Applied Information Technology And Computer Science Vol. 5 No. 1 (2024) 1171-1190

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Deep Learning Development for Automatic License Plate Detection with Text Extraction Capability

Sabri Nasser Hussein Murshed Dokhan[1], Nureize Arbaiy[1*]

[1]Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2024.05.01.066
Received 20 June 2023; Accepted 18 June 2024; Available online 30 August 2024

Abstract: Vehicle identification and tracking are essential in traffic surveillance systems, which prioritize safety and efficiency. Security personnel still manually observe medium-sized structures. Security officers physically record license plate numbers after visually examining passing automobiles. This process is time-consuming and error prone. This project aims to create a web-based deep learning system for automated license plate identification with text extraction at UTHM, Malaysia. A variety of car license plates will train the system. A convolutional neural network will immediately detect and interpret license plate text. The user may submit a vehicle photo to the online interface, and the system will recognize the license plate number and return a picture with the highlighted plate. The system will be tested on a set of concepts using accuracy and precision criteria. Traffic management and vehicle identification might employ the created technology.

Keywords*: Automatic License Plate Detection with Text Extraction Capability*

1. Introduction

Vehicles are widely used in many aspects of industry as well as in daily life. In the vehicle registry of the issuing region, the registration identifier—a numeric or alphanumeric ID—uniquely identifies the car or its owner. Vehicle detection and tracking are important and successful in the field of traffic surveillance systems, where effective traffic management and safety are the primary concerns [1]. The vehicle License Plate (LP) number provides an effective and unique method of identifying vehicles [2]. Traffic offenses are becoming more obvious to the public for each car as the number of vehicles continues to rise. Traffic offences, including highway or parking toll fraud, speeding, and vehicle theft. So, identifying the vehicle number is important for that purpose [3]. Vehicle security information received from LPs can be used in a variety of ways, including access and flow management at tolls and border crossings, locating suspicious vehicles, and even fighting crime [4]. The newest system of Vehicle Number Plate Recognition (NPR), also known as Registration Plate Recognition (RPR) or

**Corresponding author: nureize@uthm.edu.my*

License Plate Recognition (LPR), employs AI, ML, deep learning, and computer vision-powered technologies to read such license plates on automobiles without requiring human intervention [5].

The case study in this project is conducted at UTHM Security Department. The Security Department serves as the University's security management Centre and is committed to raising service standards so that all employees and students can carry out their assigned tasks and responsibilities without being concerned about security issues on campus. One of the functions of this department is to monitor traffic safety in the campus area. A system for automated license plate recognition (ALPR) may be required by the department for several reasons. One rationale might be to increase campus security by allowing the department to swiftly and precisely identify vehicles that are not permitted to be on campus. Vehicles that have been stolen or whose owners have been prohibited from the campus for safety or security purposes may fall under this category. An ALPR system could also be used to track the movements of vehicles on campus, potentially helping to identify patterns or behaviors that may be of concern to the security department. Additionally, an ALPR system could be used to enforce parking regulations on campus, helping to ensure that all vehicles are properly registered and parked in designated areas. Overall, an ALPR system could help the UTHM Security Department to carry out its responsibilities more effectively and efficiently, improving the overall safety and security of the campus community.

Without more information about the specific practices and procedures of the UTHM Security Department, it is difficult to accurately describe the current method they use to detect license plate numbers. However, in general, there are several methods that security departments or law enforcement agencies might use to identify and track vehicles. One common method is manual observation, where security personnel or law enforcement officers visually scan the plates of passing vehicles and record the numbers manually. This can be time-consuming and prone to errors, especially if the plates are dirty or difficult to read. Another method is the use of handheld scanners or mobile devices equipped with specialized software and cameras, which can automatically capture and process plate numbers. While these methods can be effective in certain circumstances, they may not be as efficient or reliable as an automated license plate recognition (ALPR) system, which uses specialized cameras and software to automatically capture and process plate numbers in real-time.

There are several potential difficulties that may be encountered when using manual or semi-automated methods to detect license plate numbers. One challenge is the time and labor required to manually scan and record the numbers of passing vehicles. This can be especially challenging if the security department or law enforcement agency is responsible for covering a large area or if there is a high volume of traffic. Another challenge is the potential for human error, as manual observation or the use of handheld scanners may be prone to mistakes or inaccuracies. Additionally, manual, or semi-automated methods may not be effective in low light or adverse weather conditions, which can make it difficult to accurately read or capture plate numbers. Overall, these difficulties may make it challenging for the UTHM Security Department to accurately and efficiently detect and track vehicles using current methods. An automated license plate recognition (ALPR) system could potentially help to overcome these challenges by automating the process of capturing and processing plate numbers in real-time.

A web-based automated license plate recognition (ALPR) system can provide a solution to the difficulties of manually detecting and tracking license plate numbers. ALPR systems use specialized cameras and software to automatically capture and process plate numbers in real-time, eliminating the need for manual observation or the use of handheld scanners. The system can be accessed through a web-based interface, allowing authorized users to view and analyze data from anywhere with an internet connection. With an ALPR system in place, the UTHM Security Department could potentially improve the efficiency and accuracy of its vehicle tracking and identification processes, potentially helping to enhance the overall security and safety of the campus.

This paper is divided into five sections. The project's background is described in Section 1, and the related works are discussed in Section 2. The approach is discussed in Section 3, and the results and discussion are summarized in Section 4. The conclusion is given in Section 5.

2. Related Work

This section explains the literature review on previous work related to vehicle plate number detection system, algorithm model, and observes related concerns in previous work.

Automated number plate recognition (ANPR) systems scan car registration plates using optical character recognition (OCR) and other image processing methods (Patel et al., 2013). One of the most accurate and widely used computer vision technologies is ANPR. The methods used to enhance automatic license plate recognition software's functionality, accuracy, cost-effectiveness, durability, and scalability are constantly evolving [6].

ANPR frequently uses a variety of various techniques that are used as part of the computer vision pipeline. The steps required to transform input from images or videos into useable, complete data are part of the vision pipeline [7]. These are the most important components: object detection in real time. Deep learning is used for object recognition to recognize automobiles in video stream images, including different vehicle classifications (bus, truck, car, van, motorcycle, etc.). Modern object recognition techniques, including YOLOv5 and YOLOv7, utilize neural networks trained on a collection of images [8].

Automatic Number Plate Recognition (ANPR) is a technology used to recognize and extract information from vehicle license plates. It involves two main steps: detection, in which the system identifies the location of the license plate in an image or video frame, and recognition, in which the system reads and translates the characters on the license plate into text. The output of an ANPR system is usually the license plate number, along with a region or country identifier. ANPR systems can be customized for different character sets and can be used in a variety of applications, such as checking a license plate against databases of registered plates, whitelisted and blacklisted plates, and returning recorded information about a car, such as the registered owner's name and address. While ANPR can be effective, it can be improved with the use of deep learning techniques, which are a type of artificial intelligence that uses multiple layers to analyze and learn from large amounts of data. Deep learning is often used in real-time applications and has a high level of accuracy.

The well-known discriminative deep learning architecture, Convolutional Neural Network (CNN) or ConvNet learns directly from the input without the need for human feature extraction [9]. The design of traditional ANNs, such regularized MLP networks, is therefore enhanced by CNN. Every CNN layer takes model complexity into account while also considering the ideal parameters for a meaningful output. CNNs are widely utilized in visual identification, medical image analysis, image segmentation, natural language processing, and a few other applications since they are built to handle a variety of 2D shapes. Since it can automatically identify important properties from the input without human involvement, it is more effective than a typical network [10]. According to their capacities for learning, several CNN iterations—including visual geometry group (VGG), AlexNet, Xception, Inception, ResNet, etc.—can be used in a variety of application domains.

CNNs are made to analyze data that has a grid-like architecture, like an image, and they can recognize and learn from patterns and characteristics in this data. CNNs can be used for a variety of tasks, such as object detection, facial recognition, and picture and video classification. They are especially helpful for jobs that require a lot of data and complicated patterns, like classifying images and videos, since they can learn from new data over time and adapt to it, which improves their performance and accuracy as they process more and more data.

CNNs are used to evaluate and categorize photos and videos in the field of image and video classification based on the patterns and features that they contain. This can be used to categorize the image or video's content as well as identify objects, persons, and other elements in the image or video (e.g., as a photo of a cat or a video of a sporting event). CNNs are employed in the field of object detection to find and categorize items in an image or video. The presence and location of things like pedestrians, cars, or other interesting characteristics can be determined using this. CNNs are employed in the field of facial identification to examine and pinpoint each person's particular facial features. Applications like security and identity verification can make use of this. Figure 1 shows the flowchart of CNN and its pseudocode algorithm in Figure 2.

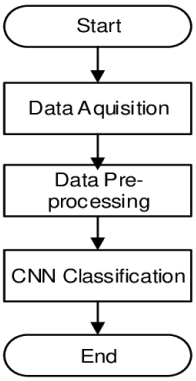

Figure 1: CNN Flowchart

```
PSEUDOCODE OF CONVOLUTIONAL LAYER
for (l = 0;l<L ;l ++){
    for (m = 0;m<M ;m ++){
        for (n = 0;n<N ;n ++){
            sum = bias[l];
            for (k = 0;k<K ;k ++){
                for (s1 = 0;s1 <S1;s1 ++){
                    for (s2 = 0;s2 <S2;s2 ++){
                        sum+=weight[k][l][s1][s2] × input [k][m + s1][n + s2];
            }}}
        output [l][m][n] = activation_func(sum);
}}}
```

Figure 2: CNN Pseudocode

The three systems [11], [12], [13] under study were constructed by a group of researchers and subject-matter specialists, resulting in a system that is nearly flawless, technologically advanced, and equipped with all necessary characteristics. Some of these characteristics are used as a guide in completing the recommendation system. The proposed system will serve as the foundation for any future system modifications that might include new functionality. Table 1 provides the summary.

Table 1: System's Comparison

System / Features	SentiVeillance	Rekor Scout™	DTK LPR Solution	ALPR System (UTHM)
Log In interface	√	√	√	√
LP detector	√	√	√	√
Real-time recognition	√	X	√	√
Algorithm	CNNs, RNNs	CNNs	CNNs	CNNs
Cost of the software	Above $5000 per year	Paid solution	Paid solution	Prototype
Data destination	Cloud Storage, Local database	Cloud storage	Cloud Storage, Local database	Local database
Vehicle color	X	X	√	X

Vehicle make	√ not 100% accurate	X	X	X
Vehicle model	√ not 100% accurate	X	X	X

3. Methodology

This section explains the deep learning process used in this project study.

3.1 Deep Learning Methodology

The process for creating a deep learning model is described in the following.

1. Data collection and preparation:

Collecting and preparing data is a crucial step in developing a deep learning-based automatic license plate recognition (ALPR) system. Data collection and preparation steps are as follows:

- Collect a large dataset: To train a deep learning model, we need a large dataset of images of license plates. The more diverse the dataset, the better the model will be able to generalize to real-world situations. In this project, a quite a good number of car images that contain license plates has been collected.
- Label the data: To train a deep learning model, label each image with the corresponding license plate number. In this project the image labelling has been done using an automated labelling tool (labelImg).
- Split the data into train, validation, and test sets: To evaluate the performance of the model, it is common practice to split the dataset into a training set, a validation set, and a test set. The model will be trained on the training set, and its performance will be evaluated on the validation set. The final performance of the model will be evaluated on the test set. In this project the dataset has been split into 70% training set, 25% validation set and 5% for test set.

2. Data Pre-processing is an important step in developing a deep learning-based automatic license plate recognition (ALPR) system. Data pre-processing are summarized as follows:

- Resize the images: It is important to resize the images to a consistent size before training a deep learning model. This is because the model expects input images to be a certain size, and resizing the images ensures that they meet this requirement. In this report the images have been resized to height and width (640,640) pixels.
- Normalize the pixel values: Normalizing the pixel values is important to ensure that the model can learn from the data effectively. This has been done by scaling the pixel values to the range [0, 1].
- Convert the images to a suitable format: The images should be converted to a suitable format for training a deep learning model. In this project the images have been converted to grayscale.
- Handle missing or corrupted data: It is important to handle missing or corrupted data before training a deep learning model. This involve removing incomplete or corrupt images from the dataset, this has been done by deleting the rows with missing values.

3. Training a deep learning model involves adjusting the weights and biases of the model to minimize the loss function:

- Choose a suitable model architecture: There are several different types of deep learning models that can be used for ALPR. A model architecture (Convolutional neural networks (CNNs) is selected in this project due to its suitability to the data (car images) and the task (detect license plate).
- Choose a suitable loss function: The loss function measures how well the model can predict the correct output given the input data. A loss function that is the CTC (Connectionist Temporal Classification) loss is employed. This loss function is designed for tasks that involve sequential data, such as recognizing characters on a license plate. The CTC loss function compares the predicted sequence of characters with the ground truth sequence and measures the difference between the two using a cost function.
- Choose a suitable optimizer: The optimizer is responsible for updating the model's parameters during training to minimize the loss function. Adam optimizer is chosen.
- Train the model: To train the model, need to feed it the training data and labels and adjust the model's parameters to minimize the loss function. This is typically done using mini-batch gradient descent, where the model is trained on small batches of data at a time.
- Monitor the training process: It is important to monitor the training process to ensure that the model is learning effectively and to identify any issues that may arise. The training process is monitored using metrics such as the training loss, the validation loss, and the accuracy on the validation set.
- Save the trained model: Once the model is trained and saved using a tool as TensorFlow's Saved Model format.

4. Fine-tuning a deep learning model involves adjusting the hyperparameters of the model to improve its performance. In the context of an automatic license plate recognition (ALPR) system, fine-tuning the model may involve adjusting the number of layers, the learning rate, the batch size, or the type of optimizer used to update the model's parameters during training. There are several ways to fine-tune a deep learning model, including:
    - Random search: In random search, a range of values for each hyperparameter is specified, and the model is trained and evaluated for a randomly chosen to set of hyperparameter values.
    - It is important to fine-tune the model on a separate validation set, rather than the training set, to ensure that the model can generalize to new data. Once the optimal combination of hyperparameters is found, then model can be retrained on the entire training set using those hyperparameters.
    - Testing a deep learning model is an important step in evaluating its performance and determining its suitability for deployment. Testing the learning model contains the following steps: Choose a suitable evaluation metric: There are several different metrics that can be used to evaluate the performance of an ALPR system, such as accuracy, precision, recall.
    - Prepare the test data: To evaluate the performance of the model, need to have a separate test dataset that was not used for training or validation. This test dataset should be representative of the data that the model will encounter in real-world use.
    - Run the test: To test the model, feed the test data to the model and evaluate its performance using the chosen evaluation metric.

- Analyse the results: After running the test, analyse the results to determine whether the model is suitable for deployment. If the model performs poorly on the test set, go back, and fine-tune the model or consider using a different model architecture.

5. Deploy the model: The model can deploy for use in a practical application after testing and analysing the effectiveness of an ALPR system based on deep learning. Model deployment steps are as follows:

- Choose a suitable platform that meets the computational and storage needs of the model, a local device like a laptop was chosen as platform in this project.
- Package the trained model and the dependencies that needed to run the web based ALPR system.
- The deployment has been tested in this project by running a few test queries and verifying the results.
- The model has been monitored in production by collecting metrics such as accuracy, latency, and availability.

3.2 Prototype methodology

The software process model chosen for use in this project development life cycle is system prototyping, one of the fast application development methods (SDLC). Instead of creating a whole new system, this software process model was created to assess design options and collect requirements. Prototypes are used to reduce system risk by ensuring that the major concerns are identified prior to the real system being constructed. Many of the user-suggested features may be vaguely specified. Table 2 lists the activities for prototyping.

Table 2: Project activities

Phase	Activities	Deliverables
Planning Phase	Identifying the problem, scope, and objectives. And then find out how to build the web-based system and its features based on the data collected	The proposal and the Gantt chart
Analysis Phase	Gather user requirements define functional and non-functional requirements, identify technical requirements conduct analysis feasibility	UML diagram (Use case, activity diagram, sequence diagram) Class diagram, Flowchart
Design Phase	Create system architecture, database schema and data dictionaries and system user interface	System architecture Database schema and data dictionaries The interface of the system.
Implementation Phase	Develop the system to include all the functions and features.	Code program and environment setting. Test cases
Documentation phase	Discussion on what documentation should be created for future reference and maintenance.	Final report

A functional requirement is a list of services that the programmed must provide. Table 3 indicates the functional requirements for this system.

Table 3: Functional Requirements

Modules	Function	User
Register and Login Module	• The system should allow user to create an account. • The system should display an error message when empty field found. • The system should allow the user to login into the system using user_id and password that has been registered. • The system should verify user role to (Admin – User) after successfully login. • The system should redirect to desired dashboard based on user role once the user successful login. • The system should show error when user_id and password is wrong. • The system should redirect user to reset password page when clicking on forget password.	Administrator & User

Admin Dashboard	• The system should allow admin to activate license plate detector. • The system should display the result of detected license plate on image. • The system should allow admin to modify user accounts. • The system should allow admin to view the data stored in database. • The system should allow the admin to delete and modify vehicle records. • The system should allow the admin to search by license plate and view the filtered result. • The system should allow the admin to specify available space parking and display the space vacant and occupied space. • The system should send LP detection result notifications to the admin Dashboard. • The system should allow the user to logout and redirect to home page.	Administrator
User Dashboard	• The system should redirect to dashboard once the user successful login. • The system should allow the user to edit profile information. • The system should allow users to register their vehicle. • The system should allow the user to view the available space parking and display the space vacant and occupied space. • The system should be able delete registered vehicle. • The system should be able to display the user profile information. • The system should display the Vehicles information registered. • The system should allow the user to logout and redirect to home page.	Resident and visitor to UTHM campus.

Use case specifications are an important part of the development process, as they help to ensure that the system meets the needs of its users and behaves as expected in various scenarios. Figure 5 shows the use case diagram for the developed system.

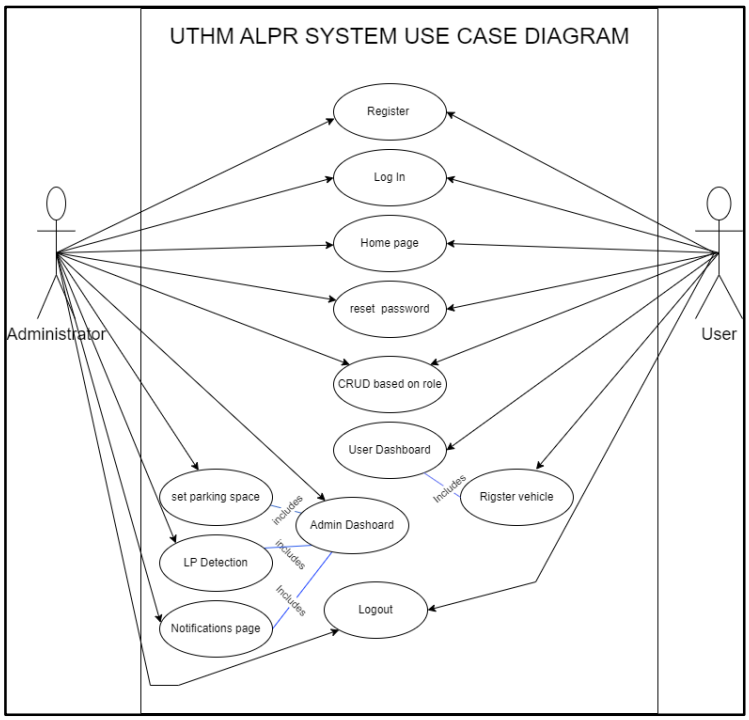

Figure 5: Use Case Diagram

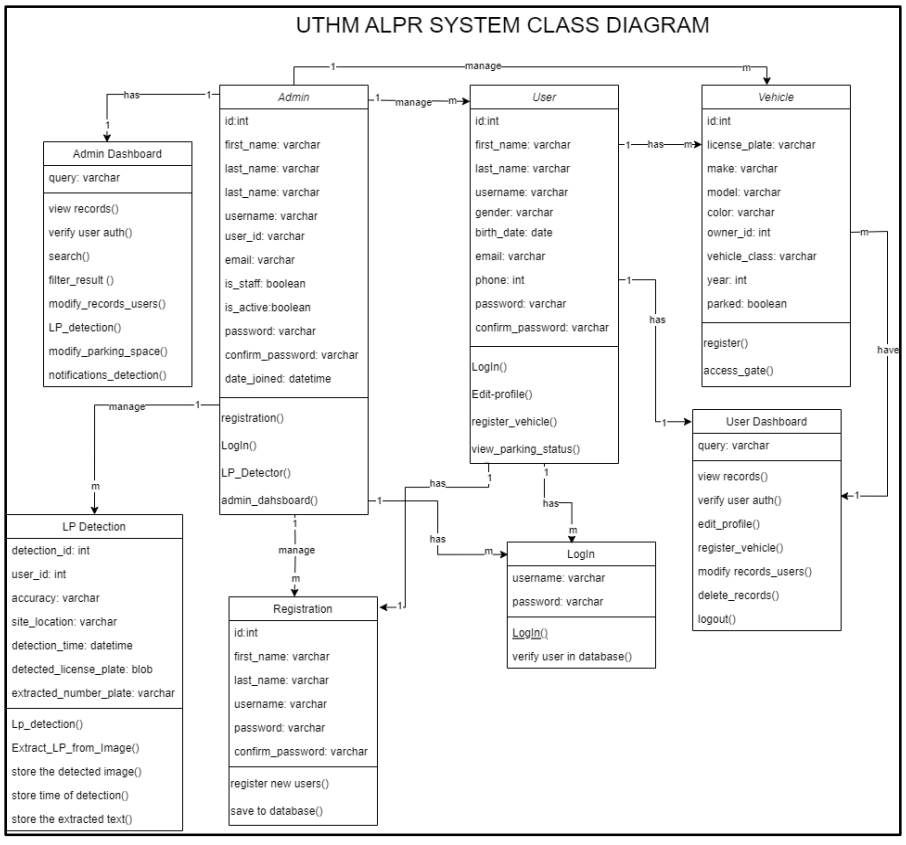

Figure 6: ALPR System Class Diagram

The class diagram shown in Figure 6 represents the static view of an application. Furthermore, it`s used in object-oriented software development to model the design of a system before it is implemented.

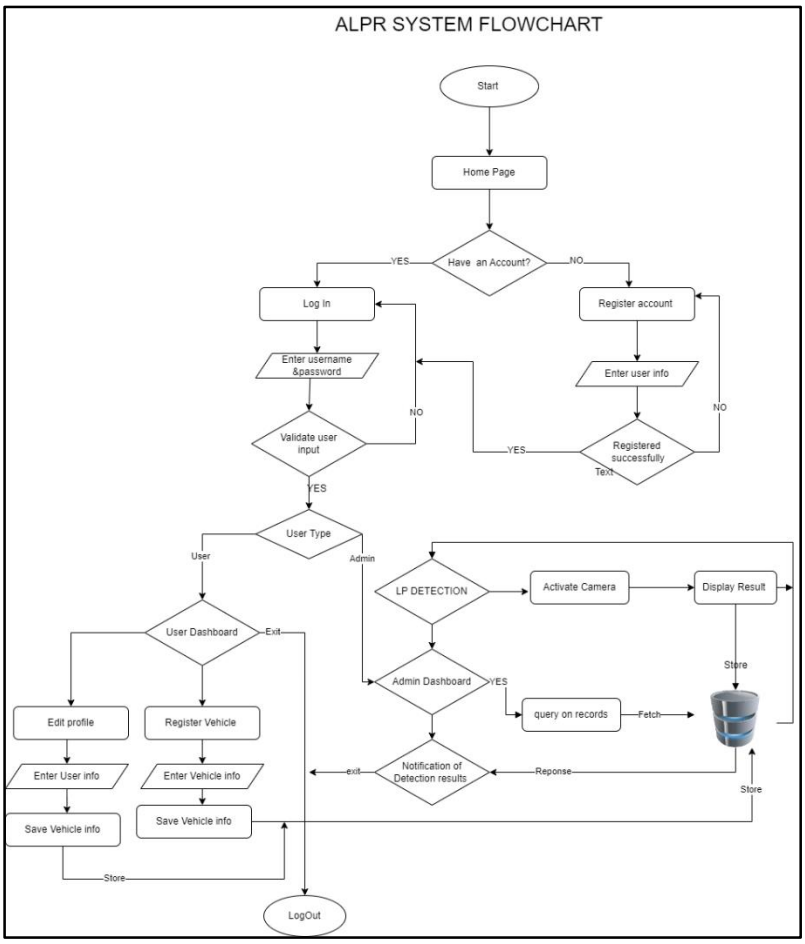

Figure 7: System flowchart

The architecture of the system involves a combination of frontend, backend, and database components, which work together to deliver the system functionality. The frontend communicates with the backend through APIs (application programming interfaces). The front end refers to the user interface of the system and the backend refers to the server-side components of the system, which handle tasks such as data storage and processing by host PostgreSQL server. Figure 8 shows the architecture.

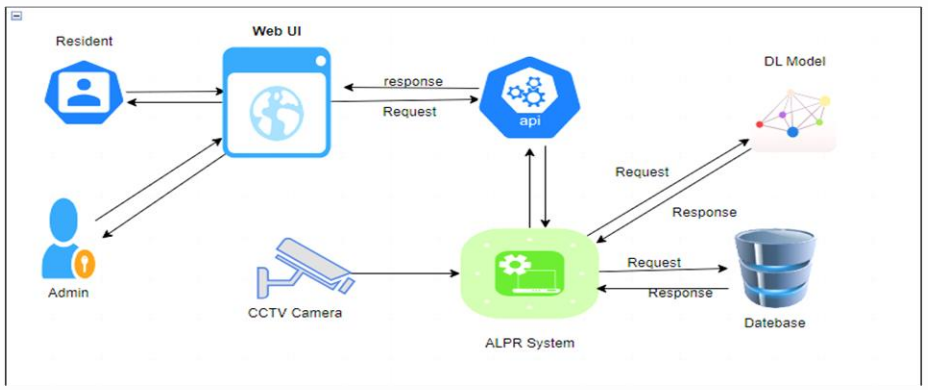

Figure 8: UTHM ALPR System Architecture

The relationship schema is as follows:

I. *Users (id, first_name, last_name, username, birth_date, gender, password, confirm_password, is_superuser, phone).*
II. *Detection result (id, user_id, accuracy, site_loaction, detection-_time, detected_plicense_plate, extracted_number_plate).*
III. *Vehicles (id, number_plate, make, model, year, color, owner, vehicle class, park status)*

Figure 8 – 13 demonstrates the design of the system user interface.

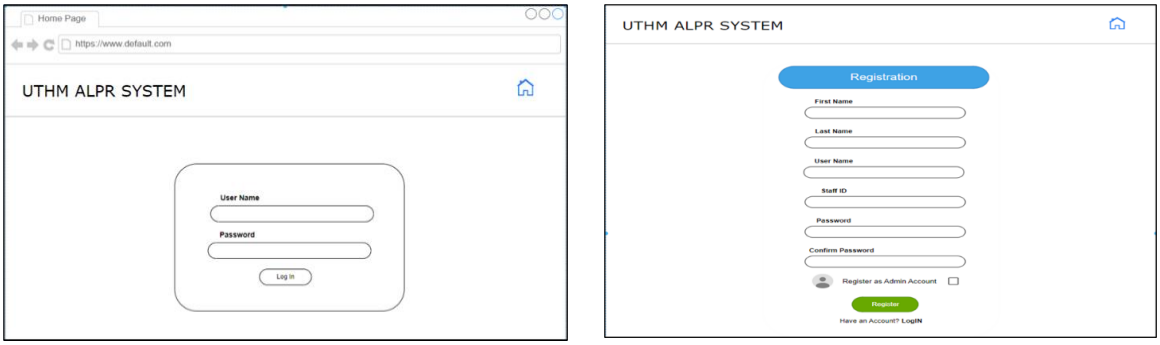

Figure 8: Login and Registration interface

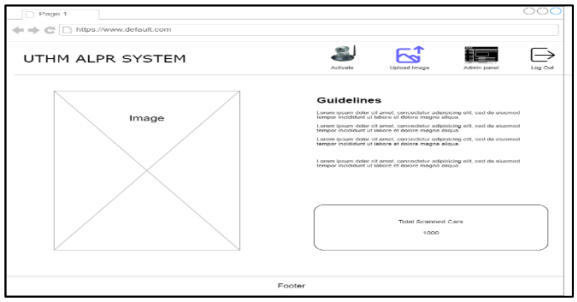

Figure 10: Dashboard interface

Figure 11: Sample result

4. Results and Discussion

This section provides a discussion on the two important phases involved in the system development process, namely the implementation phase and the testing phase.

4.1 Module Development.

The implementation phase includes the use of Visual Studio Code, anaconda, and Google Colab using the Python programming language through the Django Framework, web programming language, Google Colab environment for training deep learning models, PostgreSQL service for hosting and database.

a. User Registration and User Registration interface

Figure 12 and Figure 13 show the server-side coding and the user interface of the account registration. Python programming language is used to code the account registration process. The code snippet shows a Django custom form made for user registration. The primary objective is to increase the functionality of the UserCreationForm that Django's authentication system already includes.

This customized form, called UserRegistrationForm, includes additional fields that are necessary for user registration, such as first and last names, email addresses, and a Boolean field (is_staff) that indicates whether a user has staff privileges. These fields on the form allow users to enter the necessary registration information. In the form, the Meta class is crucial in providing the form's associated metadata. In this instance, the User model is the model that is linked to the form.

Figure 12: Registration code in Django python

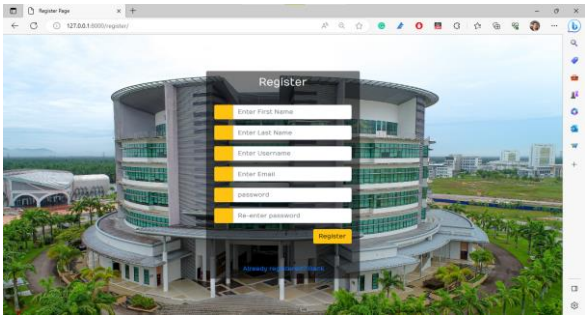

Figure 13: Account Registration

Figure 13 shows the user interface of the account registration page. Generally, the user interface consists of an input box which takes the username first name, user last name, username, email, and password input. The register button will execute the code, if user already have account needs to click on Already registered and will be redirected to login page or if clicked on Back will be redirected to the main page that display option to login or sign up.

b. Login Module

Figure 14 shows the code of a Python function that handles user login in a Django web application. It checks if the request method is POST and retrieves the username and password from the request. Then, it attempts to authenticate the user using the provided credentials. If authentication is successful, the user is logged in and redirected to the dashboard. Otherwise, an error message is displayed. If the request method is not POST, the login page is rendered.

Figure 14: Login python code snippet

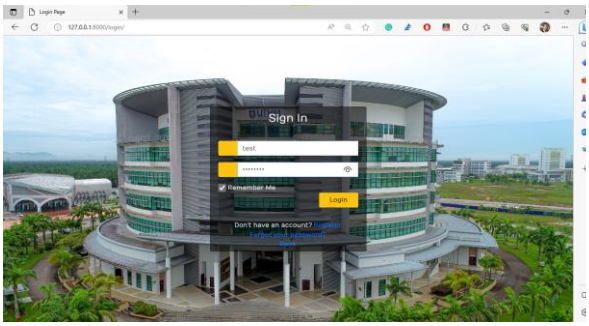

Figure 15: Login page Interface

Figure 15 shows the user interface of login page. Generally, the user interface consists of input box which takes the username and password and check box (Remember me) so that system remember the username and password then next time user open the system will directly log in without a need to input username and password. Login button will execute the code, if user does not have account needs to click on Register and will be redirected to account registration page. Else if a user forgets the password need to click on Forget your password and will be redirected to reset password page. Finally, if clicked on Back will be redirected to the main page that display option to login or sign up.

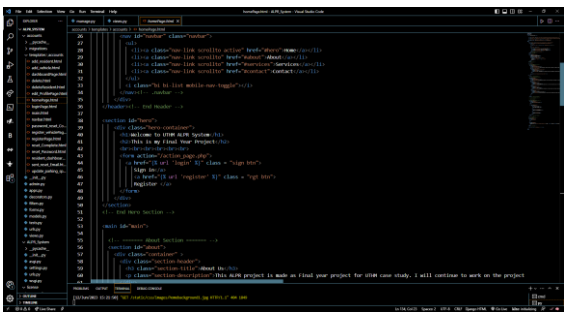

Figure 16: Home page HTML Bootstrap code

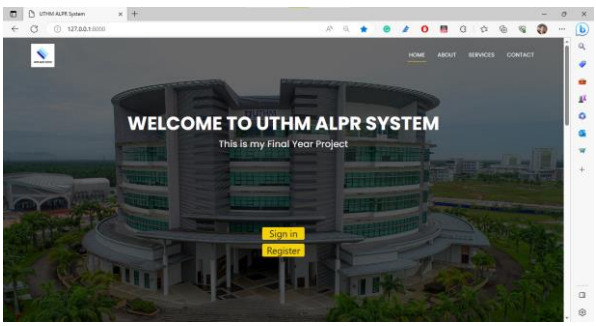

Figure 17: Home page of ALPR system

c. Home page Module

Figure 16 shows the code for the "UTHM ALPR System" web application. It consists of sections for displaying project information, services offered, and contact details. The template uses Bootstrap for styling and includes a fixed header, hero section, about section, services section, contact section, and footer. Figure 17 shows the user interface of ALPR system home page. Generally, the user interface consists of a navigation bar. And the button for Sign in will redirect the user to Login page and register button that redirect user to account registration page.

Figure 18: Django for User Dashboard

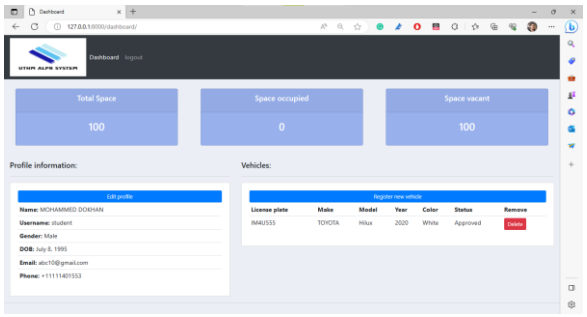

Figure 19: User Dashboard

d. Dashboard of Normal User

Django is made to construct a user interface for a web page that displays data about parking spaces and resident profiles based on Figure 18. There are two primary sections to the page. Presenting information regarding parking space is the primary goal of the first segment. It has three cards that list the number of parking spaces overall, the number of spaces that are filled, and the number of spaces that are vacant. The information is presented on these visually appealing cards in a straightforward and succinct manner.

The resident personal details and any registered automobiles are displayed in the second section. There is an option to change the profile, and the subheading says "Profile Information" at the top. The resident's name, username, gender, date of birth, email address, and phone number are all included in the card-like style of the profile details. Following the profile details, a subheading with the word "Vehicles" appears. Residents may register new automobiles in this part, which also offers a table listing each vehicle's information. The table has columns for each vehicle's identification number, make, model, year, color, and status. Additionally, an option to delete the matching car is provided for each row in the database. Figure 19 illustrates the interface of this Django User Dashboard.

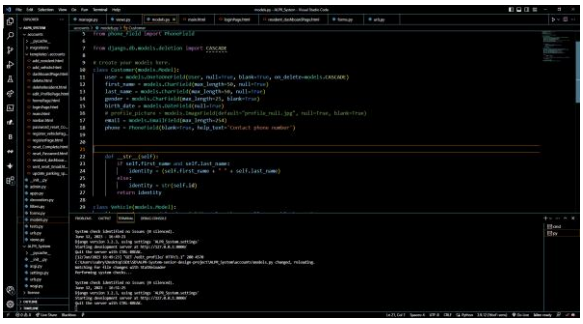

Figure 20: Edit user profile code.

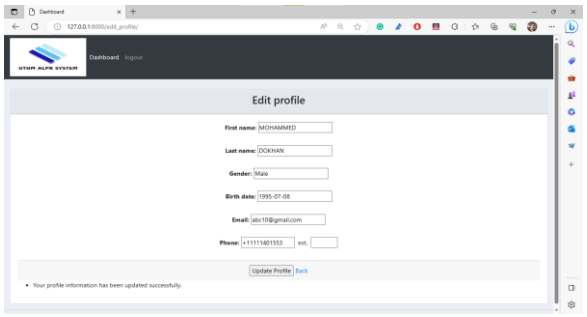

Figure 21: Edit user profile interface

According to Figure 20, the Django class's Python code depicts a system user and has fields for the user's first and last names, gender, birthdate, email address, and phone number. It has a one-to-one relationship with the "User" model. A string representation of the user is provided by the model, which combines the user's first and last names or, in the absence of a name, their ID.

The user profile editing interface is shown in Figure 21. To do this, enter the user's first and last names, gender, date of birth, email address, and phone number. When the user clicks the update profile button, the code is then run, and the system notifies the user that the update was successful. Finally, the user dashboard will be displayed if they click the back button.

Based on Figure 22 the python code for the "Vehicle" model class in Django represents a vehicle and includes attributes such as license plate, owner, make, model, year, color, vehicle class, parked status, and approval status. It enables the storage and retrieval of vehicle information in the application's database.

Based on Figure 23 that shows the interface of registration of a new vehicle and it contents of input box for license plate, make of car, model, year, color and vehicle class then Register button that will execute code after that system feedback notification that's state the vehicle registration has been submitted successfully for approval by administrator. Finally, if user click on back will be redirected to the user dashboard.

Figure 22: Vehicle registration code

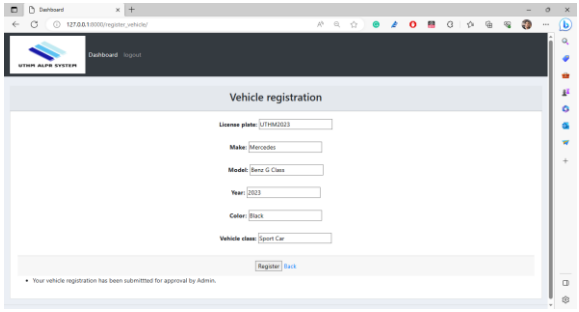

Figure 23: Vehicle registration interface

e. Administrator Dashboard

Based on Figure 24 shows a Django template designed to create a user interface for a web page that displays information related to parking space and resident profiles. The template is divided into two main sections. The first section consists of three cards that provide information about the total parking space, space occupied, and space vacant. Each card is visually styled and contains relevant data, such as the number of parking slots or the occupancy status.

Figure 24: Administrator Dashboard code

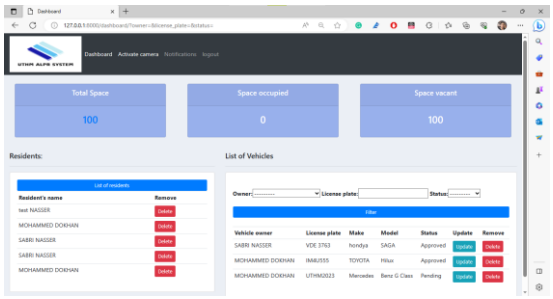

Figure 25: Administrator Dashboard interface

The second section focuses on residents and their vehicles. It includes a subheading for "Residents" and provides a button to access a list of residents. A table is displayed with columns for the resident's name and an option to delete them. Additionally, there is a subheading for "List of Vehicles" followed by a search filter form. A table displays details about each vehicle, including the owner's name, license plate, make, model, status, and options to update or delete the vehicle.

Based on Figure 25, the user interface for the Administrator user shows a navigation bar with the options dashboard, activate camera, notification, and logout. By selecting any of these options, the user is taken to the appropriate module, where they can view information about the parking status and availability. The Administrator can also set the maximum number of parking spaces. The inhabitants table displays the entire name of the inhabitants and offers an option to delete records from the table.

Finally, the vehicles list table has a filter function that can filter records based on car owner name or by status of vehicle Approved or Pending. It also has a search function by license plate. Vehicle table has two buttons to update the record and delete the record of vehicles. Figure 26 shows the code of

Django that extends the 'accounts/main.html' template. It displays a form to update the total parking spaces. The form is enclosed in a card and includes a CSRF token. Each form field is rendered with its label and corresponding input field. There is a submit button at the bottom of the form.

The interface in Figure 27 consists of a web page with a form to update the total parking spaces. The form is displayed within a card and includes a header, form fields with labels, and a submit button. The design is clean and organized, making it easy for users to update the desired information.

Figure 26: Update Parking Space Code

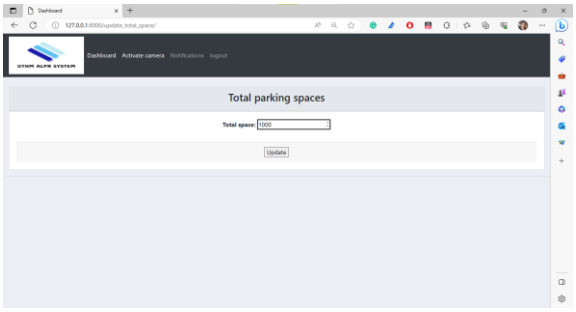

Figure 27: Update Parking Space Interface

Based on Figure 28, the provided code is a Django web application with two views: `detectFn` and `notificationsFn`. The function `detectFn`: This view captures video frames from a camera feed and applies license plate detection and optical character recognition (OCR) using OpenCV. It saves the scanned plate image, adds a timestamp, and performs OCR to extract the license plate information. Figure 29 shows the page for activating license plate detection function.

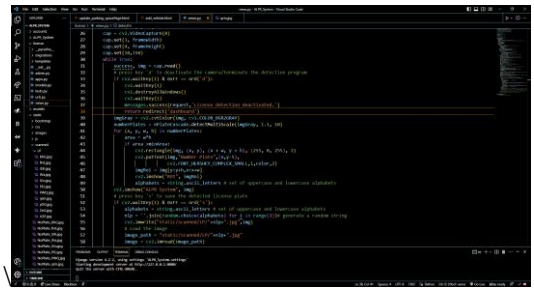

Figure 28: Activate LP Detection Code.

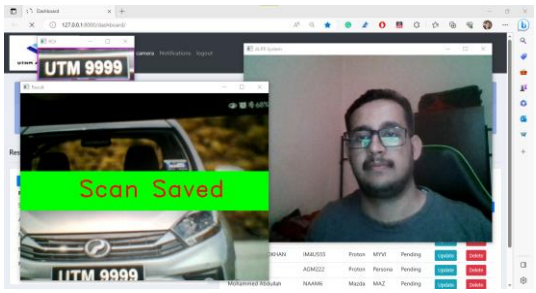

Figure 29: Activate LP Detection Interface.

It updates the parking status of the corresponding vehicle in the database or displays an access denied message if the vehicle is not registered. The function `notificationsFn`: This view renders a template to display notifications. Both views require authentication, and they can be associated with specific URLs in the application. Figure 29 depicts the interface for activating the license plate activation function. Based on Figure 31 the system will display this notification if the LP detect from camera activated. And the back button to redirect to the system dashboard.

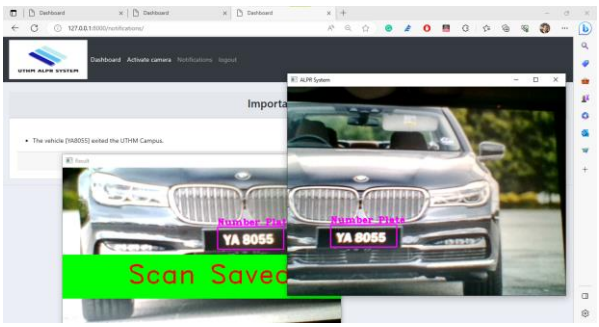

Figure 30: Activate LP Detection Interface

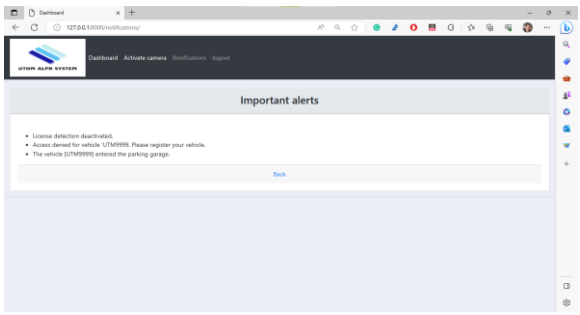

Figure 31: System Notification of LP Detection Interface

4.2 Testing

In this section User Acceptance Test (UAT) is utilized to perform testing.

a. Account Registration and Login

Table 4 shows the test case for Account Registration and Login module. The purpose of this test is to verify whether the administrator is allowed to register for an account, login into the system, and whether the system will restrict login if an incorrect credential is entered.

Table 4: Test Case for Account Registration and Login Module

Module: Account Registration and Login				
Test Case ID	Description	Expected Result	Actual	Result
M1-1	To check whether user can register for an account	The user should be able to create for an account	The user has successfully created for an account	Pass
M1-2	To check whether system will validate user input	The user enters the input in registration form and get system feedback for the input field.	The user successfully get the system feedback on input validation.	Pass
M1-3	To check whether a user can login into the system	The user should be able to login into the system	The user has successfully logged into the system	Pass
M1-4	To check whether the system will restrict login whenever a wrong credential is entered	The system should restrict login when an incorrect credentials has been entered	The system restricted the login when an incorrect or no credentials has been entered	Pass
M1-5	To check whether system will allow redirect the user to forget password page	The system should redirect user to forget password page.	The system successfully redirect user into forget password page	Pass
M1-6	To check whether system will save user login information and remember it on next login	The system should save the user's login information and automatically fill in the login fields	The system successfully login user automatically.	Pass

b. User Dashboard module

Table 5 summarizes test cases for the User Dashboard module. The test cases cover accessing the dashboard, editing user profiles, registering new vehicles, and logging out. Each test case has passed successfully.

Table 5: User Dashboard Model

Module: User Dashboard				
Test Case ID	Description	Expected Result	Actual	Result
M2-1	To check whether the user can access the dashboard after successful login	The user should be able to access the dashboard	User successfully accesses the dashboard	Pass
M2-2	To check whether the user can edit their user profile on the dashboard	The user should be able to edit their user profile	User can successfully edit their profile	Pass
M2-3	To check whether the user can register a new vehicle on the dashboard	The user should be able to register a new vehicle	User can successfully register a new vehicle	Pass
M2-4	To check whether the user can log out from the dashboard	The user should be able to log out from the dashboard	User can successfully log out from dashboard	Pass

c. Administrator Dashboard module

Table 6 provides test cases and their results for the Administrator Dashboard module. It verifies various functionalities and their expected outcomes. Each test case is assigned a unique ID and described briefly. The table covers accessing the dashboard, displaying parking slot information, managing residents and vehicles, activating the camera, license plate detection, notification sending, and logging out. It serves as a concise summary of the module's functionality and test outcomes.

Table 6: Administrator Dashboard Module

Module: Administrator Dashboard				
Test Case ID	Description	Expected Result	Actual	Result
M3-1	To check whether the administrator can access the dashboard after successful login	The administrator should be able to access the dashboard	Administrator successfully accesses the dashboard	Pass

Module: Administrator Dashboard				
M3-2	Verify the display of total parking slots in the "Total Space" card	The "Total Space" card should display the correct total number of parking slots	Successfully displayed the total number of parking slots	Pass
M3-3	Verify the display of occupied parking slots in the "Space Occupied" card	The "Space Occupied" card should display the correct number of occupied parking slots	Successfully displayed the number of occupied parking slots	Pass
M3-4	Verify the display of vacant parking slots in the "Space Vacant" card	The "Space Vacant" card should display the correct number of vacant parking slots	Successfully displayed the number of vacant parking slots	Pass
M3-5	Verify the functionality of the "List of residents" button	Clicking the "List of residents" button should redirect to the "dashboard" URL	Successfully redirected to the "dashboard" URL	Pass
M3-6	Verify the display of residents in the table	The table should display the list of residents correctly	Successfully displayed the list of residents	Pass
M3-7	Verify the functionality of the "Delete" button for a resident	Clicking the "Delete" button for a resident should redirect to the "delete_resident" URL with the corresponding resident ID	Successfully redirected to the "delete_resident" URL with the correct resident ID	Pass
M3-8	Verify the display of vehicles in the table	The table should display the list of vehicles correctly	Successfully displayed the list of vehicles	Pass
M3-9	Verify the functionality of the search form for vehicles	Entering search by the licence plate and clicking the "Filter" button should filter the vehicles accordingly.	Successfully filtered the vehicles based on the search criteria	Pass

M3-10	Verify the functionality of the "Update" button for a vehicle	Clicking the "Update" button for a vehicle should redirect to the "update_vehicle" URL with the corresponding vehicle ID	Successfully redirected to the "update_vehicle" URL with the correct vehicle ID	Pass
M3-11	Verify the functionality of the "Delete" button for a vehicle	Clicking the "Delete" button for a vehicle should redirect to the "delete_vehicle" URL with the corresponding vehicle ID	Successfully redirected to the "delete_vehicle" URL with the correct vehicle ID	Pass
M3-12	Verify functionality of activate camera	Clicking on activate camera the LP detection should be activated and access camera of the system	Successfully the system activates the LP detection.	Pass
M3-13	Verify the LP detection perform detecting License plate successfully	The LP detection model should detect licence plate.	Successfully LP detection model detect license plate	Pass
M3-14	Verify the system send notification of LP detection result	The system should send notification of the LP detection result to the administrator notifications page that can be viewed by administrator	Successfully system sent the notification to the notification page in the administrator side	Pass
M3-15	To check whether the administrator can log out from the dashboard	The administrator should be able to log out from the dashboard	Administrator can successfully log out from dashboard	Pass

5. Conclusion

In conclusion, the UTHM ALPR system has been found to be an effective and efficient tool for automating the process of license plate recognition. The system was able to accurately recognize and extract license plate numbers from a variety of different images, even in cases where the plates were

partially occluded or had low contrast. The system was also able to perform well in a real-world setting, with a high rate of success in correctly identifying license plates in a large dataset of images.

Overall, the results of this study demonstrate the potential of the UTHM ALPR system as a useful tool for a wide range of applications, including traffic monitoring, security, and law enforcement. The system's high accuracy and efficiency make it an attractive option for organizations seeking to automate license plate recognition tasks. Based on these findings, it is recommended that the UTHM ALPR system be further developed and implemented in a wider range of applications.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] Bhaskar, P. K., & Yong, S. P. (2014, June). Image processing-based vehicle detection and tracking method. In *2014 International Conference on Computer and Information Sciences (ICCOINS)* (pp. 1-5). IEEE.

[2] Rafique, M. A., Pedrycz, W., & Jeon, M. (2018). Vehicle license plate detection using region-based convolutional neural networks. Soft Computing, 22(19), 6429-6440.

[3] Puranic, A., Deepak, K. T., & Umadevi, V. (2016). Vehicle number plate recognition system: a literature review and implementation using template matching. *International Journal of Computer Applications*, *134*(1), 12-16.

[4] [4] Selmi, Z., Halima, M. B., & Alimi, A. M. (2017, November). Deep learning system for automatic license plate detection and recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)* (Vol. 1, pp. 1132-1138). IEEE.

[5] Sun, Z., Bebis, G., & Miller, R. (2006). On-road vehicle detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, *28*(5), 694-711.

[6] Laroca, R., & Menotti, D. (2020, November). Automatic license plate recognition: an efficient and layout-independent system based on the YOLO detector. In Anais Estendidos do XXXIII Conference on Graphics, Patterns and Images (pp. 15-21). SBC.

[7] Chan, L. Y., Zimmer, A., da Silva, J. L., & Brandmeier, T. (2020, September). European union dataset and annotation tool for real time automatic license plate detection and blurring. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-6). IEEE.

[8] Sung, J. Y., & Yu, S. B. (2020, November). Real-time automatic license plate recognition system using YOLOv4. In 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia) (pp. 1-3). IEEE.

[9] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). IEEE.

[10] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

[11] Automatic Licence Plate Recognition (ALPR) algorithm (2022) SentiVeillance. Available at: https://www.sentiveillance.com/automatic-license-plate-recognition-algorithm/ (Accessed: January 9, 2023).

[12] Security & Automation Solutions - OpenALPR by Rekor. Available at: https://www.openalpr.com/solutions/security-and-automation (Accessed: January 9, 2023).

[13] Software, D.T.K. (no date) License plate recognition solulution, LPR solution, ALPR solution, ANPR solution, DTK Software. Available at: https://www.dtksoft.com/lprsolution (Accessed: January 9, 2023).