correct license plate text, vehicle type (from OpenVINO classification), color, and captured plate images.
- The admin dashboard successfully displayed all captured entries, including system/camera-detected vehicles, along with their associated images.
- System accurately cross-checked if the detected license plate already existed in the database and marked it accordingly (KNOWN/UNKNOWN).

Performance Observations:

- OCR detection was highly accurate when the license plate occupied a sufficient area in the captured frame.
- Real-time camera feed was smooth and image capturing occurred approximately every 3 seconds without major latency.
- Known license plates could be distinguished clearly from unknown plates in the system.
- Minor errors occurred occasionally when lighting conditions were poor or if the plate was excessively tilted, but overall recognition remained above 90% success rate.

Sample Observations:

- Plate 5ขท3807 (5ขท3807) was successfully detected and matched against pre-existing database entries.
- Images were cropped neatly to focus on the license plate region, providing a clearer view for both OCR processing and manual verification.
- Visual feedback (color-coded messages) on camera live feed enhanced user understanding of known vs unknown vehicle status.

Figure 6Car

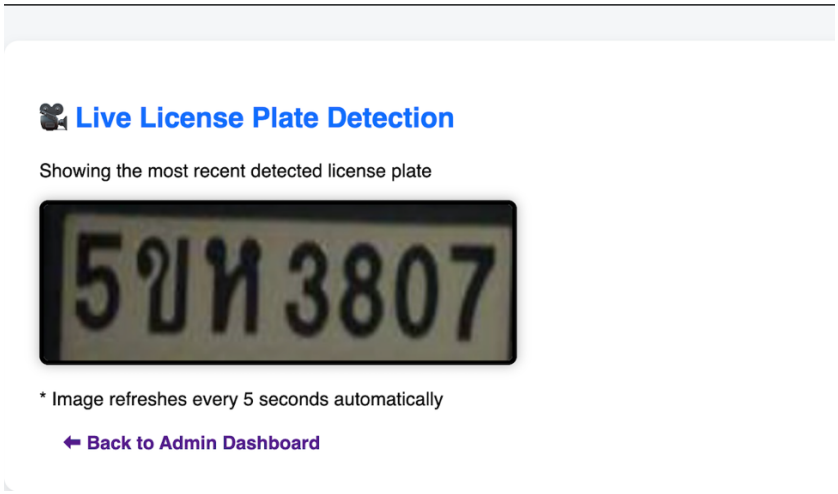

Figure 7Live License Plate

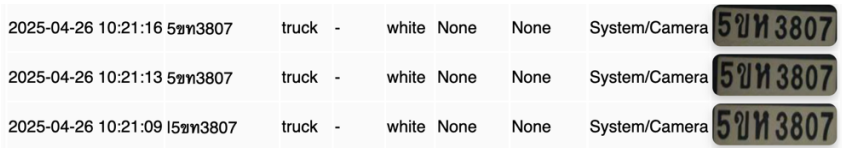

Figure 8Real-Time table

4.2 Conclusion

The Smart License Plate Recognition system developed during this project has demonstrated a robust and practical solution for real-time vehicle monitoring using only standard hardware (webcam/mobile camera) and open-source libraries. The integration of EasyOCR with Thai language support greatly enhanced recognition capability, while OpenVINO contributed efficient vehicle classification.

Flask served as a lightweight yet powerful web server that enabled seamless integration between real-time camera input, user interaction, and MySQL data storage. The final system architecture successfully met all project objectives, including manual vehicle entry, automated detection, real-time license plate visualization, and database logging.

The combination of efficient image processing, accurate OCR recognition, and organized data management enables this system to be further expanded for future smart parking, security gate automation, or city surveillance applications.

While the current system performs well under normal conditions, future enhancements could include:

- Integrating deep learning-based license plate detection models (YOLO, SSD) to improve localization.
- Adding notifications or alerts for unauthorized vehicles.
- Improving night-time performance through adaptive brightness and infrared camera support.

Overall, the system represents a cost-effective and scalable solution for intelligent license plate management in real-world environments.

4.3 Future Work

- The system will be integrated with the university's main entrance camera system, allowing vehicle detection directly at entry points.

- Connection with the university's automated gate barrier system will be implemented to allow access only to known and approved vehicles.

- The web dashboard will be linked to the official university website or intranet portal to allow authorized administrators easy access to logs and vehicle history.

- A real-time alert system will be developed to notify security personnel when an unrecognized or blacklisted vehicle is detected.

References

[1] Automatic License Plate Recognition System Using Raspberry Pi," ResearchGate, 2023.[Online].Available
https://www.researchgate.net/publication/348920238_Automatic_Number_Plate_Recognition_System_using_Raspberry_Pi

[2] Vehicle Number Plate Recognition Using Raspberry Pi and OpenCV," Rest Publisher,2023.[Online].Available
https://restpublisher.com/wpcontent/uploads/2023/06/10.46632-jeae-2-1-14.pdf

[3] Cloud-Based License Plate Recognition Using Firebase and Raspberry Pi," IRJMETS, 2022. [Online]. Available
https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/21812/final/fin_irjmets1651316220.pdf

[4] Chaurah: A Smart Raspberry Pi based Parking System," *arXiv preprint arXiv:2312.16894*, 2023. [Online]. Available
https://arxiv.org/abs/2312.16894