Department of Computer Engineering and Informatics

Middlesex University

Parking Payment Monitoring System using Deep Learning

Student name: Lowell Aaron Lobo

Student ID: M00795791

Supervisor: Mr Judhi Prasetyo

PDE3112 Major project

BEng Computer Systems Engineering

May 2023

Acknowledgements

First and foremost, I would like to thank God, who gave me the strength, inspiration and courage to brace myself, move onward and complete this project. I am incredibly grateful and would like to express my deepest thanks for the successful completion of this project.

Next, I would like to express my utmost gratitude to my project supervisor, Mr Judhi Prasetyo, for his constant support, timely advice and gentle disposition with me during the duration of this Major Project. His encouragement and guidance were imperative to the successful completion of the project.

I am also grateful to my friends and colleagues who bore with me and supported me during the project. Their valuable feedback and suggestions helped in improving the quality of the project. I especially want to thank Kuruvilla and Saad, who pushed me to strive for the best project possible.

Finally, I would like to thank my family for their constant encouragement, love and motivation, which helped me stay focused on the goal and kept me motivated throughout the project's timeframe.

Once again, I would like to convey my sincere gratitude to all those who contributed to the building and completion of this project.

Table of Contents

Abstract

This project is an actual-life implementation of License Plate Detection concepts where deep learning is used to verify parking fee payment and levy a fine if necessary. The system uses a Raspberry Pi to capture images of parked cars with license plates visible (ML Activation), and these images are sent to a hosted server using an API call. The Flask server then receives the images, accurate license plate detection is performed, license plate number is extracted, and parking payment is verified. All inferred data is then stored in a secure database solution called Supabase. Authorised personnel can view the details of all captured license plates by utilising a hosted Next.js web application. Access to the Next.js application is restricted, and users must authenticate themselves.

Three deep learning models, MobileNet SSDv2, YOLOv7 and YOLOv7 Tiny, and trained for license plate detection, and the models are compared with each other based on the set criteria. Different methods for inference are thus built, resulting in a total of twelve inference methods which are compared with each other.

Three different methods of pre-processing are implemented and compared, along with four different OCR libraries for image-to-text conversion, to extract the license plate number from the image. The libraries utilised are, Tesseract, Keras-OCR, EasyOCR and PaddleOCR and each library is used to inference with each pre-processing method.

It is found that the YOLOv7 is the best model for detection, assuming sufficient resources have been provided; else, YOLOv7 Tiny is the better model. Thus, YOLOv7 Tiny was implemented for ML Activation on the Raspberry Pi, and YOLOv7 was used in the Flask Backend. PaddleOCR is the better OCR library among the rest after testing; thus, PaddleOCR is used for image-to-text conversion. Systematic Review shows that YOLOv7 and PaddleOCR are both methods that have not been widely used in application but show prominent results.

Finally, the images stored on Supabase can be used to collect training images to improve the built deep learning models further.

Introduction

With the recent developments in robotics and machine learning, the world is moving towards automation. The benefits of automation are being constantly researched, and most researchers even strive to implement automation wherever possible. Automation can help reduce human interference in many fields, especially in scenarios where the tasks remain repetitive and monotonous yet are extremely important or costly. The appeal of automation is purely because of its ability to increase efficiency, safety, turnover, and accuracy. One field that would benefit from implementing automation is license plate detection. There are many use cases for this technology. License plate detection is heavily used for parking management and traffic monitoring. Governments also use license plate detection for toll collection and border control and to verify vehicle access to restricted facilities. Each institution uses its own custom-built license plate detection model which is tuned for specific scenarios. Yet, although license plate detection has been researched a lot, there is yet to be a perfect, global solution.

This project aims to build a system that can help reduce or completely negate human interference in any sector and to research newer techniques for license plate detection. Specifically, license plate detection is implemented here to check whether the parking payment has been made and to levy a fine for any infraction. A device captures images of cars parked with the licenses, which are then sent for verification. License plate detection and parking fee payment will be processed during the verification. The entire information will be stored on a secure cloud-hosted database, and authorised officials can view the information collected on a build web app. Multiple prototypes were built and compared, from which the best is proposed and discussed in detail.

This report is organised as follows: Section 3 provides a systematic review of license plate detection models and methods. The requirements of the project and the resources required to complete the project are listed in Section 4. The complete project design and development are highlighted in Section 5. In-depth testing and analysis are demonstrated in Section 6. Finally, the entire project is concluded, and the limitation and future scope are listed in Section 7.

Literature Review

Title

A Systematic Review on License Plate Detection Steps, Methods and Models

Abstract

This paper compares the various License plate detection methods over the past nine years. Although there have been revolutions in the CNN models, some researchers still prefer image processing methods to detect license plates. All methods and models have been researched, and application in real-time has been discussed. Sixty papers have been analysed, and findings have been compared systematically. The comparison shows that the split between those that use machine learning and those that use standard image processing techniques is relatively even.

Introduction

Rationale

In the past few years, there has been an increase in the use of machine learning over different spectrums. One is its implementation in license plate detection for traffic analysis, intelligent transport systems and parking management, to name a few problem areas. License plate detection combines many smaller components, mainly license plate area localisation and license plate number recognition. For the first part, machine learning and image processing have been used, and object character recognition techniques have been implemented for the second part.

Machine learning techniques have replaced traditional image processing methods in license plate area localisation. Yet many researchers still prefer image processing and morphological and contour processes to detect license plate areas. Both machine learning and image processing techniques will be compared based on their efficiency, speed and ease of usage.

In license plate number recognition, all image-to-text conversion methods have been compared, especially image pre-processing methods.

The present study intends to compare 60 research papers discussing license plate detection and find what methods are used in license plate detection by various researchers and how each researcher has implemented the techniques to attain low loss metrics in their simulations. This research aims to collect all relevant information together for the ease of future researchers in license plate detection.

Objectives

The present study aims to analyse all the relevant literature, journal articles and conference papers only within the past nine years, collect all information related to license plate detection, and fill in any gaps in the existing literature. The advancements to existing technology and its implementation are analysed. Primarily, the main objective of this research is to collect all information related to license plate detection and to find the best way to perform this in a real-time situation. This research checks if newer technology and machine learning models can be implemented to increase efficiency, speed and ease of usage for license plate detection. For example, the usage of the recently released YOLOv8 model in license plate detection.

Methods

Protocol

Based on the aims of the review, a specific review protocol was built to remain as guidelines for the literature search. The review protocol contains the search terms, the databases used, the documents selected, and the screening criteria.

Eligibility Criteria

The research had to meet three predefined criteria for a paper to be considered. The first criterion is that the report should be open-access, full-length, and published in English. Since the review is technology related, the second criterion is that the paper should have been published within the past nine years. The final criteria are that the documents should explain the concept behind the license plate detection method, implement the discussed solution and display the simulation result in performance metrics.

Information Sources

An extensive literature search was performed in IEEE Xplore, Web of Science and ScienceDirect. The collection of literature was completely and thoroughly conducted on February 2023, and additions were made to the literature collection.

Search Strategy

The data was obtained from IEEE Xplore on February 16, 2023, Web of Science on February 16, 2023, and ScienceDirect from February 18 to February 22, 2023. The search terms were ‘License Plate’ and ‘Machine Learning’. The two search terms were combined with the Boolean Operator ‘AND’ and used to explore every published paper’s metadata. The issue date year was filtered to only display papers from 2014 till 2023, as of February 2023.

Study Selection Process

The screening process consisted of reading the title and abstract to see if the paper in question meets the eligibility criteria stated above. The screening was done using Rayyan ai. The screening was done using based on the titles and abstracts. Next, the full text of each paper was downloaded and scanned to see if it still met the eligibility criteria. Even those previously included documents using Rayyan ai were removed if they did not fully meet the requirements. The included and maybe papers were exported into an excel file from Rayyan ai to keep track of the full-text screening process. The articles were all screened by one person.

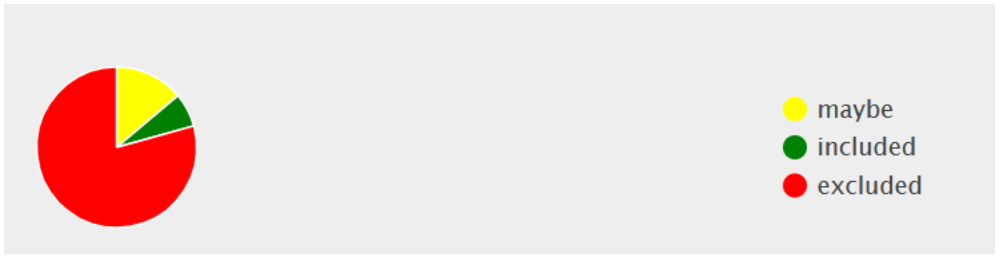


Fig. 1. Pi chart of screening on Rayyan ai (Haddaway *et al.*, 2022)

Data Collection Process

All selected papers were read through, and notes were taken based on similar themes and noted in an excel sheet. The articles were all read through by one person.

Data Items

All recurrent themes in the final screened papers were identified and recorded in an excel sheet. Each article is linked to a theme, and key findings were extracted and noted in the same excel sheet.

Results

Study Selection

This study reviewed 60 papers about License Plate Detection. The screening process has been displayed as a PRISMA flow model in Fig. 2. Initially, the literature search yielded 723 articles and reports from three different databases, from which 9 were removed since Zotero detected duplicates. Of the collected reports, 547 were Journal Articles, 117 were Conference Papers, and the last 50 were Book Chapters, as shown in Fig. 3.

Although the literature search yielded 714 reports, only 148 were screened after duplicate removal. The other 566 reports were not at all related to License Plate Detection. The massive number of false reports comes from the fact that the search terms were used to search through the entire text. License Plate Detection is widely used in Intelligent Transport Systems and Parking Management Systems, Prediction and Monitoring of Traffic Congestion, and Detection of Parking Slots, and is used as an easy way to test Scene Text Recognition. Thus, many papers mention “License Plate” and “Machine Learning” somewhere in the article, even if License Plate Detection as a concept is never used or implemented. Next, 18 more reports had to be removed since access was denied during retrieval. The full texts of the 130 reports were scanned through, and an additional 70 papers were excluded. The reason for report exclusion has been stated in the PRISMA flow diagram. The reports that do not deal with License Plate Detection, even though the abstract mentions License Plate Detection, use License Plate Detection as a means to perform another operation, for example, detecting a vehicle using License Plate Detection.

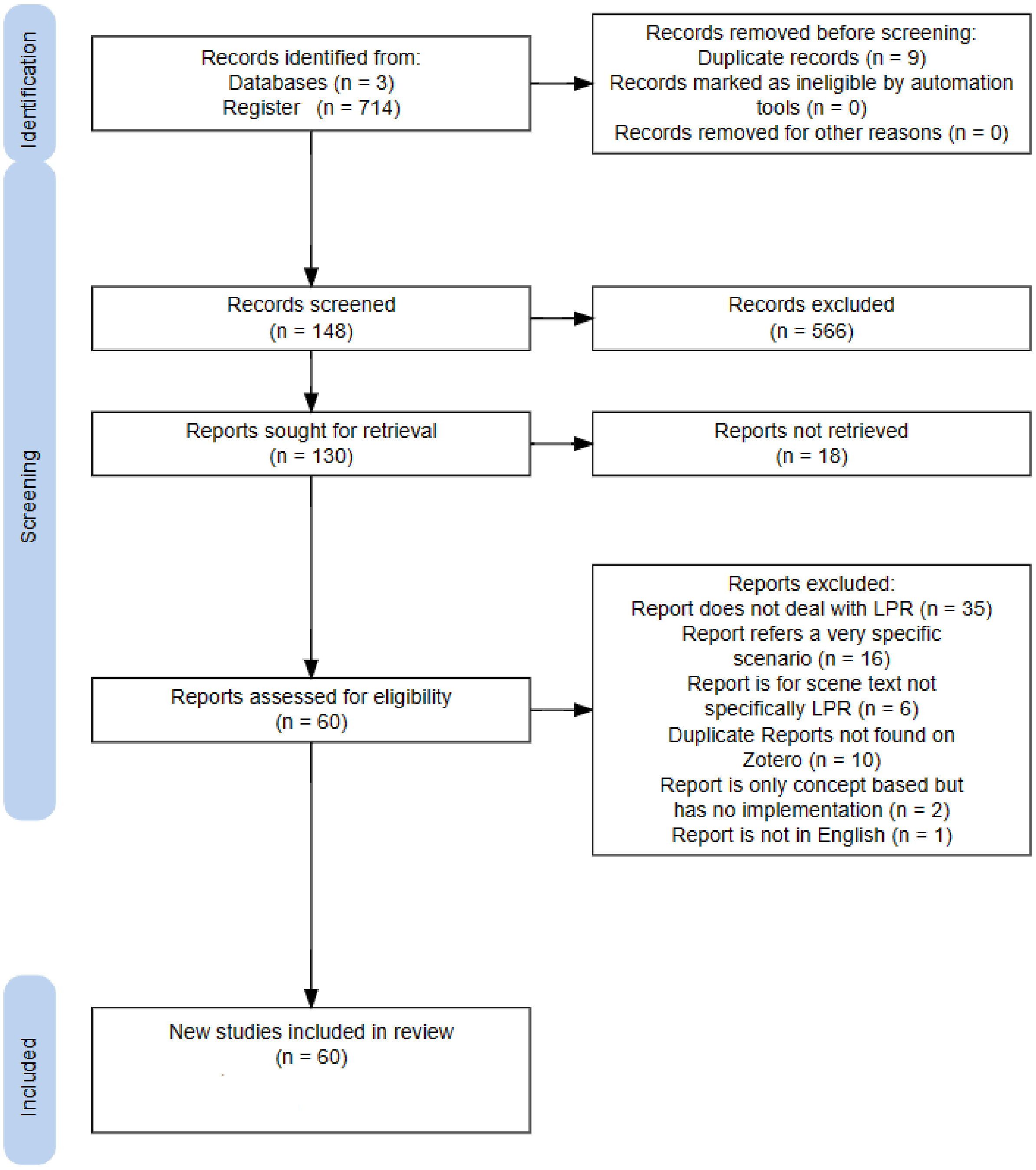
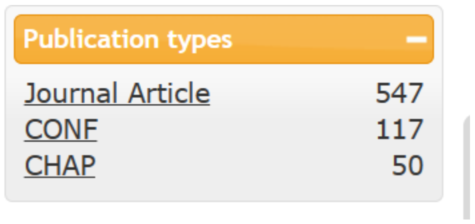

Identification
Screening
Included
Records identified from:
Databases (n = 3)
Register (n = 714)
Records removed before screening:
Duplicate records (n = 9)
Records marked as ineligible by automation tools (n = 0)
Records removed for other reasons (n = 0)
Records screened
(n = 148)
Records excluded
(n = 566)
Reports sought for retrieval
(n = 130)
Reports not retrieved
(n = 18)
Reports assessed for eligibility
(n = 60)
Reports excluded:
Report does not deal with LPR (n = 35)
Report refers a very specific scenario (n = 16)
Report is for scene text not specifically LPR (n = 6)
Duplicate Reports not found on Zotero (n = 10)
Report is only concept based but has no implementation (n = 2)
Report is not in English (n = 1)
New studies included in review
(n = 60)

Fig. 2. PRISMA flow diagram. (Haddaway *et al.*, 2022)


Publication types

Journal Article	547
CONF	117
CHAP	50

Fig. 3. Distribution of publication types of reports before screening.

Of the 60 finally screened reports, the segregation by publication year has been displayed in Fig. 4. Most reports have been taken from recent years since technology is a growing field and improvements can happen frequently.

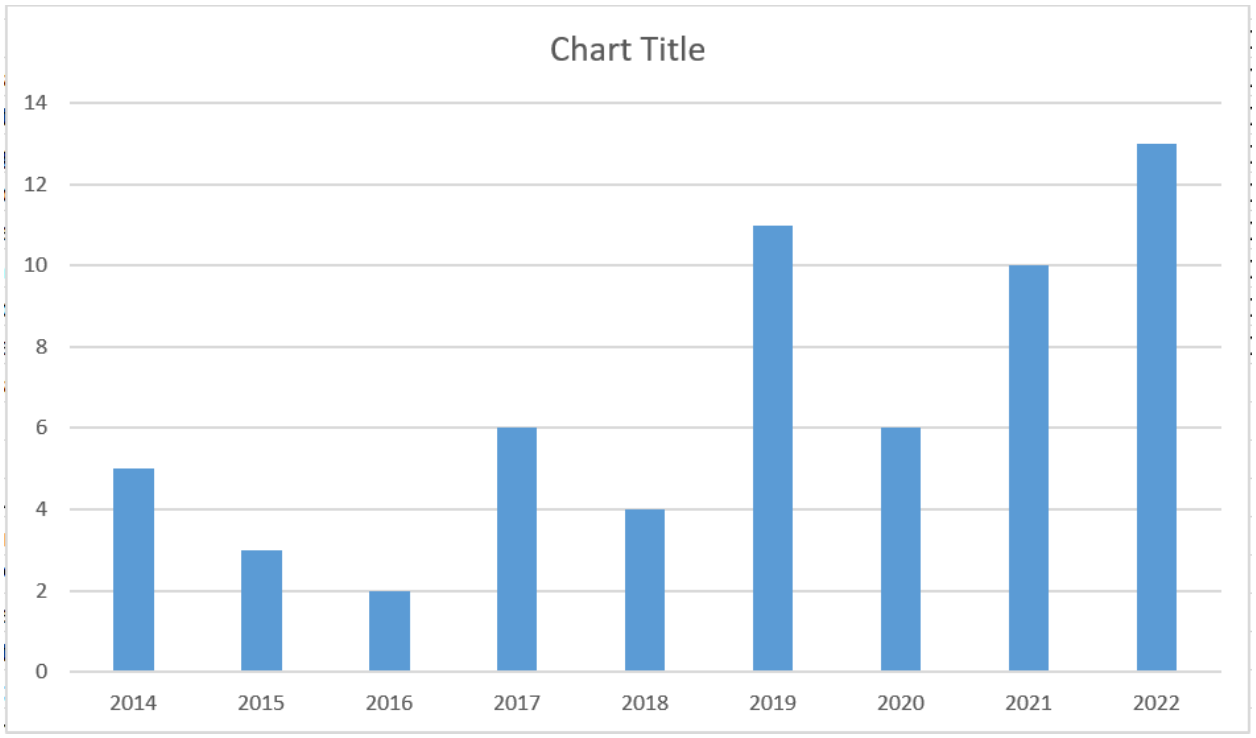


Fig. 4. Distribution of reports based on publishing year

Study Characteristics

Table. 1 lists the themes and places each of the 60 reports within each theme, if applicable.

Reports	Pre-processing	Image Processing	Machine Learning	Character Segmentation	OCR	Comparison
(Liu and Lin, 2017)					Yes	Yes
(Tabrizi and Cavus, 2016)	Yes	Yes		Yes	Yes	
(Nguyen and Nguyen, 2018)				Yes	Yes	Yes
(Polishetty, Roopaei and Rad, 2016)	Yes		Yes		Yes	
(Varma P *et al.*, 2020)	Yes	Yes			Yes	
(Awan *et al.*, 2019)		Yes				
(Kessentini *et al.*, 2019)			Yes	Yes	Yes	Yes
(Trapeznikov, Priorov and Volokhov, 2014)	Yes			Yes		
(N. Ramya, M.	Yes	Yes	Yes		Yes	Yes

Annamalai, and K. Santhosh, 2022)						
(Sunil, Samuel and C V, 2022)	Yes		Yes	Yes	Yes	
(Das and Kumari, 2021)	Yes	Yes				
(Lyasheva, Lyasheva and Shleymovich, 2021)	Yes	Yes				
(Elhadi, Abdalshakour and Babiker, 2019)			Yes	Yes	Yes	
(Radha and Viju, 2022)			Yes	Yes		
(Sasi, Sharma and Cheeran, 2017)		Yes		Yes	Yes	Yes
(El-Latief *et al.*, 2017)			Yes			

(Tang *et al.*, 2022)						
(Lubna, Mufti and Shah, 2021)						
(Bhagat and Thakur, 2021)		Yes		Yes	Yes	
(Rokonuzzaman *et al.*, 2017)			Yes	Yes	Yes	
(Hsieh *et al.*, 2022)			Yes	Yes	Yes	
(Dhar *et al.*, 2019)		Yes		Yes	Yes	Yes
(Roy *et al.*, 2015)					Yes	
(Onim *et al.*, 2022)			Yes		Yes	Yes
(Neto *et al.*, 2015)		Yes		Yes	Yes	Yes
(Wang *et al.*, 2022)				Yes	Yes	Yes
(Zhang *et al.*, 2021)						

(Selmi, Ben Halima and Alimi, 2017)	Yes	Yes	Yes	Yes	Yes	Yes
(Sarif *et al.*, 2020)			Yes	Yes	Yes	Yes
(Lien *et al.*, 2019)			Yes	Yes	Yes	
(Selmi *et al.*, 2020)			Yes	Yes	Yes	Yes
(Zhang *et al.*, 2019)		Yes	Yes	Yes	Yes	
(Aljelawy and Salman, 2022)			Yes		Yes	
(Al-Shemarry and Li, 2020)	Yes		Yes			Yes
(Al-Shemarry, Li and Abdulla, 2018)	Yes		Yes			Yes
(Al-Shemarry, Li and Abdulla, 2023)	Yes		Yes			Yes

(Kim, Lim and Kim, 2019)		Yes		Yes	Yes	
(Ying, Xin and Wanxiang, 2018)			Yes			
(Yao and Yi, 2014)			Yes			
(Yang, Yin and Chen, 2015)	Yes		Yes	Yes		
(Asif *et al.*, 2019)	Yes	Yes	Yes			Yes
(Rabindranath *et al.*, 2022)	Yes	Yes		Yes	Yes	
(Shafi *et al.*, 2022)	Yes		Yes	Yes	Yes	Yes
(L. Qin, W. Wei, and Z. Ma, 2021)			Yes		Yes	
(Al-Hmouz and Aboura, 2014)	Yes	Yes				
(Gou *et al.*, 2014)	Yes	Yes		Yes	Yes	

(Satsangi, Yadav and Sudhish, 2018)		Yes				
(Kumar, Barman and Pal, 2019)	Yes	Yes				
(Kassm and Achkar, 2017)	Yes		Yes	Yes	Yes	
(Ali *et al.*, 2021)			Yes	Yes	Yes	Yes
(Henry, Ahn and Lee, 2020)			Yes		Yes	Yes
(Min *et al.*, 2019)	Yes		Yes			Yes
(Sathya, Vasuhi and Vaidehi, 2020)			Yes		Yes	
(Ashrafee *et al.*, 2022)			Yes		Yes	Yes
(Maduri *et al.*, 2021)			Yes		Yes	
(Saranya, Kanagavalli		Yes		Yes	Yes	

and Priyaradhika devi, 2021)						
(Nguyen *et al.*, 2021)			Yes		Yes	Yes
(Chao Gou *et al.*, 2014)		Yes		Yes	Yes	
(Li *et al.*, 2019)	Yes			Yes	Yes	
(Pourhadi *et al.*, 2022)			Yes		Yes	

Table. 1. Table of themes

Synthesis of Results

Reports	Description
(Liu and Lin, 2017)	This report was purely for license plate number recognition. They first used K-means clustering to recognise the alpha-numeric values, but since K-means is not very precise, next, an SVM is used to recognise those characters that have a high chance for false recognition. The results were compared with template matching methods and all the SVM model types.
(Tabrizi and Cavus, 2016)	First, all images are converted to grayscale and then fed into the system. Image processing is used to locate the license plate. The image is dilated, and logic is applied based on general information about the license plate's width and height. Segmentation is done using image processing morphology, and then K-NN is used to recognise the characters. Since K-NN can't distinguish between similar-looking characters, an SVM is used to recognise those characters.

(Nguyen and Nguyen, 2018)	In this report, license plate locating and license plate number recognition is all done using the same custom-built CNN model called CNN-L3, which was trained using the SGD algorithm. The whole idea is to make an end-to-end model that will unify all the steps of license plate detection into one. The result was compared against three different methods for license plate locating and tesseract for recognition.
(Polishetty, Roopaei and Rad, 2016)	First, images are converted to grayscale and edge detection is performed. Next, the license plate is located using a custom 9 Layer CNN. The license plate area is cropped out, binarised, and fed into a custom 7 Layer CNN for character recognition. The paper also reported that sign boards are detected as license plate candidates using their location model but not classified as one.
(Varma P *et al.*, 2020)	The input images are converted to HSV format; then morphological operations are performed. Next, the images are binarised, and the license plate is located using opencv contours and logic. The supposed license plate area is deskewed using an opencv rotation matrix and then passed into a K-NN for character recognition. Also, since opencv may detect many contours, the contour area with the highest value in character recognition is selected as the license plate area.
(Awan *et al.*, 2019)	This report uses vehicle detection using a trained Faster R-CNN model to ensure that the license plate area is not false. Then, image processing is used to extract the license plate area by converting the image to HSV and performing morphological operations. The report mentions that the vehicle is detected in images with extreme reflective glare but not the license plate.
(Kessentini *et al.*, 2019)	Here the license plate area is located using a trained YOLOv2 model. Next, character recognition is done through two methods. The first method uses a custom CNN model that combines segmentation and recognition. The second method uses YOLOv2 to segment and later recognise the characters. The results were compared with other papers.

(Trapeznikov, Priorov and Volokhov, 2014)	This report is purely for character segmentation. The images were grayscaled and then binarised, and then sent for segmentation. Character segmentation was done using the Harris algorithm on the histogram of oriented gradient features of the image.
(N. Ramya, M. Annamalai, and K. Santhosh, 2022)	The images are first converted to grayscale and then passed into a Haar cascade classifier to detect if the image contains a vehicle. Next, morphology is used to detect the license plate. The detected license plate area is cropped out and sent to EasyOCR for character recognition. The report compares tesseract with EasyOCR for character recognition. The method proposed is for videos but can’t be used in real-time since the video is converted into images and saved in a folder, and then the inference is run.
(Sunil, Samuel and C V, 2022)	The images are first converted to HSV format, and then CLAHE, erosion and dilation operations are performed on the image. Next, YOLOv5 is used to locate the license plate, and another YOLOv5 model is used for character segmentation. Character recognition was done using two methods, LSTM and scene text recognition. Roboflow was used for annotating the images, and the report states that character recognition using tesseract did not work. The report also mentions that minimum requirements of a medium-quality camera and proper lighting are required.
(Das and Kumari, 2021)	The images are grayscaled, and then canny edge detection and extended Hough transform are applied to extract the license plate area. The license plate detection method does not work unless the images are taken from a specific distance and the license plates need to have black text on a white background.
(Lyasheva, Lyasheva and Shleymovich, 2021)	The image is grayscaled, and then a weight image is made. Next, a series of morphological erosion, OTSU thresholding and morphological erosion is done. Finally, contour logic is used to find the license plate area.
(Elhadi, Abdalshakour and Babiker, 2019)	First, a Faster R-CNN Inceptionv2 model finds the license plate area. Next, character segmentation takes the binarised, cropped license plate image and performs some logic. Finally, a custom CNN model is used for character

	recognition. The report claims that Faster RCNN is faster than image processing.
(Radha and Viju, 2022)	The license plate is located using an Adaptive Neuro Fuzzy Inference System (ANFIS). ANFIS is also used for character segmentation and recognition, but recognition binarisation and edge detection are performed before segmentation super-resolution and grayscaling are done. The report also mentions using TensorFlow and tesseract, but where and how is unclear.
(Sasi, Sharma and Cheeran, 2017)	Image processing was used and three methods were compared for locating the license plate. The first uses morphology with edge detection, the second uses ant colony optimisation and the last method uses modified ant colony optimisation. Character segmentation was done using two methods, connected component analysis and a Kohonen neural network. For character recognition, a combined approach using RULES-3 and SVM was used. Here RULES-3 is used to classify the characters into the 36 classes quickly, and the SVM is used to get the character values properly. The report compares the accuracy of each of their defined models.
(El-Latief *et al.*, 2017)	Here, the images are grayscaled, and features are extracted using the sliding window operation. Then using a Gabor filter, license plate candidates are found. Next, template matching is done to reduce the number of candidates, and finally, the remaining candidates are passed into an SVM to find the actual license plate.
(Tang *et al.*, 2022)	This is a systematic review of license plate detection.
(Lubna, Mufti and Shah, 2021)	This is a systematic review of license plate detection, although systematic review is not stated in the title.
(Bhagat and Thakur, 2021)	The license plate, in an image, is located using the Local Binary Pattern algorithm and applying some logic. Then the extracted license plate is binarised, and then edge detection is performed. The subsequent image is deskewed and then segmented. After this, character recognition is done on the segmented characters. The method for segmentation and recognition has not

	been described. The report defines how a seamless end-to-end license plate detection project works but does not detail how each step is performed. The report also states that their method only works on specific plates and will not work if the font or camera quality is different.
(Rokonuzzaman *et al.*, 2017)	This report proposes a method where a robot operating system is used for its modular approach and multithreading capabilities. First, the vehicle is detected using a Haar cascade classifier, and the license plate area is found using another Haar cascade classifier. Then linear interpolation, histogram equalisation, grayscaling and binarisation are performed on the extracted license plate. The characters are segmented and fed into a KNN classifier for character recognition using contours. The report states that vehicle advertisements can lead to false license plate predictions.
(Hsieh *et al.*, 2022)	In this report, YOLOv4 is used to find the license plate area. Then grayscaling, rotation, binarisation, dilation, filtering and logical operations are used to segment the characters. Next, character recognition was performed. The segmentation and recognition process is not mentioned.
(Dhar *et al.*, 2019)	The input image is filtered, binarised and then morphological operations are performed. Next, the image is filtered again and then filtered again with an aspect ratio. Then, programming logic was used to extract the license plate. For character segmentation, the image is binarised, then morphological operations were performed, the shape is verified, and the license plate tilt was corrected. The characters are segmented using the color information and recognised using HoG and LBP features trained in a AdaBoost Classifier. The report compares the method with other papers. The report also states that the method only works when the license plate font colour is black and the background is white.
(Roy *et al.*, 2015)	This report is mainly for scene text. The text is classified using a Bayesian classifier and connected component analysis. Finally, recognition is done after pre-processing the image and passing it into tesseract.

(Onim *et al.*, 2022)	First, the vehicle in the image is detected using an extended NASNet-Mobile model that they customise. Then the license plate area was located using an extended Inception-v3 model. Finally, the cropped area is fed into a custom model, which includes the image pre-processing for recognition as part of the model. The method uses transfer learning of NASNet and compares their method with other papers.
(Neto *et al.*, 2015)	The license plate is located using gradient operation, canny edge detection, hough transform, rectangle detection, and finally, applying some logic. Character segmentation is done by applying a mathematical function after thresholding. Finally, the least squares, least mean square and a multilayer perceptron classifier trained on an extreme learning machine are used for character recognition. The report also states that their method only works in short distances and will not work if there is too much tilt and imperfect illumination.
(Wang *et al.*, 2022)	This report shows segmentation as if it were an object detection problem. Character segmentation is done using Faster-RCNN built on Inceptionv2. Then cross-class removal is done using logical operations, and template matching is used to get missing or remove extra characters. Template matching is done by comparing predefined license plates. If even one character is found, then template matching can infer all characters. Finally, character recognition is done using a broad learning system: two systems, one for digit and letter recognition, are built. The report compares their method with other papers in different environmental conditions.
(Zhang *et al.*, 2021)	This report is a survey where they look at four different methods of license plate detection. They compare image processing with the tesseract method, YOLOv3 with CRNN, which uses a combination of CNN and LSTM, RPNet model for location and segmentation, and a pooling ROI with a KNN to recognise characters.
(Selmi, Ben Halima and Alimi, 2017)	First, the image is converted from RGB to HSV format, then morphological operations are performed, and Gaussian blur is done. The license plate area is

	found using adaptive thresholding, contours and geometric filters. Then the processed image is passed into a custom CNN built in TensorFlow to find candidate regions. The actual license plate is found using logic. After this, the image is grayscaled, and contrast maximised. Canny edge detection, contours and geometric filters are used to segment the characters. The recognition is done using a custom TensorFlow CNN. The report compares their model with other papers.
(Sarif *et al.*, 2020)	The license plate area is detected using YOLOv3. Then the extracted area image is binarised, and a greedy graph algorithm is used for segmentation. Finally, a custom CNN is used for recognition. The report compares their model with other papers.
(Lien *et al.*, 2019)	The license plate is located, and characters are segmented and recognised using separate YOLOv2 models. In this report, they build WebGL to modify images to make them into different environmental conditions artificially, like rainy weather images, night time images and a license plate with large tilt. The method for segmentation and recognition is not properly stated.
(Selmi *et al.*, 2020)	The license plate locating is done using a Mask-RCNN, and character segmentation on the extracted image is done on a Mask-RCNN. The Mask-RCNN after segmentation gives extra ROI; thus, logic is used to remove the false detections. Finally, Mask-RCNN is used in recognition. Mask-RCNN is used overall since features won't need to be re-extracted. The report compares the model with other papers and also states that the model doesn't work in complex scenes, low resolution and bad illumination. Overall the model is slow.
(Zhang *et al.*, 2019)	The license plate is found using sobel edge and colour features. If the license plate area is not located using the first method, then the image is put into an MSER for license plate candidate detection. Next, SVM is used to ensure that only license plates are taken. Finally, grayscaling, OTSU thresholding and contour logic are used for segmentation. The segmented characters are passed

	into a modified LeNet-5 for recognition. This method doesn't work with license plates of all formats.
(Aljelawy and Salman, 2022)	The license plate is located using three different methods YOLOv4, a cascade classifier and a Haar cascade classifier. The characters are then recognised using EasyOCR.
(Al-Shemarry and Li, 2020)	The images are grayscaled, and then the Gaussian filter and ECHE combined with CLAHE are applied. Next, feature extraction is done using MHoG and LBP. Both extracted features are sent to the model. Two methods were used for locating the license plate, SVM and ELM classifier, where the mean shift is used to remove extra ROI. The method proposed is compared with other papers.
(Al-Shemarry, Li and Abdulla, 2018)	The input is grayscaled then Gaussian filter and CLAHE are applied. Three different features are extracted. Each feature is extracted after a single pre-processing step. These features are then passed into an Adaboost classifier for locating the license plate. The method is compared with other papers and in different simulated environments.
(Al-Shemarry, Li and Abdulla, 2023)	First, grayscaling, histogram equalisation and ECLACHE are performed. Then features are extracted using MRELBP and SURF, and the license plate is located using a trained ELM model. The model is compared with other algorithms, classifiers and other papers.
(Kim, Lim and Kim, 2019)	The license plate area is found using image processing, then the common characters, like numbers, are found, and a classifier is used to match the license plate with a predefined license plate. Next, character recognition is done. The proposed method works only with predefined license plates, and segmentation takes a long time.
(Ying, Xin and Wanxiang, 2018)	This paper solves object detection by treating it as a binary classification. A custom CNN and SVM model is used for license plate locating.

(Yao and Yi, 2014)	The license plate is located using an Adaboost classifier and verified using HIS colour checker and SVM. All steps are done; if even one step fails, the area is taken as a non-license plate area.
(Yang, Yin and Chen, 2015)	Since sliding window is computationally complex, thus, they defined a new way to check the pixels. First, the image is broken into smaller rectangles. Then these segments are passed into a deep network built using a sparse auto encoder and softmax classifier. Next, all license plate segments are combined together using gradient density and fixed ratio logic.
(Asif *et al.*, 2019)	The paper uses a different method for license place locating. First, the taillights are found using corona extraction, morphology and then, logic is used to find ROI. Next, grayscaling, Gabor filter and edge detection logic are used to find license plate candidate regions. Then features are extracted using AlexNet CNN, and an SVM classifier is used to verify license plates. The method is checked in different environments. The method does not work if the tail lights are not red enough or are blocked from view. This method also only works on images from the rear end.
(Rabindranath *et al.*, 2022)	First, the image is grayscaled, and Gaussian blur is applied. Next, the license plate area is found using OTSU, and feature logic and segmentation are also done using logic. Finally, recognition is done using a custom CNN.
(Shafi *et al.*, 2022)	First, the BGR image is converted into RGB, the intensity is normalised and Gaussian noise is removed. License plate area is found using YOLOv3, segmentation is done using adaptive thresholding, and recognition is done with Darknet-53 on YOLOv3. The method is compared with other methods.
(L. Qin, W. Wei, and Z. Ma, 2021)	SSD is used to predict and soft nms is used for IOU, while recognition is done with a custom CNN where the last two layers are changed to LSTM and CTC loss.
(Al-Hmouz and Aboura, 2014)	Input is grayscaled, thresholded, and DFT logic is used to get the license plate area.

(Gou *et al.*, 2014)	Grayscaling and top-hat transform is performed, and MSER is extracted to find the license plate area. Segmentation is done using HoG, and values are found using ELM classifier.
(Satsangi, Yadav and Sudhish, 2018)	The license plate area is found using the Viola-Jones Adaboost algorithm
(Kumar, Barman and Pal, 2019)	The input images are converted to HSV, histogram equalised on V, converted to RGB, grayscaled, and sobel operator applied. Next, binarisation and contour logic is used in extraction. The method doesn't work with double rows and different coloured license plates.
(Kassm and Achkar, 2017)	The image is resized, and a pyramid structure is built using a sliding window. This structure is passed into a CNN cascaded classifier for locating. Deskewing is done using CNN, and segmentation is done using contours. Finally, recognition is done using a neuro-evolution CNN.
(Ali *et al.*, 2021)	The license plate area is found with YOLOv5, segmentation is done with custom CNN, and recognition is done using CNN. The method proposed is compared with online LPR APIs. This method does not work in uneven illumination and is more focused on accuracy. The model has the option to ignore Arabic characters.
(Henry, Ahn and Lee, 2020)	Tiny YOLOv3 is used for locating the license plate, and YOLOv3-SPP (spatial pyramid pooling) is used for both segmentation and recognition. Finally, logic is used to get the correct order of license plate numbers. The paper compares their model with many others.
(Min *et al.*, 2019)	K means++ clustering is used to detect vehicles, and YOLO-L is used to find the license plate. Then logic is used only to extract the license plate from the vehicle area. The method is compared with other methods. YOLO-L is a model built based on YOLOv2.

(Sathya, Vasuhi and Vaidehi, 2020)	The license plate area is found with a Capsule network. The training images are augmented, tilted, flipped, etc., before training to be a comprehensive model. The same Capsule network is used in recognition.
(Ashrafee *et al.*, 2022)	The license plate area is found using the Haar cascade classifier and MobileNet SSDv2. The Haar cascade classifier removes unnecessary frames while the SSD takes three images of the same car. Haar cascade is used as an activation feature; only if Haar cascade detects then the image is passed to SSD. Then google vision API is used for recognition. The method is compared with other models on accuracy and speed. The model can't detect far-away license plates.
(Maduri *et al.*, 2021)	The license plate is detected using MobileNetv2 SSD, and the extracted regions are sent to EasyOCR for recognition. The method doesn't work on high-speed vehicles because of low shutter speed on the camera.
(Saranya, Kanagavalli and Priyaradhikadevi, 2021)	First, the vehicles are detected using YOLO. Then, license plates are extracted using grayscaling, binarisation, edge detection, morphological operations, and connected component analysis. Next, line segmentation and word plus character segmentation are performed. Finally, character recognition is done using a custom CNN.
(Nguyen *et al.*, 2021)	A ResNet50 gets the feature maps which an FPN merges, and these features are passed to a triple detector. The output gives confidence and license plate region. Spatial transformation and recognition are performed using the method defined in their previous paper. The proposed method is compared with many other models.
(Chao Gou *et al.*, 2014)	License plates are extracted by logical analysis on the morphologically operated and thresholded image. Then segmentation uses a class-specific ER with a decision tree classifier, and logic is used to remove non-plate areas. A single-hidden layer feedforward neural network (SLFN) with HoG features is used in recognition which is trained using a self-adaptive evolutionary extreme learning machine (SaE-ELM). The license plate area is verified based on the

	number of characters recognised. The method can’t detect partly-occluded license plates or recognise blurred characters.
(Li *et al.*, 2019)	The image is first grayscaled and gaussian filtered. Then MSER is used to find all supposed character areas. Then contour logic is used to select specific contours, and the removal of false area is done using SVM. Finally, SVM is used in recognition. This method directly gets the license plate number and skips the detection step.
(Pourhadi *et al.*, 2022)	License plate locating is done using YOLOv5. The clarity of the cropped area is increased using Selective SR-GAN, and then YOLOv5 is used for recognition.

Discussion

Summary of Evidence

Steps for License Plate Detection

There is no globally accepted value for the number of steps required in a complete license plate detection model. Many papers try to reduce the number of steps (Sathya, Vasuhi and Vaidehi, 2020; L. Qin, W. Wei, and Z. Ma, 2021; Aljelawy and Salman, 2022; Ashrafee *et al.*, 2022) are a few of the examples where the overall method of License Plate Detection has been condensed into two steps. Taking into account the above-mentioned papers, any License Plate Detection model always contains steps for locating the license plate and extracting the number values from the license plate. Although, (Nguyen and Nguyen, 2018) go one more step behind and integrate the full process of locating and recognising the license plate into one step. Overall, it can be said that locating the license plate and getting the text from the cropped image are the usual steps followed.

As such, although both steps are ever-present, both steps can massively affect the performance of the overall model. Most papers agree that correct detection of the license plate area is needed for character recognition to be performed. License plate area detection is the most important step. If the license plate cannot be properly found, thus the license plate number can never be properly found. Yet, although locating is the most essential, character recognition is the hardest step. Character recognition can widely affect the accuracy of the entire model. (Shafi *et al.*, 2022) perfectly displays how character recognition can affect the whole of license plate detection. In (Shafi *et al.*, 2022), they compare their YOLOv3 method with a Mask R-CNN method. The Mask R-CNN method achieves plate detection accuracy of 96.24%, but the character recognition accuracy is only 84.01%. There is a loss of about 12% roughly in terms of overall accuracy because of failure in character recognition. (Shafi *et al.*, 2022) is one of the many papers that faced this problem.

Thus, to reduce the chances of errors, false detections and inaccurate detections, researchers add extra steps to help increase the overall accuracy. Papers like (Rokonuzzaman *et al.*, 2017; Dhar *et al.*, 2019; Kessentini *et al.*, 2019; Min *et al.*, 2019; Saranya, Kanagavalli and Priyaradhikadevi, 2021; N. Ramya, M. Annamalai, and K. Santhosh, 2022; Onim *et al.*, 2022) propose addition of steps like segmentation for recognition and vehicle detection for license plate area detection. Most other researchers implement pre-processing to prepare the image to increase accuracy. (M. S. Al-Shemarry, Y. Li, and S. Abdulla, 2022) put forth a novel pre-processing method to increase accuracy with distorted images.

All in all, the various steps used in various papers during the full process of license plate detection are:

- Pre-processing of Image before License Plate Area Detection
- Vehicle Detection
- License Plate Area Detection
- Pre-processing of Cropped License Plate Area
- Character Segmentation
- License Plate Character Recognition

Input Types

Of the 60 papers, only 13 of them explicitly mention that videos were used as input to their License Plate Detection model. The rest used images for the evaluation of their models. The difference between the use of images and videos is not much since all videos can be considered as a stream of images. But video detection takes more processing because every frame of the video needs to be tested. Thus, the major difference is that those 13 papers in which videos were used considered their models in the real-time scenario. Thus, their models were developed based on this fact.

External Factors That Affect Accuracy

There are many external factors on image quality that can affect the accuracy of the entire system. Most of these external factors come based on the type of camera being used and the environmental conditions.

Camera

The camera is an aspect that greatly affects the performance of the system. (Varma P *et al.*, 2020) states that a still camera with a good shutter speed is required, and (Maduri *et al.*, 2021)'s model finds challenges in detecting high-speed vehicles because of the low shutter speed in the RPi camera. (Sunil, Samuel and C V, 2022) states that at least a medium-quality camera is required for good image inputs, and the accuracy in (Bhagat and Thakur, 2021) was affected because of their low-quality camera. Camera resolution is another aspect that will affect the model's accuracy (Pourhadi *et al.*, 2022), but finding the perfect resolution is weird since low resolution badly affects the model (Sasi, Sharma and Cheeran, 2017) while high resolution from HD cameras requires high memory consumption, are computationally intensive and are expensive to maintain overall (Lubna, Mufti and Shah, 2021). Optimal and good camera settings (Chao Gou *et al.*, 2014; Sasi, Sharma and Cheeran, 2017), good angle placement of cameras (Pourhadi *et al.*, 2022), camera perspective (Sasi, Sharma and Cheeran, 2017) and camera alignment for higher visual coverage (Bhagat and Thakur, 2021) are advocated. Yet more problems can surface from relative motion between the camera and vehicle (Pourhadi *et al.*, 2022) and from image distortions due to optical distortion in the camera lens (Tang *et al.*, 2022). Finally, character recognition highly depends on the existence of clear characters in the image, which can be affected by camera zoom.

Environmental Factors

The weather and noise can affect accuracy values as well. Weather conditions like rain, snow and dust storms make it hard to detect license plates. Even if such conditions are taken into account, the accuracy of character recognition suffers due to blurry or indistinct image input. Extreme reflective glare (Awan *et al.*, 2019) and uneven, poor or bright illumination (Pourhadi *et al.*, 2022) from natural sunlight and artificial light (Tang *et al.*, 2022) can massively affect both area detection and recognition stages.

Pre-processing of Image before License Plate Detection

22 out of the 60 papers choose to perform some form of pre-processing on the input images. It can be noticed that 14 papers performed some common operations like grayscaling, HSV conversion, histogram equalisation, application of sobel operator, edge detection, morphological operation, intensity normalisation, Gaussian blur and noise removal, erosion and dilation. The papers choose to perform some operations and omit others in varying orders. Four papers performed either CLAHE or an enhanced CLAHE for pre-processing (Al-Shemarry, Li and Abdulla, 2018; Al-Shemarry and Li, 2020; M. S. Al-Shemarry, Y. Li, and S. Abdulla, 2022; Sunil, Samuel and C V, 2022) while (Gou *et al.*, 2014) performed a top-hat transform on top of other standard pre-processing actions. Some researchers also choose to carry out very case-specific operations as part of their solution, like image segmentation into smaller rectangles (Yang, Yin and Chen, 2015), corona extraction for tail light detection (Asif *et al.*, 2019) and construction of pyramidal feature structure (Kassm and Achkar, 2017).

Vehicle Detection

It is very common that false plate detection takes place, which may affect the overall accuracy of the system. To solve these false plate detections, seven papers suggest performing vehicle detection right before license plate detection. By performing vehicle detection, either the area of the image that requires analysis is reduced or the authenticity of plate detection is increased. The license plates detected outside the vehicle bounding box are not considered. This system makes sure that sign boards or advertisements on cars are not detected as license plates. Vehicle detection is done using both image processing and machine learning methods, but the majority prefer the machine learning approach. (Awan *et al.*, 2019) used a Faster RCNN (Rokonuzzaman *et al.*, 2017; N. Ramya, M. Annamalai, and K. Santhosh, 2022) used a Haar cascade classifier (Onim *et al.*, 2022) used a Nasnet-Mobile, (Min *et al.*, 2019) used a K means++ clustering and (Saranya, Kanagavalli and Priyaradhikadevi, 2021) used a YOLO model for vehicle detection using machine learning. (Dhar *et al.*, 2019) was the only paper that demonstrated an image processing vehicle detection method.

License Plate Area Detection

License plate detection can be done using either image processing, machine learning or a combination of both image processing and machine learning.

Image Processing Approach

Nineteen papers use image processing to locate the license plates, where mathematical operations or logic are used to estimate the license plate location. Image processing is much faster than machine learning approaches since only math or logic is involved, but image processing approaches don't work in all scenarios (Varma P *et al.*, 2020). (Nguyen *et al.*, 2021) on the other hand, state that even though image processing is faster, they are not very accurate. Proper detection of license plate areas can't take place in environments where the background is not constant or predefined (Ali *et al.*, 2021). If there is less data, then it is preferable to use image processing. Otherwise, machine learning methods are preferable. In a real-time scenario, image processing will always detect something as a license plate, even if no plate is in the frame.

There are five different methods in which license plates are found using image processing which are detections using colour features, edge information, texture-based information, character-based information and specific features extraction.
Detections using colour features become invalid if, in the same image frame, there exists an area where the colour information is similar to that of license plates (Chao Gou *et al.*, 2014; El-Latief *et al.*, 2017; Elhadi, Abdalshakour and Babiker, 2019) like billboards. Colour features are also affected by noise, illumination and camera settings (Chao Gou *et al.*, 2014; El-Latief *et al.*, 2017).
Detections using edge detection are computationally complex, highly affected by noise and don't work well in complex environments (Chao Gou *et al.*, 2014; El-Latief *et al.*, 2017; Elhadi, Abdalshakour and Babiker, 2019). Although morphology can help reduce such errors.
Detections using textures are computationally heavy, and character-based detection is affected by scene text (Elhadi, Abdalshakour and Babiker, 2019).

Machine Learning Approach

Thirty papers implement a trained machine learning model as part of their solution for license plate detection. The methods use either a CNN, SVM, ELM or a classifier.
CNN's were implemented by either building new models, modifying existing models (Min *et al.*, 2019),

making use of the transfer learning mechanism (Onim *et al.*, 2022; Wang *et al.*, 2022) or training custom models using YOLO and TensorFlow.
(Chao Gou *et al.*, 2014) notes that FNN's have slow learning speeds, but it is the most commonly used method. Other papers propose SVM's or ELM's trained on some extracted features like (Al-Shemarry and Li, 2020; M. S. Al-Shemarry, Y. Li, and S. Abdulla, 2022). Lastly, a few other papers implement license plate detection using a classifier like the Haar cascade classifier or the AdaBoost classifier (Nguyen and Nguyen, 2018; Dhar *et al.*, 2019; Ashrafee *et al.*, 2022).

Hybrid Approach

Three papers advocate the use of image processing together with machine learning models for accurate license plate detection. (Nguyen *et al.*, 2021) states that machine learning can be computationally heavy and require expensive supporting devices, and thus using a hybrid method might be better.
(Asif *et al.*, 2019) used grayscaling, Gabor filter, and edge detection followed by feature extraction using a AlexNet CNN used in training a SVM classifier.
(Zhang *et al.*, 2019) used sobel edge and color features or MSER features, based on the output values, passed into a SVM.
(Selmi, Ben Halima and Alimi, 2017) used adaptive thresholding, contours and geometric filter to find candidate regions followed by a TensorFlow model to get the actual plate area.

Pre-processing of Cropped License Plate Area

Direct character recognition of extracted license plate can give vague outputs. This is because while cropping the license plate the image might become blurry and noise needs to be removed again. Yet pre-processing here is not always necessary since some character recognition solutions don't require clear images. (Sathya, Vasuhi and Vaidehi, 2020) states that the best steps are Gaussian noise removal, gradient filtering, OTSU thresholding followed by edge detection, but this doesn't work in all cases. Pre-processing is done based on how the character recognition model is built. (Al-Hmouz and Aboura, 2014) also notes that OTSU requires perfect thresholding values that change based on illumination and other factor.

Character Segmentation

Character segmentation can be done in two ways, using image processing or using machine learning. Image processing uses mathematical functions, operator or feature extraction to quantify and separate the characters. As such most character segmentation using image processing may seem like pre-processing followed by logical analysis for segmentation. (Trapeznikov, Priorov and Volokhov, 2014) uses Harris algorithm and HoG features while (Sasi, Sharma and Cheeran, 2017) uses connected component analysis passed into a Kohonen Neural Network. For machine learning, segmentation is assumed to be an object detection problem and CNN's like Faster-RCNN or YOLO are used in segmentation. (Wang *et al.*, 2022) use Faster-RCNN to find the characters and then use template matching with a predefined license plate to remove false detections and find missing characters.

License Plate Character Recognition

Character recognition is done using either a custom build CNN, a preexisting CNN model, template matching, classifiers or existing libraries. Most methods first segment the characters and pass them one by one for recognition. The most common errors that occur are from false predictions of similar looking characters like { Q, D, O, 0 } , { 1, I, 7, Y } and { S, B, 8 }, to name a few (Liu and Lin, 2017).
15 papers built a custom CNN model and 3 used YOLO trained on custom or publically available datasets. Others used SVM, ELM and classifiers like KNN and AdaBoost for character recognition. Finally others use available libraries like EasyOCR, Tesseract or Google Vision API in their solution.

Challenges

Challenges faced in perfect license plate analysis occur due to the wide variety of license plates available all throughout, the multitude of fonts and diversity of license plate colours. License plates also come in varying shapes and sizes. All these affect not only image processing methods but even machine learning solutions if due consideration is not taken. Low quality images and other external factors also make it impossible to have a perfect solution. Most solutions present are either case specific or only work with one license plate standard (Asif *et al.*, 2019; Das and Kumari, 2021). False detection of license plates caused by billboards and advertisements on vehicles (Polishetty, Roopaei and Rad, 2016; Rokonuzzaman *et al.*, 2017) and incorrect prediction of character values also highly affect overall accuracy.

Limitations

The search terms were reduced to only "License Plate" and "Machine Learning", but the topic of License Plate Detection is addressed by many different titles. Automatic License Plate Detection, Number Plate Recognition, Automatic Vehicle Identification and Car Plate Recognition are different terminologies used is referring to the same solution (Lubna, Mufti and Shah, 2021). Yet the search terms restrict the overall database search to only one category.

The literature search was restricted to ScienceDirect, Web of Science and IEEE Explore thus papers that could contribute to the analysis that might not have been indexed in the above mentioned databases but were indexed in other databases like EBSCO or Scopus were missed.

Some papers were not available for analysis even though they had passed the initial screening process on Rayyan ai, because the papers were either not open access or there was a failure in retrieval of the report.

There be a risk of bias since the systematic review was done by one individual, without any opposing views there might be a small chance that something was missed out.

Conclusions

This paper surveyed many proposed and implemented methods for solving the problem of License Plate Detection. The overall process of License Plate Detection was examined and split into the corresponding steps. Every examined paper was mapped to the steps they performed and the solutions they used in each step were looked at. It can be seen that even though template matching or image processing was the initial way for license plate area detection or character recognition, researcher even now still prefer using template matching like SVM or image processing in their models. It is evident that most papers, even though papers from 2023 were included, prefer to use custom CNN's trained on the COCO dataset or stop at using YOLOv5. No reports were found that tried License Plate Detection using newer technologies like YOLOv7.

In terms of efficiency, machine learning is more accurate in complex environments while both image processing and machine learning methods work in case specific scenarios.

In terms of ease of implementation, image processing is easier to use since machine learning methods need a lot of training images, time for training and fine-tuning for it to perform properly.

In terms of speed, image processing is always faster than machine learning analysis since image processing uses mathematical functions to solve complex problems.

The following paper can be used as a basic to expand upon, researchers can see where others have stopped and continue onward from there. With rapid development in AI and technology over the recent years, especially the recent release of ChatGPT-4, researchers should try to implement newer technologies, like YOLOv8 which was recently released, to build systems that can be all encompassing.

Funding

I, Lowell Lobo, verify that I have no affiliations with or have any involvement in any organisation with any financial or non-financial in the topic discussed in this systematic review.

Requirements specification

Software Development

Project management is crucial for the successful completion of any project. It ensure that project are completed on time, within the budget while fulfilling all the set requirements. Having a proper project structure helps increase efficiency, minimise risk and improve communication.
The project approach used is a combination of waterfall and agile approaches, it is what one would call a Hybrid Model. The Initiation, Planning, Design, Development and Testing follow a linear waterfall stream but internally all tasks were complete using the Kanban method.
Kanban was chosen internally as it is an approach that works even in individual projects as compared to the Scrum model which seems more team focused.

Requirements

To begin any project, the first step is to draft all preliminary requirements. Requirements define the boundary of the project while also alluding to what the final outcome will be. Requirements are crucial and if they are not formulated properly in the beginning it can lead to many problems.

Functional Requirements

Functional requirements correspond to the working of the system. It is the functionality expected from the system to be qualified as a full, complete, and working prototype.

- A device to capture images of cars with license plates included
- Detect the parked cars using the device
- Send the captured images to a remote server
- License Plate detection and recognition followed by parking payment validity check
- Store the information with the image on a database
- Display the extracted information to the authorised user

Non-Functional Requirements

Non-Functional requirements are those aspects which are needed to improve the functionality of the system.

- Security- Security is very important especially since the information is to be stored in a database and software needs to be hosted. Secure storage and authorised access to applications.
- Reliability- The license plate detection needs to be precise to ensure that the overall aim and outcome of parking fee payment is achieved and the hosted application work as intended.
- Performance- Performance mainly looks at the speed and response time. The web application and the ML pipeline need to have fast response time.
- Portability- Image capturing software is able to run on different microprocessors.
- Compatibility- The hosted application should work effectively in different environments. Web application working on different web browsers and browser sizes.
- Serviceability- The software and hardware should be easy to upgrade and maintain with no drawbacks.

Software and Framework Requirements

- Google Colabs- A platform provided by Google where the training and testing of the deep learning models are performed
- VS Code- The code editor which will be used for coding
- Flask- A framework, written in Python, which is used to build the ML backend
- Node.js- The base framework used in JS web development
- Next.js- The JS framework used to build the web application
- Tailwind CSS- A class-based CSS library to design and build web pages
- Arduino IDE- The development environment for the Arduino UNO

Cloud Hosting

- Hugging Face Spaces- The platform used to host the Flask ML API Backend which provides ample resources for deep learning applications
- Vercel- The platform which is recommended for deploying Next.js applications
- Supabase- The online database for storage on SQL data and images
- GitHub- The service used for version control and data sharing across many users and environments

Language Requirements

- Python- A widespread language used to train deep learning models, run inferences and is used to build the Flask backend
- C/C++- A low-level language used in Arduino IDE to program the Arduino UNO
- TypeScript- The language used in the development of the Next.js web application

Hardware Requirements

- Raspberry Pi 3B- The microprocessor used in the device that captures images. It is a compact device that is perfect for real-time applications that require high processing.
- Arduino UNO
- Pi NoIR Camera Module v2
- Webcam
- Laptop
- Wires
- Motor
- L293D Motor Driver
- Breadboard

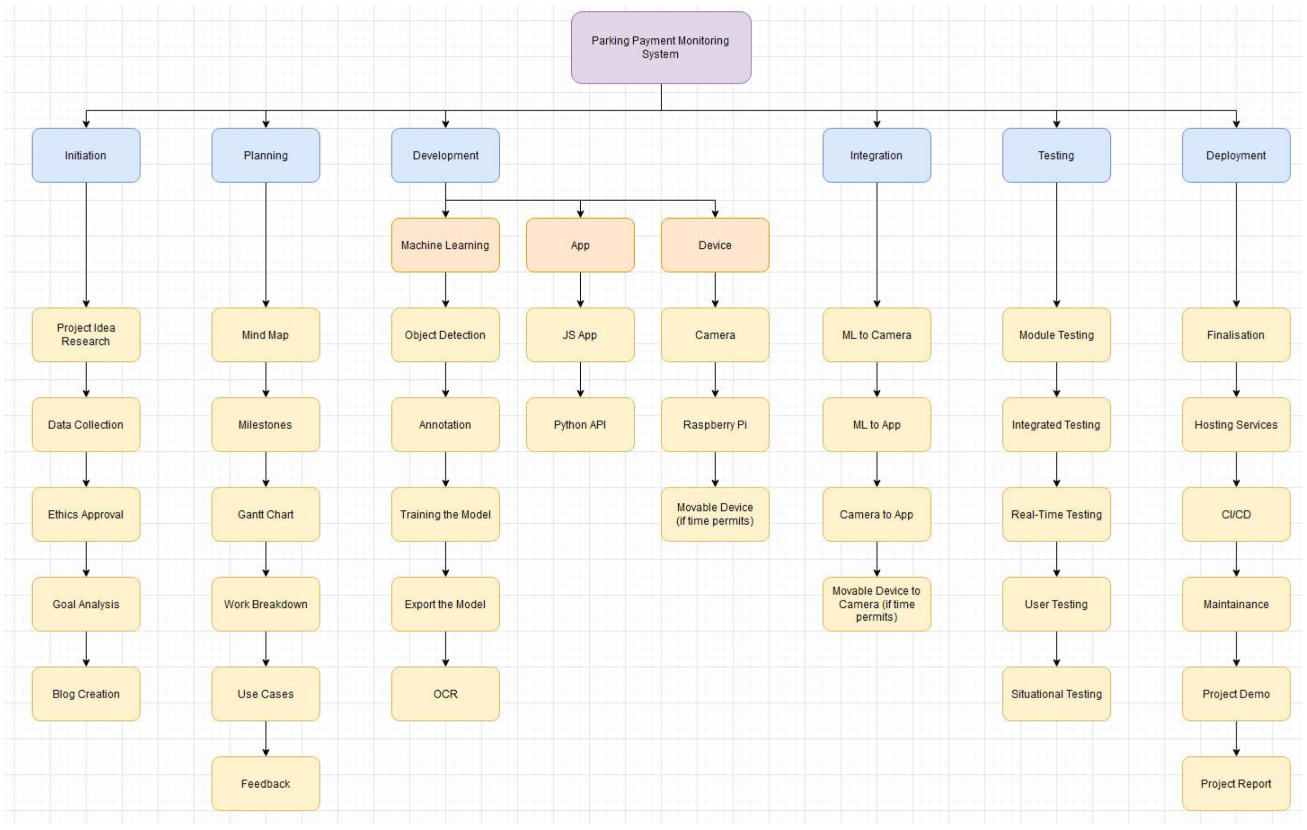
Design & Development

The most crucial part in any project is its design and development phase. The design and development of any project is completely dependent on the requirements defined. In this project, the design and development phase is split into three parts; Preliminary Design, Development and Comparison, and Final Design.

Preliminary Design

The preliminary design is the realisation of the requirements in the most abstract sense. It involves constructing the skeleton of the entire project, the structure to follow during development, and even building UML diagrams like Class Diagram and Sequence Diagram.

Work Breakdown Structure

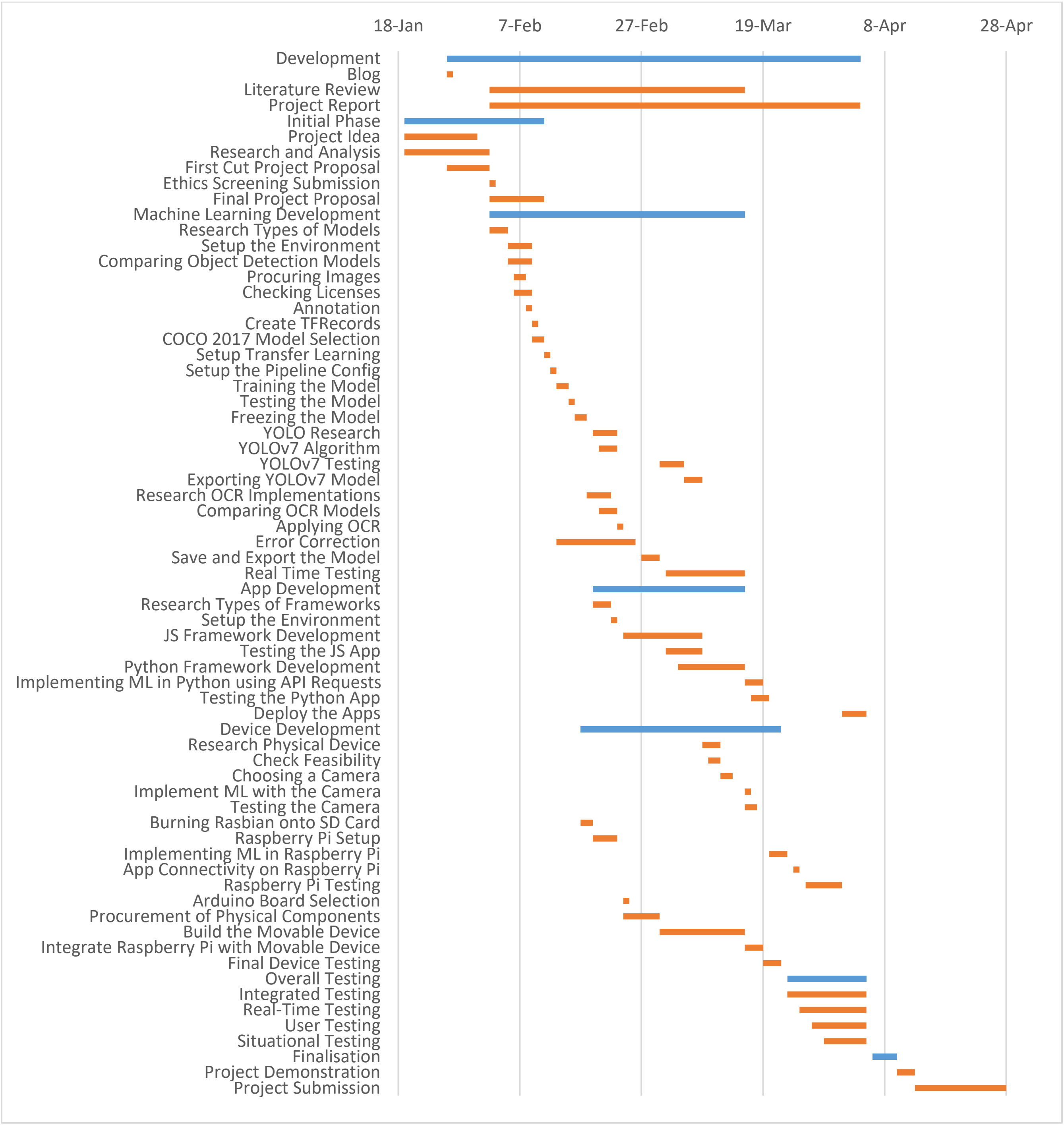


Project Timeline

Task	Start	End	Duration	Weeks
Development	26-Jan	4-Apr	68	
Blog	26-Jan	27-Jan	1	0 Weeks
Literature Review	2-Feb	16-Mar	42	6 Weeks
Project Report	2-Feb	4-Apr	61	8 Weeks
Initial Phase	19-Jan	11-Feb	23	3 Weeks
Project Idea	19-Jan	31-Jan	12	1 Weeks
Research and Analysis	19-Jan	2-Feb	14	2 Weeks
First Cut Project Proposal	26-Jan	2-Feb	7	1 Weeks
Ethics Screening Submission	2-Feb	3-Feb	1	0 Weeks
Final Project Proposal	2-Feb	11-Feb	9	1 Weeks
Machine Learning Development	2-Feb	16-Mar	42	6 Weeks
Research Types of Models	2-Feb	5-Feb	3	0 Weeks
Setup the Environment	5-Feb	9-Feb	4	0 Weeks
Comparing Object Detection Models	5-Feb	9-Feb	4	0 Weeks
Procuring Images	6-Feb	8-Feb	2	0 Weeks
Checking Licenses	6-Feb	9-Feb	3	0 Weeks
Annotation	8-Feb	9-Feb	1	0 Weeks
Create TFRecords	9-Feb	10-Feb	1	0 Weeks
COCO 2017 Model Selection	9-Feb	11-Feb	2	0 Weeks
Setup Transfer Learning	11-Feb	12-Feb	1	0 Weeks
Setup the Pipeline Config	12-Feb	13-Feb	1	0 Weeks
Training the Model	13-Feb	15-Feb	2	0 Weeks
Testing the Model	15-Feb	16-Feb	1	0 Weeks
Freezing the Model	16-Feb	18-Feb	2	0 Weeks
YOLO Research	19-Feb	23-Feb	4	0 Weeks
YOLOv7 Algorithm	20-Feb	23-Feb	3	0 Weeks
YOLOv7 Testing	2-Mar	6-Mar	4	0 Weeks
Exporting YOLOv7 Model	6-Mar	9-Mar	3	0 Weeks
Research OCR Implementations	18-Feb	22-Feb	4	0 Weeks
Comparing OCR Models	20-Feb	23-Feb	3	0 Weeks
Applying OCR	23-Feb	24-Feb	1	0 Weeks
Error Correction	13-Feb	26-Feb	13	1 Weeks
Save and Export the Model	27-Feb	2-Mar	3	0 Weeks
Real Time Testing	3-Mar	16-Mar	13	1 Weeks
App Development	19-Feb	16-Mar	25	3 Weeks
Research Types of Frameworks	19-Feb	22-Feb	3	0 Weeks
Setup the Environment	22-Feb	23-Feb	1	0 Weeks
JS Framework Development	24-Feb	9-Mar	13	1 Weeks
Testing the JS App	3-Mar	9-Mar	6	0 Weeks

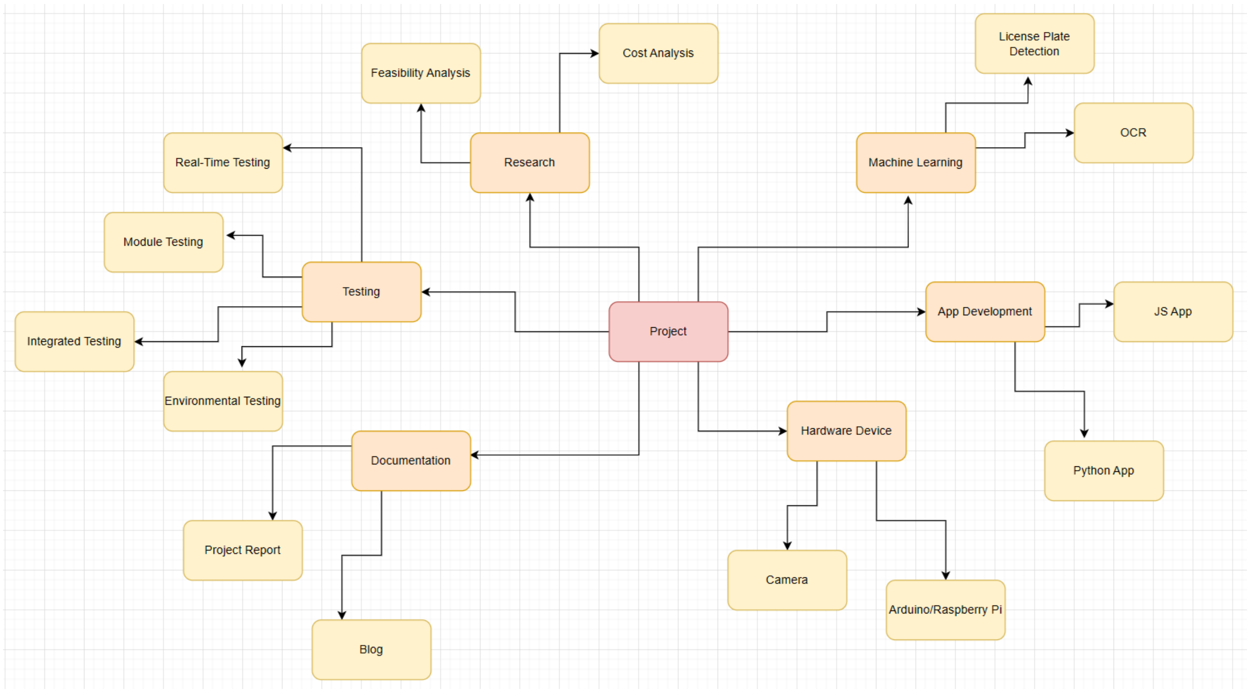
Python Framework Development	5-Mar	16-Mar	11	1 Weeks
Implementing ML in Python using API Requests	16-Mar	19-Mar	3	0 Weeks
Testing the Python App	17-Mar	20-Mar	3	0 Weeks
Deploy the Apps	1-Apr	5-Apr	4	0 Weeks
Device Development	17-Feb	22-Mar	33	4 Weeks
Research Physical Device	9-Mar	12-Mar	3	0 Weeks
Check Feasibility	10-Mar	12-Mar	2	0 Weeks
Choosing a Camera	12-Mar	14-Mar	2	0 Weeks
Implement ML with the Camera	16-Mar	17-Mar	1	0 Weeks
Testing the Camera	16-Mar	18-Mar	2	0 Weeks
Burning Rasbian onto SD Card	17-Feb	19-Feb	2	0 Weeks
Raspberry Pi Setup	19-Feb	23-Feb	4	0 Weeks
Implementing ML in Raspberry Pi	20-Mar	23-Mar	3	0 Weeks
App Connectivity on Raspberry Pi	24-Mar	25-Mar	1	0 Weeks
Raspberry Pi Testing	26-Mar	1-Apr	6	0 Weeks
Arduino Board Selection	24-Feb	25-Feb	1	0 Weeks
Procurement of Physical Components	24-Feb	2-Mar	6	0 Weeks
Build the Movable Device	2-Mar	16-Mar	14	2 Weeks
Integrate Raspberry Pi with Movable Device	16-Mar	19-Mar	3	0 Weeks
Final Device Testing	19-Mar	22-Mar	3	0 Weeks
Overall Testing	23-Mar	5-Apr	13	1 Weeks
Integrated Testing	23-Mar	5-Apr	13	1 Weeks
Real-Time Testing	25-Mar	5-Apr	11	1 Weeks
User Testing	27-Mar	5-Apr	9	1 Weeks
Situational Testing	29-Mar	5-Apr	7	1 Weeks
Finalisation	6-Apr	10-Apr	4	0 Weeks
Project Demonstration	10-Apr	13-Apr	3	0 Weeks
Project Submission	13-Apr	28-Apr	15	2 Weeks

Gantt Chart

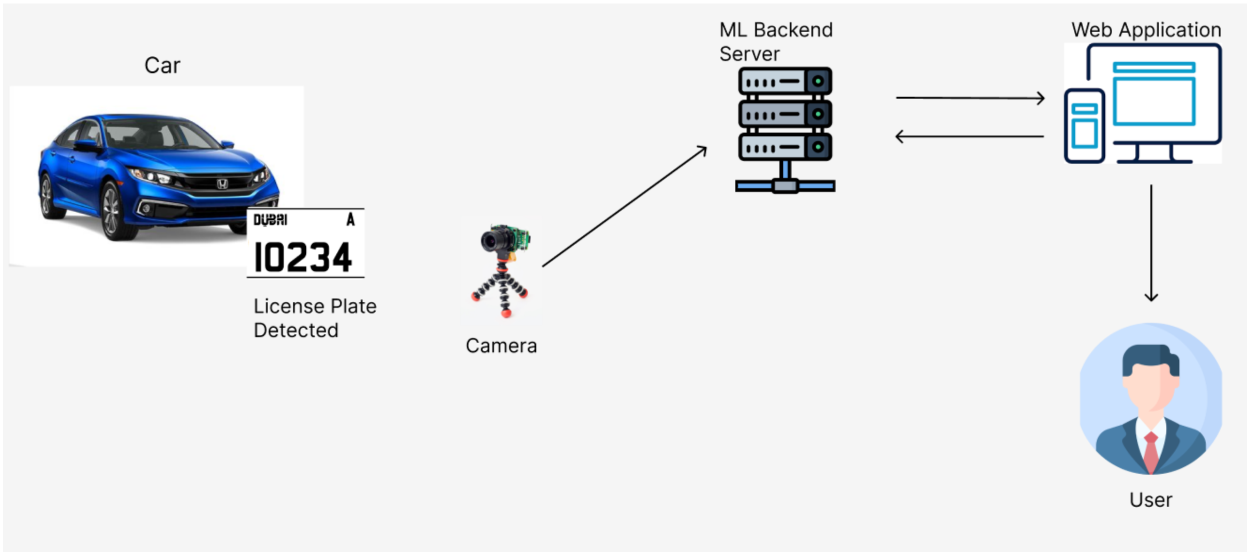


18-Jan
7-Feb
27-Feb
19-Mar
8-Apr
28-Apr
Development
Blog
Literature Review
Project Report
Initial Phase
Project Idea
Research and Analysis
First Cut Project Proposal
Ethics Screening Submission
Final Project Proposal
Machine Learning Development
Research Types of Models
Setup the Environment
Comparing Object Detection Models
Procuring Images
Checking Licenses
Annotation
Create TFRecords
COCO 2017 Model Selection
Setup Transfer Learning
Setup the Pipeline Config
Training the Model
Testing the Model
Freezing the Model
YOLO Research
YOLOv7 Algorithm
YOLOv7 Testing
Exporting YOLOv7 Model
Research OCR Implementations
Comparing OCR Models
Applying OCR
Error Correction
Save and Export the Model
Real Time Testing
App Development
Research Types of Frameworks
Setup the Environment
JS Framework Development
Testing the JS App
Python Framework Development
Implementing ML in Python using API Requests
Testing the Python App
Deploy the Apps
Device Development
Research Physical Device
Check Feasibility
Choosing a Camera
Implement ML with the Camera
Testing the Camera
Burning Rasbian onto SD Card
Raspberry Pi Setup
Implementing ML in Raspberry Pi
App Connectivity on Raspberry Pi
Raspberry Pi Testing
Arduino Board Selection
Procurement of Physical Components
Build the Movable Device
Integrate Raspberry Pi with Movable Device
Final Device Testing
Overall Testing
Integrated Testing
Real-Time Testing
User Testing
Situational Testing
Finalisation
Project Demonstration
Project Submission

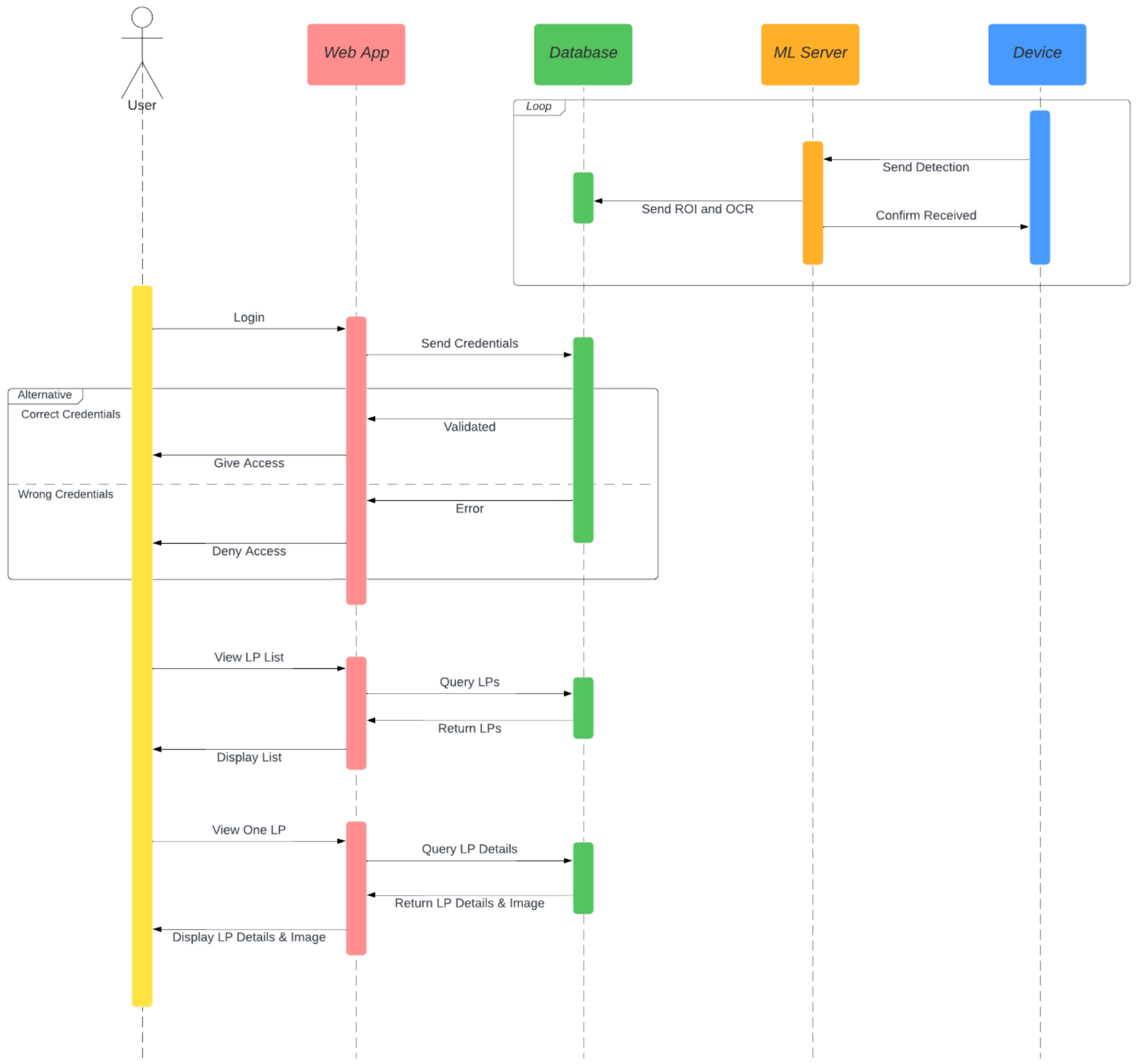
Mind Map



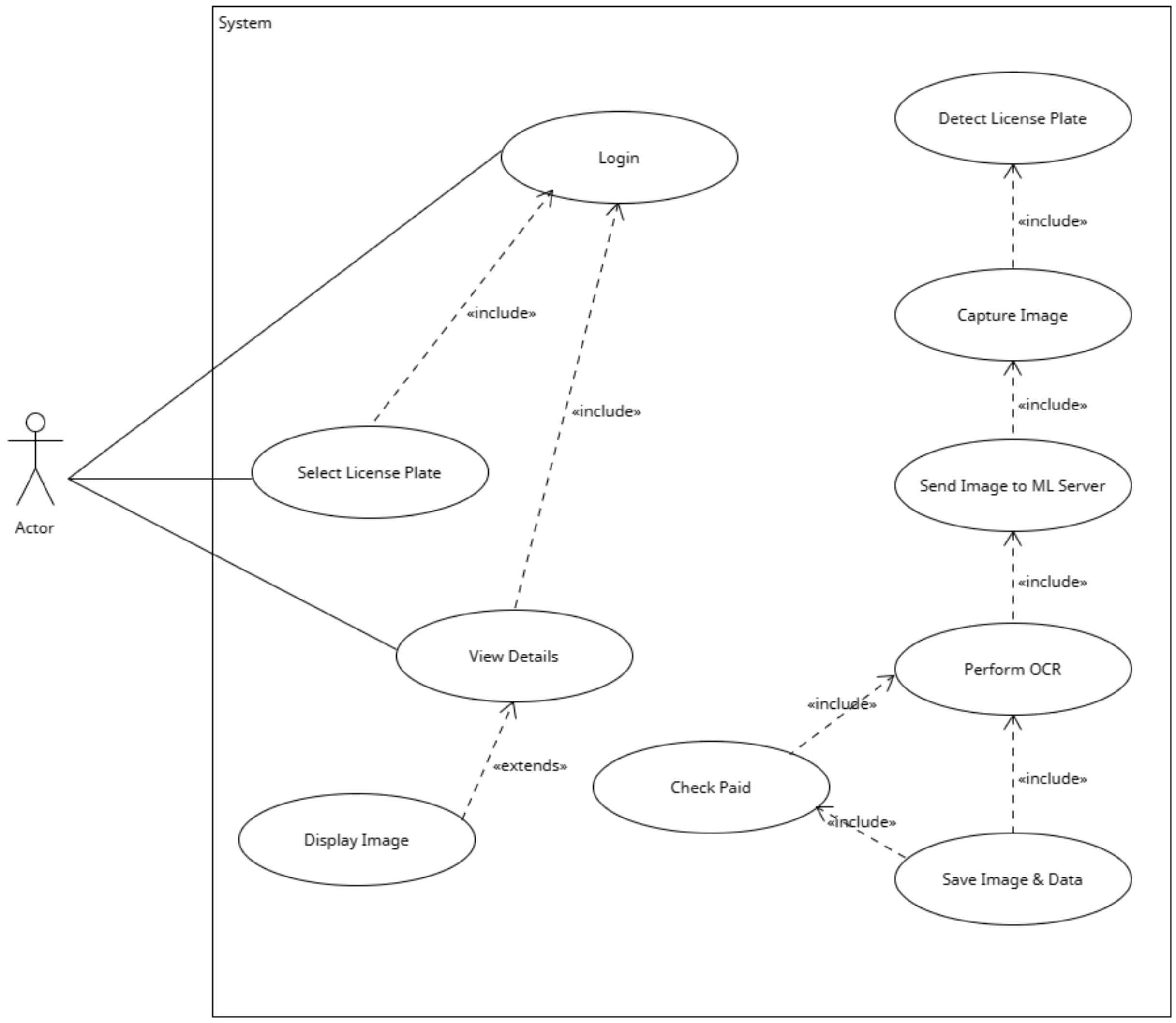
Project Flow



Sequence Diagram



Use Case Diagram



Deep Learning - Preliminary Steps

Deep learning is the most crucial component in this project since license plate detection is most precise when done using deep learning. The deep learning approach used in this project is object detection and training is done using different models. The preliminary steps for deep learning consist of- Procurement of Images and Image Annotation.

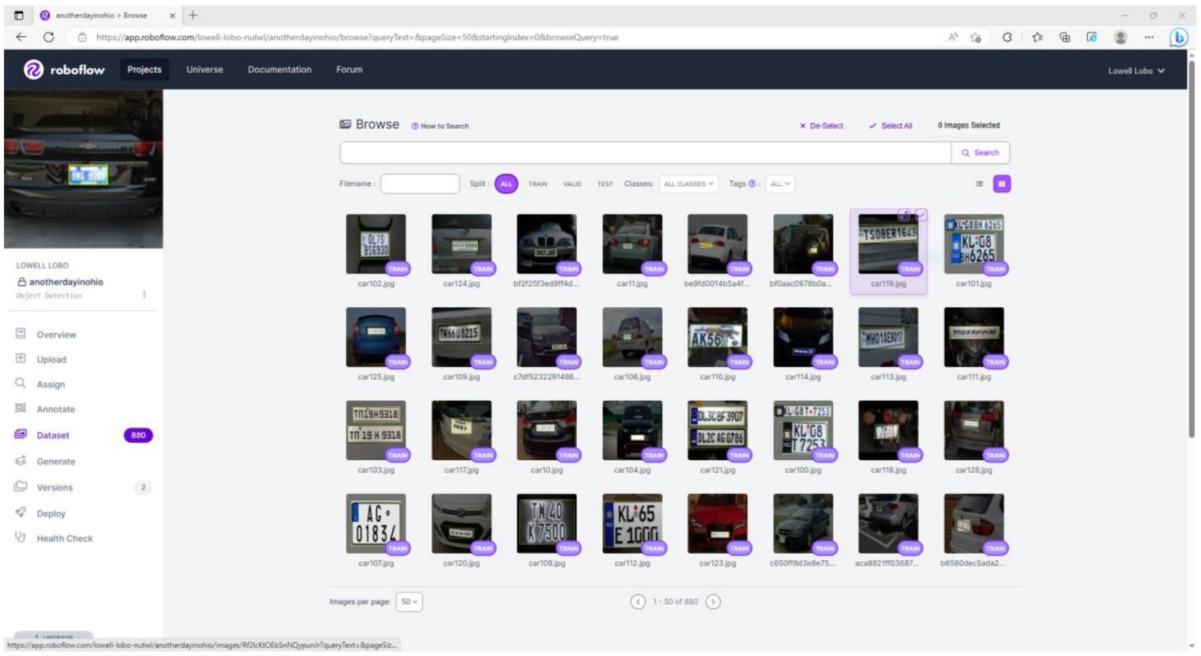
Procurement of Images

(Domingos, 2012) stated that “A dumb algorithm with lots and lots of data beats a clever one with modest amounts of it”. Thus having a greater number of images is most of the time what leads to a good model. But bias can also affect the model heavily, thus images in diff environments are required (Bzdok, Krzywinski and Altman, 2017).

Images of cars with license plates were collected from public datasets in kaggle and roboflow, namely the (Larxel, 2020), (Gosthipaty, 2021), and (Roboflow, 2021) public datasets. The annotations are provided for each image, but they are provided in a single format. Since all these images will be trained using different models, the images need to be annotated into either a universal format or annotated in software that provides easy conversion.

Image Annotation

Annotation of images is done using Roboflow (Dwyer, B., Nelson, J. and Solawetz, J., 2022). Roboflow supports many annotation formats and provides easy conversion of annotated images. Roboflow also provides libraries that make it easy to import these annotated data onto Google Colabs, which helps reduce the overall time taken in training the models.



Environmental Setup

The first step is to link the Google Drive to the Google Colabs Notebook, this is important because Google Colabs deletes any files in the runtime after 30 minutes of inactivity and after 12 hours regardless of use. It is best practise to save any files into the Google Drive immediately after creation.

Next step would be to select TPU as the runtime type. TPU enables faster training for deep learning models, reducing the time needed immensively.

Finally, all libraries are installed based on the requirements set in the documentation.

Device Development – Preliminary Steps

The device is meant to capture images and based on some criteria do something with the image. The device consists of a camera connected to a processor. Since this project will perform in a real-time scenario, for resource and cost management, a microprocessor, namely the Raspberry Pi will be used as the core processor. The Raspberry Pi is connected to a camera that will regularly check for parked cars. The detection of a parked car is done using license plate detection. If a license plate is detected in the video feed, the Raspberry Pi will take that frame of the video feed and send the frame to a Deep Learning Backend for further processing. The Raspberry Pi device acts like an ML Activation, where only if a license plate is detected in the video frame, then and only then the frame is sent to the backend.

Setup

To setup a Raspberry Pi, an SD card needs to be flashed with an operating system that is supported on the Raspberry Pi. The Raspberry Pi will boot using the SD card to perform operations. The SD card can be flashed using the Raspberry Pi Imager software (‘Raspberry Pi Imager’, 2023).

After flashing and booting using the SD card, libraries can be installed on the Raspberry Pi. The process of library installation for machine learning is not straight forward since most of them take long to build and install on the Pi. Thus a GitHub repository was built to help install ML libraries for future use (Lobo, 2023). The repository allows only 3.9 Python library installation for 64bit Rapsberry Pi systems.

Hardware Components

The device is mainly built using a camera and Raspberry Pi, but components like Arduino UNO, motor, L293D motor driver and a breadboard have also been used. The latter components are used to propose and working robotic system implementation on the current project.

App Development – Preliminary Design

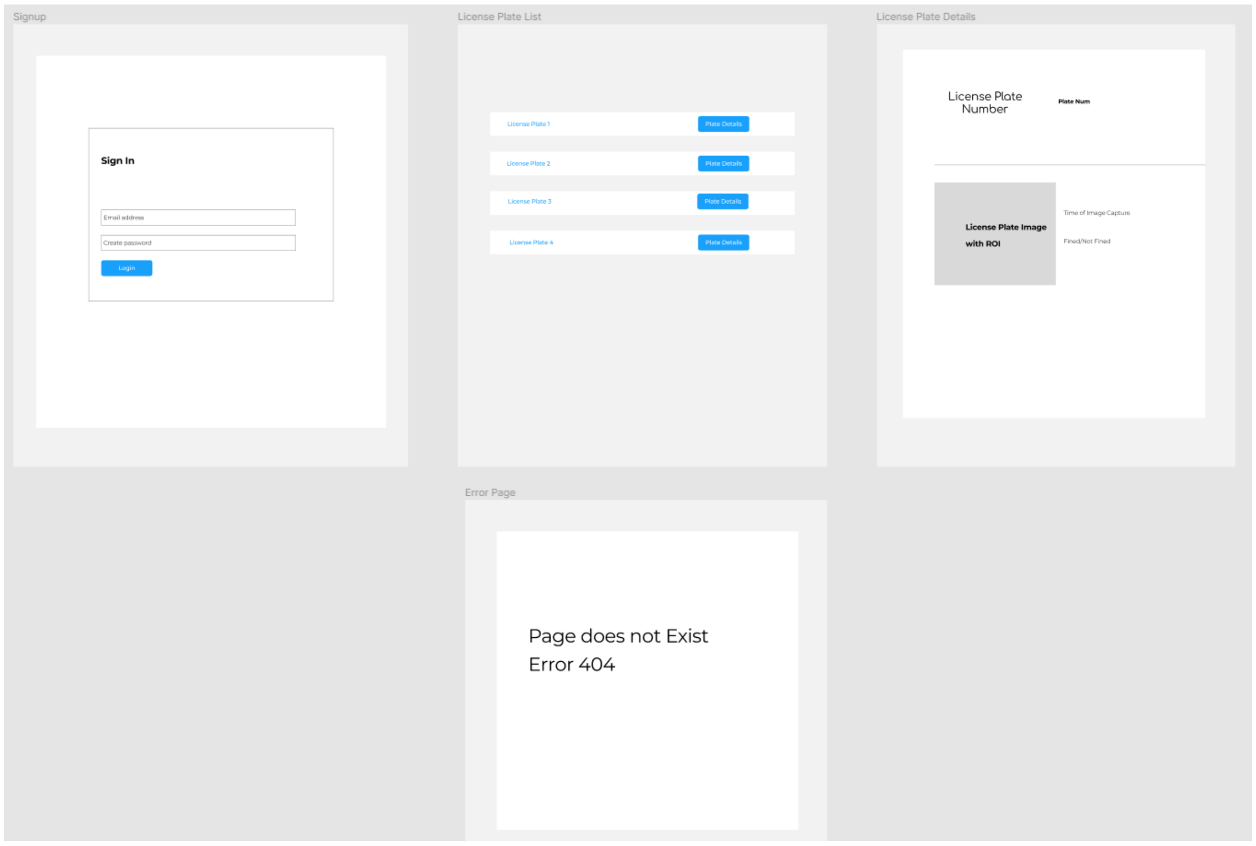
Web Application

A web application is needed to help authorised users to be able to access the system. The web application should enable the user to,

- Authenticate to be able to access the application
- View the number of license plates capture and the number of fines levied
- View details of each captured license plate
- Log out of the application

There are many different languages and subsequent frameworks used in web development. In this project, the web application is built using a JavaScript Framework, Next.js. JS frameworks are widely used, have a lot of supporting tools, provide seamless routing and help in building interactive user elements. Next.js especially is very easy to pick up for beginners.

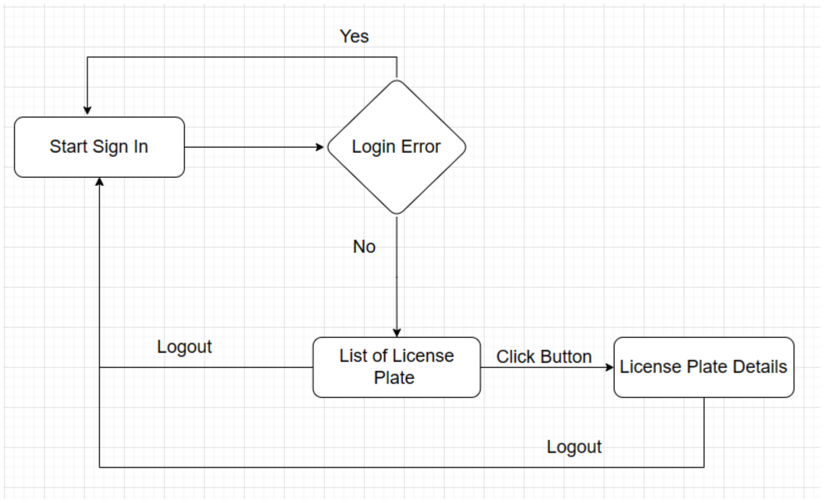
Wireframe



Provided above is the wireframe for the web application.

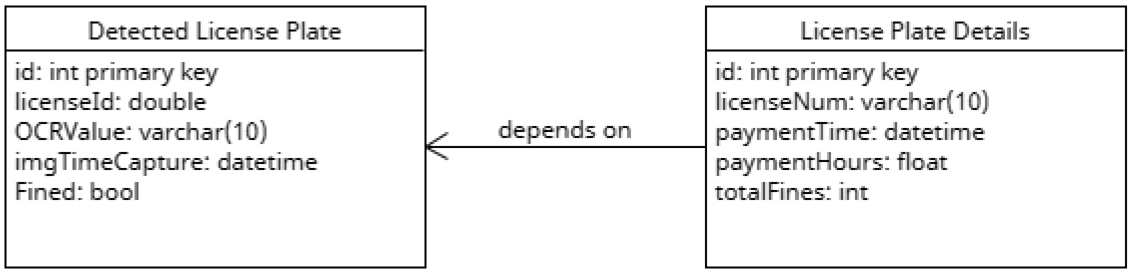
Users will first need to Sign in to gain full access to the website. After sign in, the user will be shown the License Plate List page, where all the license plates that had been captured by the device will be displayed. If the user wishes, they can click on the button to view the full details of the captured license plate. Finally the user can sign out of the website to ensure secure access to the application. There is also an Error Page which will be displayed if a non-authenticated user tries to access the site.

<u>User Flow</u>



All information will be queried from a securely hosted database and then displayed on the web application. The database selected in the Supabase real-time database which provides many easy to use libraries with Next.js which include querying, sessions and even authentication.

<u>Class Diagram</u>



There are two tables as part of this design. The License Plate Details table contains the parking details of each car, while the Detected License Plate table stores the captured images from the device and any information that is inferred from each image.

<u>Setup</u>

Web Application

To be able to start using the Next.js framework, first Node.js needs to be installed. Node.js is a server environment that provides many JS tools for web development. A Next.js project is made by using Node.js and VS Code is used to start the development.

Supabase

To be able to use Supabase, first an account needs to be created on the Supabase app portal. After account creation, Supabase allows users to create and manage different projects to separate between different application. The project is created and using the SQL Editor all the tables are also built.
Next step is to enable RLS, which is Supabase's security features, and then define some policies that allow applications to access, use and edit the previously created tables.

Deep Learning Backend

A deep learning backend receives the images sent from the device, performs license plate detection and levies fines based on extracted information. Having the deep learning backend separate from both the web application and the device enables high speed processing of the images. The complete processing cannot be performed on the device itself since processing will take time, and if during the processing another license plate is supposed to be detected, the detection will not occur since the CPU is busy with processing. This may not always occur, but on the off chance it does, it will decrease the reliability of the system. The processing cannot be performed on the web application as well since this can lead to slow rendering of user pages which is bad UX. Thus a dedicated backend solely for machine learning is required. This deep learning backend can also be hosted on a platform with more resources provided to it, separate from the web application, which can help in cost saving.

The backend is built using Flask, which is a Python framework for building APIs. Python provides a huge assortment of machine learning libraries which can be used in tandem with Flask. Flask, like all Python libraries, are easy to install and easy to use.

Development and Comparison

The development and comparison phase is the formulation based on the preliminary design. Different methods and techniques for solving the scenarios will be looked at, compared and finalised.

Deep Learning

Following the annotation, the images will be trained using different methods. The chosen methods are:

- MobileNet SSDv2
- YOLOv7
- YOLOv7 Tiny

Each of the methods are trained on the same images, no addition or deletions. This is done to keep a similar benchmark for comparison for all the models. Even the training and testing images split is kept the same.

Training the Models

Training of all models can be done using the documentation as reference. The Mobilenet SSDv2 requires the TensorFlow libraries for training while the YOLOv7 and YOLOv7 Tiny require the Torch library. Detailed steps for training are provided in each documentation, and it is even possible to import most example training scripts to Google Colabs. The only changes that need to be done are that the label map, image directory and output directory need to be changed.

First the images and the annotated formats are retrieved from roboflow using roboflow libraries. The annotation format depends on the type of training that needs to be performed. TensorFlow requires the TFRecords format while Torch requires the Pascal VOC format.

TensorFlow Training Steps

After procuring the images, a new file needs to be created called a label map which is used to map annotations to their corresponding labels and then the input and output directories need to be set.

For MobileNet SSDv2 training, transfer learning is performed. First the COCO 2007 trained model is imported from the TensorFlow Model Zoo and then training is performed on top of it for license plate detection. Training was done for 20,000 steps and loss metrics was calculated and displayed graphically using TensorBoard.

Torch Training

Torch doesn’t require any separate label mapping file, it is able to auto-map using the annotated file itself. To train using Torch the corresponding CNN Model structure is either imported or created, and then the training script is run for license plate detection.

Exporting the Models

TensorFlow Export

By default, TensorFlow provides a Saved Model format which contains the weights graph for inference. If necessary the Saved Model format can be converted into the TensorFlow Lite fomat. TensorFlow Lite is a lite model that can run on devices with low hardware resources. It has faster inference time but sacrifices accuracy in the process.

Exported formats,

- TensorFlow Saved Model
- TensorFlow Lite

Torch Export

By default, Torch provides the PyTorch format for inference. Torch also provides inbuilt methods for conversion of PyTorch into different formats.

Exported formats,

- PyTorch
- ONNX
- TorchScript

Model Inference

To run inference using the exported models many different libraries are available. But each format requires their own library for inferencing. TensorFlow Saved Model and TensorFlow Lite use the TensorFlow library, PyTorch and TorchScript inference is done using the Torch library and finally ONNX inference is done using the onnxruntime library.

The code for inference can easily found in the documentation or in example GitHub repositories (ibaiGorordo, 2023) (ultralytics, 2023). Overall the main methods used in inference are,

- MobileNet SSDv2 using TensorFlow Saved Model
- MobileNet SSDv2 using TensorFlow Lite
- YOLOv7 using PyTorch
- YOLOv7 using ONNX
- YOLOv7 Tiny using PyTorch
- YOLOv7 Tiny using ONNX

Device Development

For development, the Raspberry Pi 2 and the Raspberry Pi 3B were used and compared. The Raspberry Pi 2 has an armv1 chip which can only run 32 bit operating systems, while the Raspberry Pi 3B has an aarch64 chip which can run both 32 bit and 64 bit operating systems. Both Raspberry Pi's come with preinstalled 3.9 Python. But this leads to problems with the Raspberry Pi 2.

Most upcoming libraries release their build wheels on latest Python versions, 3.8 and above, for the aarch64 architectures. To install libraries on a 3.9 Python system using armv1 would require the libraries to be compiled from source. This takes a lot of time and processing power. There is also a high chance for the system to lag in between the build which will crash the Raspberry Pi, resulting in all build progress being reset. If the Raspberry Pi 2 does crash during compilation then the entire process will need to be repeated. For reference the process takes 12 hours for installing OpenCV on Rasberry Pi 2.

On top of that the Raspberry Pi 2 can only run 32 bit operating systems which is much slower than a 64 bit operating system, especially for machine learning operations.

Camera

Three different cameras were used throughout the project for testing and comparison, they are, the Raspberry Pi Camera Module v2, the Pi NoIR Camera Module v2 and a normal webcam. The Raspberry Pi Camera Module v2 and the Pi NoIR Camera Module v2 are Raspberry Pi specific modules. The difference between the two is that the former acts as a normal camera while the latter has a lens without an infrared filter which gives it the ability to see in the dark. Each camera possess a different resolution and focus and thus gives rise to varying inference times for the same code.

Inference

The first aspect to keep in mind before running inference on a Raspberry Pi is the existence of low RAM and the lack of inbuilt any GPU. Thus when inference is performed, it will be performed using the CPU. CPU inference is far slower as compared to GPU inference. As such this leads to heating up of the board and low FPS. OpenCV also causes issues in low FPS scenarios. How OpenCV works is that, when the video capture is begun it stores the frames into a buffer. When the script reads a frame, that frame is deleted from the buffer. If the script reads frame fast enough, as the frame enter the buffer it immediately exits the buffer. But if not, the frame fill the buffer until the buffer is full, where it drops all the frames. This implies that, especially in the Raspberry Pi, if the FPS is low there will be a visible lag between the displayed video and what is taking place and there is a high chance for there to be unanalysed frames which is not reliable.

Of the many exported model formats, four were chosen for inferencing on the Raspberry Pi. They are, TensorFlow Saved Model, TensorFlow Lite, Torch and ONNX. Plus to solve the OpenCV problem a method of inference using threads is implemented (Rosebrock, 2015).

The normal method for inference involves,

- Reading a frame from the buffer
- Converting the frame to the correct format
- Running inference
- Performing action based on the inference

The thread inference method involves,

- Creating a thread for video capture
- Reading a frame from the video capture by requesting it from the thread
- Converting the frame to the correct format
- Running inference
- Performing action based on the inference

Thus in total there are twelve different methods for inferencing on the Raspberry Pi and all methods were tested and compared. The Raspberry Pi will inference over its full lifetime and continue finding license plates and sending them to the Flask API. This inference is considered as an ML Activation, it is the first step in the whole process of license plate detection. This image frame is not taken as a guaranteed detection of license plate, but a probable detection. Proper detection will then be done in the Deep Learning Backend.

Posting Data to API

During inference, when a license plate is found, the frame of the video is sent to the Flask API using a POST request. It is impossible to directly the frame using a POST request. The frame needs to be converted into a format that is supported in HTTP Requests. Image frames are in the ndarray format and must be converted to either bytes or string format. The converted frame is then sent to the Flask API for further inference and analysis.

App Development

Web Development

Next.js simplifies routing and page creation in that it automatically creates views based on files placed inside the pages folder. The website consists of four pages, and thus four files are placed in the pages directory. TypeScript is used to design the pages in the web application.

Pages

The first page is for the signin page. Once the user reaches the signin page, the application will first check if the user is authenticated or not. If the user is authenticated then the page will redirect to the licenses page, else an authentication menu from Supabase auth-ui will render on the screen. The user will have to enter in the correct email address and password to gain access to the full website.

The licenses page is the view that the authenticated user is redirected to. In the licenses page, there is a list of all the licenses that have been captured by the device. The list displays the image URL, the results from OCR and the time when the image was captured. The image URL is coloured red or green based on whether the car owner was fined or not. If the URL is red, then the car owner has been fined else no fine was levied. The user can click on the image URL to view in-depth details of the captured license plate.

When the user clicks on the image URL they are redirected to a page that displays the captured image with the bounding box of the detected license plate. On the side is a table that displays the image URL, the OCR value, the time of image capture, the date of image capture and whether the corresponding plate was fined or not.

Finally, a custom Error 404 page is built with a button to redirect to the signin page.

Routing

Most of the routing is done by having the user press buttons, but there are some cases where routing is performed automatically. There are two scenarios when this occurs. The first is when an unauthenticated user tries to access the website the application will automatically redirect to the signin page. The second is when a user enters a random URL which might not exist in the app. Here the user is redirected based on their authorisation. If the user is unauthenticated, it will redirect to the signin page, else it will redirect to the licenses page.

This routing is performed using Next.js Middleware. Middleware allows developers to run code before the request is completed (NEXT.js, 2022). Then based on the request the corresponding response can be modified. Middleware is highly used in authentication, where if a session cookie is not found it means the user has not been granted a session token, and this session token shows whether the user is authenticated or not. Thus based on the session cookie information the response to the request can be altered which leads to redirects.

Database Connection

The Next.js app only queries information from the Supabase database and no permissions for editing data have been enabled. This is because the web application is purely for viewing all captured license plates. Querying data can be done in two ways, one is using the Supabase Client inside the default function or by using a Server Supabase Client in the Server Side Props Function. Doing so using the Supabase Client can lead to long wait times, thus Server Supabase Client is always preferred. Server Side Props allows the use of pre-rendering which can be use to pre-fetch data from Supabase. The pre-fetched data is then send to the main page as props which need to be destructured for futher use. The data is retrieved in the JSON format and can be used directly since Next.js supports the JSON format.

To be able to create a Client connection to Supabase the Supabase URL and Anon Key need to be set as environment variables. The Client will automatically build a connection with Supabase and requests can be processed through the object constant.

Deep Learning Backend

Flask allows developers to build web application along with APIs for use, but in this project only the API aspect has been used. The API is built such that it is able to receive an image object along with other data, perform analysis on the received image and then save any extracted data onto Supabase.

<u>Request Types</u>

There are two types of requests which have been enabled on the built API, GET and POST request. GET is the default request type, while POST request was added for security. The API will only works with a POST request, if any application were to initiate a GET or any other request type, the API will return an error message.

<u>Request Information</u>

The request is expected to contain a JSON object with three pieces of information. The first is the image frame in the bytes or string format, the second is a secret key and the third is the minimum plate number. The image frame will be used for inference and further analysis since the ML Activation is only considered as a probable license plate candidate. The secret key is used to verify whether the request is valid and the initiation address is legal. The secret key sent is verified with the secret stored in the Backend application for authentication. Finally the minimum plate number is used to verify the accuracy of the OCR. The minimum plate number is a case specific value and will vary based on location. For example if the country possesses license plate with 7 digits the minimum plate number will be 7. This information will be sent from the Device and not preconfigured on the Backend application because the Device may be used in varying parts of the world.

Inference and Analysis

License Plate Detection

Inference is performed in the exact manner as mentioned above, but before inference can be performed the image frame needs to be converted from the bytes or string format to the ndarray format. After conversion, license plate detection is performed. The model used here is a heavier model i.e. the model is more powerful with higher accuracy but has greater resource requirements. Depending on the resources provided the speed of inference also varies. A heavier model is used here to increase the reliability of the system and since the Backend is to be hosted, as long as the resources are provided, there will be no issues.

Depending in the detection confidence, further analysis is performed and if the minimum criteria is not met the image frame is dropped. If the detection confidence is above the threshold then all detected license plates are passed on the next phase.

Preprocessing

After detection, each detected plate is first cropped to only get the license plate region and then pre-processing is performed on the image. Since the image frame was cropped and zoomed there is a high chance for a decrease in resolution. Thus pre-processing helps negate errors that may occur due to the cropping process. Three different methods of pre-processing were tried, tested and compared.

OCR

The text from the cropped and pre-processed image is then extracted using Python libraries. Python provides many libraries for OCR which are easy to use and have good accuracy rates. Mainly, four libraries were looked at, they are,

- Tesseract
- Keras-OCR
- EasyOCR
- PaddleOCR

Each library provides in-depth documentation on how to perform OCR operations and how to modify the inference to suit the requirements.

Database Operations

After OCR has been performed the extracted text value is compared with existing plate values in the database to verify parking payment. The Supabase Python Client library is used to create a connection with Supabase. The library provides a function to create a connection where the Supabase URL and Anon Key are to be provided as parameters. Throughout the API Supabase can be accessed using the Client object. The application is granted permission only to read and add data into Supabase, delete and edit permissions have not been given.

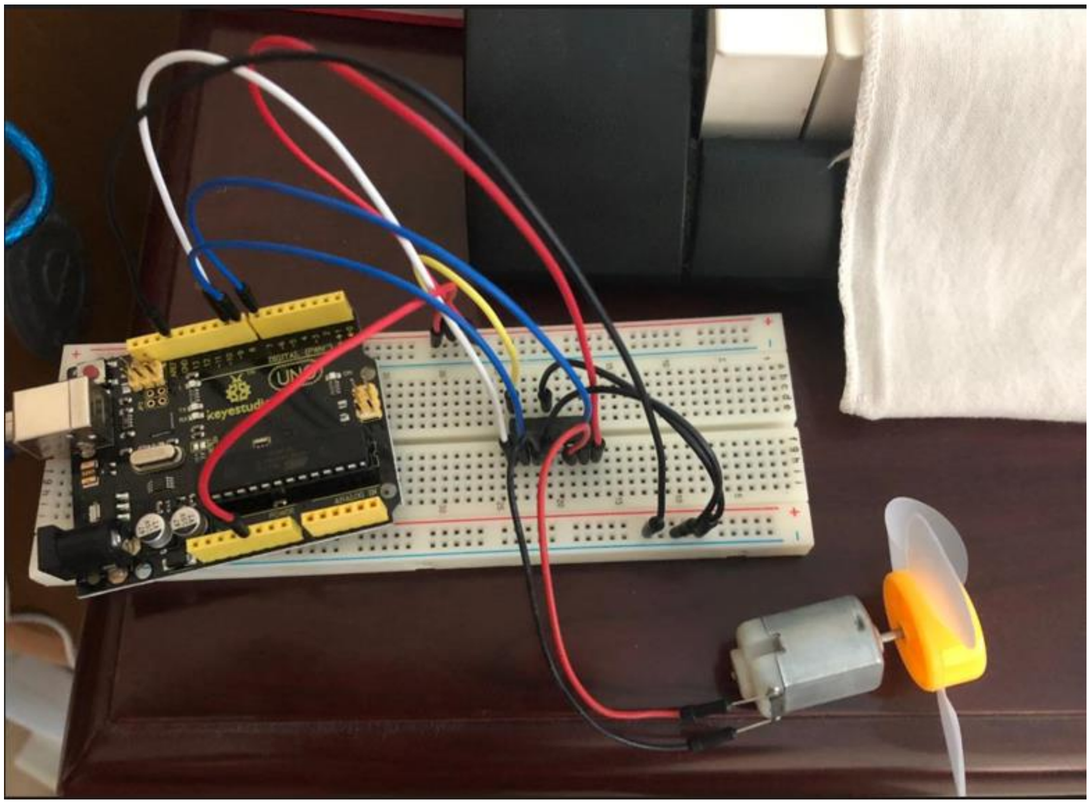
First the extracted text value is used to query the Supabase license plate details. If Supabase returns an empty array, that means the car owner has not payed the parking fees and needs to be fined, else the latest parking time is compared with the current time to see whether the car's parking is within the time frame or has extended. The details that have been extracted will then be saved into Supabase along with the information about whether the car owner was fined or not. Along with the image details, the image with bounding box is stored in a Supabase bucket to be used in displaying in the Next.js application.

Final Design

Hardware

The device consists Raspberry Pi 3B which has a Pi NoIR Camera Module v2 docked onto it. Through the serial post an Arduino UNO is connected to the Raspberry Pi. The Raspberry Pi send commands to the Arduino UNO through the serial port and the Arduino responds by performing some action. The Arduino is connected to a motor through the L293D motor driver placed on a breadboard. The motor driver is used to operate the motor where the motor driver helps to define the speed and direction of rotation of the motor. Based on information from the serial port the Arduino has defined 3 states on the motor. One is where the motor is at top speed, the other where the motor is slowed down and the last where the motor is halted.

The motor usage in the project is purely experimental. It has been used to demonstrate a potential future scope where the device can be made into an autonomous robot. The device is now able to display physical changes purely based on the inference performed in the Raspberry Pi. The Arduino handles movement while the Raspberry Pi will act as the brain of the system.



The Raspberry Pi is connected to a monitor using a HDMI cable and the microprocessor is controlled using a keyboard and a mouse. The power supply is only connected to the Raspberry Pi, and the Raspberry Pi provides power to the Arduino. If in case the power supply is insufficient and alternate power supply can be connected to the L293D motor driver.

Software

Device Inference Script

The Device runs a Python script which can be executed either via SSH or by connecting the Raspberry Pi to a monitor using an HDMI cable. After the script is executed the peripherals can be unplugged but a constant power supply must be maintained.

In the Python script, first, the necessary libraries are imported for use. Next all peripheral and logical components are initialised for use.

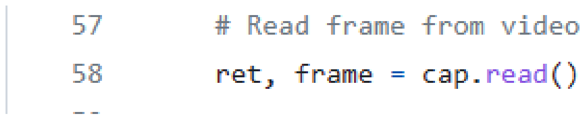
```
11  # Initialise the webcam
12  cap = cv2.VideoCapture(0)
13
14  # Initialise the Serial Port
15  arduino = serial.Serial('/dev/ttyACM0', 115200, timeout=1)
16  arduino.reset_input_buffer()
17  num = '1'
18  time.sleep(1)
19
20  # Initialize YOLOv7 object detector
21  model_path = "models/yolov7-tiny.onnx"
22  yolov7_detector = YOLOv7(model_path, conf_thres=0.5, iou_thres=0.5)
23  frame_rate_calc = 1
24  freq = cv2.getTickFrequency()
25  cv2.namedWindow("Detected Objects", cv2.WINDOW_NORMAL)
26  scores = []
27  arduino.write(bytes(num, 'utf-8'))
```

In line number 11, the webcam is initialised and linked to a cv2 object followed by the serial port connection to the Arduino. In lines 21 and 22, the ONNX model is loaded for further inference.

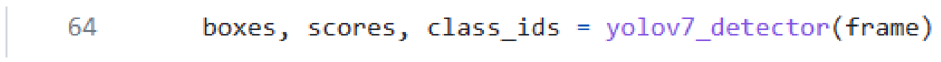
In the finalised design, the ONNX file of the YOLOv7 Tiny model is used in ML Activation. For ONNX inference, especially for YOLOv7, a yolov7 library specifically for ONNX inference was cloned from GitHub and used. The yolov7 library provides the YOLOv7 class which helps simplify inferencing.

During booting, the inference will start and the motor will be set to full-speed rotation.

After all the initialisations, a while loop that runs indefinitely is initiated and constantly the video frame is read and inference in run on the frame.

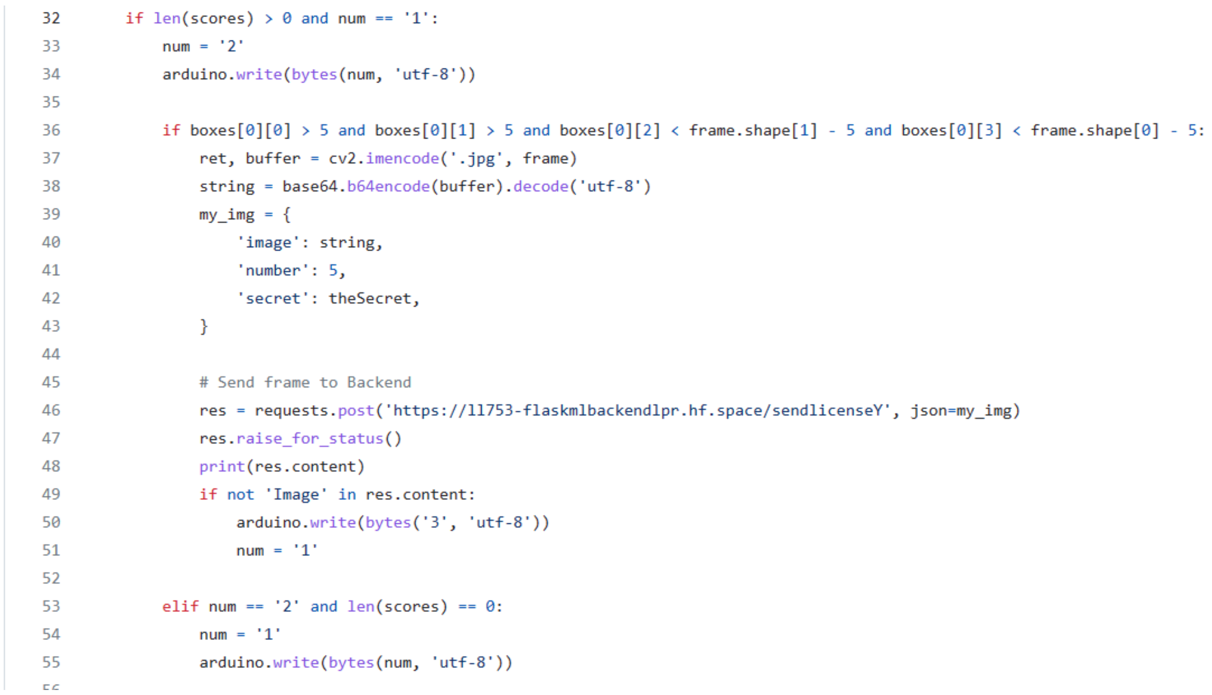


```
# Read frame from video
ret, frame = cap.read()
```



```
boxes, scores, class_ids = yolov7_detector(frame)
```

The boxes variable holds the coordinate values of the area of the probable license plate and the scores variable holds the corresponding confidence percentages.



```
32  if len(scores) > 0 and num == '1':
33      num = '2'
34      arduino.write(bytes(num, 'utf-8'))
35
36      if boxes[0][0] > 5 and boxes[0][1] > 5 and boxes[0][2] < frame.shape[1] - 5 and boxes[0][3] < frame.shape[0] - 5:
37          ret, buffer = cv2.imencode('.jpg', frame)
38          string = base64.b64encode(buffer).decode('utf-8')
39          my_img = {
40              'image': string,
41              'number': 5,
42              'secret': theSecret,
43          }
44
45          # Send frame to Backend
46          res = requests.post('https://11753-flaskmlbackendlpr.hf.space/sendlicenseY', json=my_img)
47          res.raise_for_status()
48          print(res.content)
49          if not 'Image' in res.content:
50              arduino.write(bytes('3', 'utf-8'))
51              num = '1'
52
53      elif num == '2' and len(scores) == 0:
54          num = '1'
55          arduino.write(bytes(num, 'utf-8'))
```

Line 32 checks whether any license plate have been detected, if yes, the following code snippet will execute.

Line 36 is a condition for edge detection of the license plate. There are times when the model detect half a license plate as a license plate. This will then lead to the corresponding pipeline to execute. But the information provided is incomplete. Thus, the data frame should be send to the Backend API only if the detected license plate is not near the edge of the frame. As long as the detected license plate is away from the edges, it ensures that the full license plate image is being sent to the Backend.

Before sending the frame, the frame is converted into a suitable format using lines 37 and 38. A JSON object is initialised from line 39 to 43 and finally a post request is send to the Backend with all the necessary information.

The num variable is used to communicate what mode should the motor be on and depending on the response from the Backend a corresponding response is sent to the Arduino.

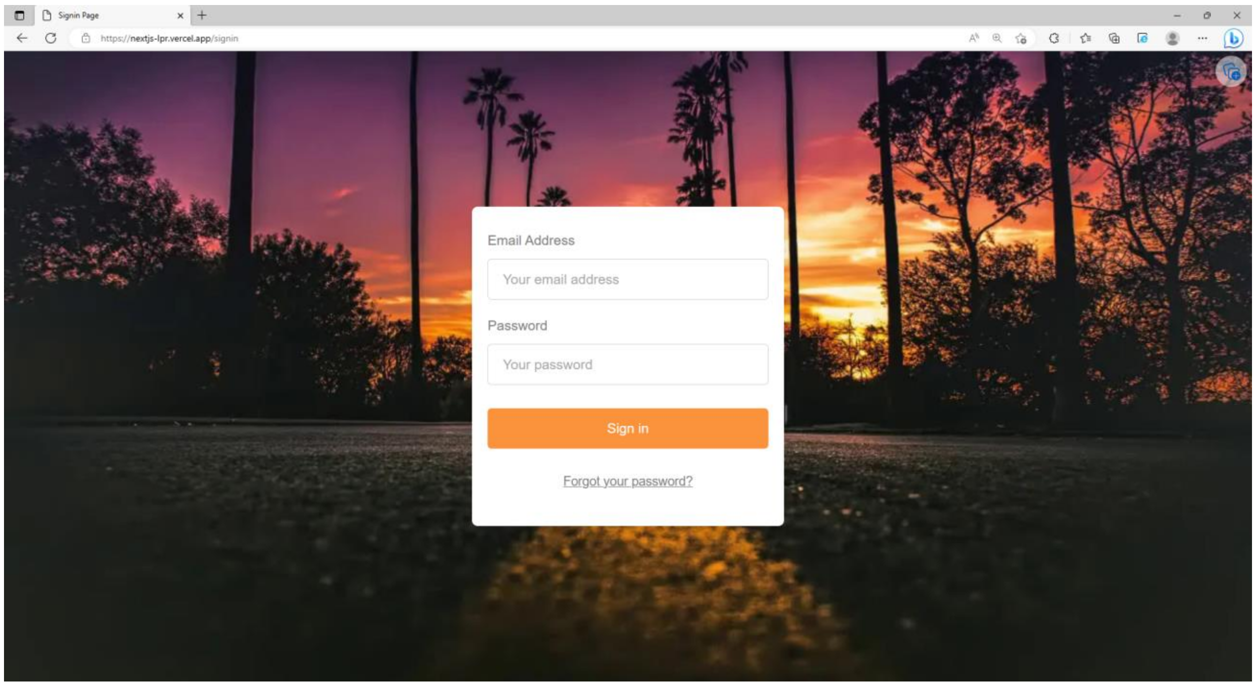
The num variable is also used to lock the system. When a license plate has been detected and analysis is run there might be a chance that the license plate does not leave the frame. In such a scenario the same license plate will be sent to the Backend once again. This can lead to two fines being levied if the car owner has not payed the parking and also leads to redundant data. To avoid this, the num variable is set so that if a license plate has been detected the POST request will be sent only once. Once the license plate exits from the frame, the num variable resets and normal operation will resume.

Next.js Web Application

The web application was built using the Next.js framework where TypeScript was used as the design language and UI was designed using pre-defined classes from Tailwind CSS.

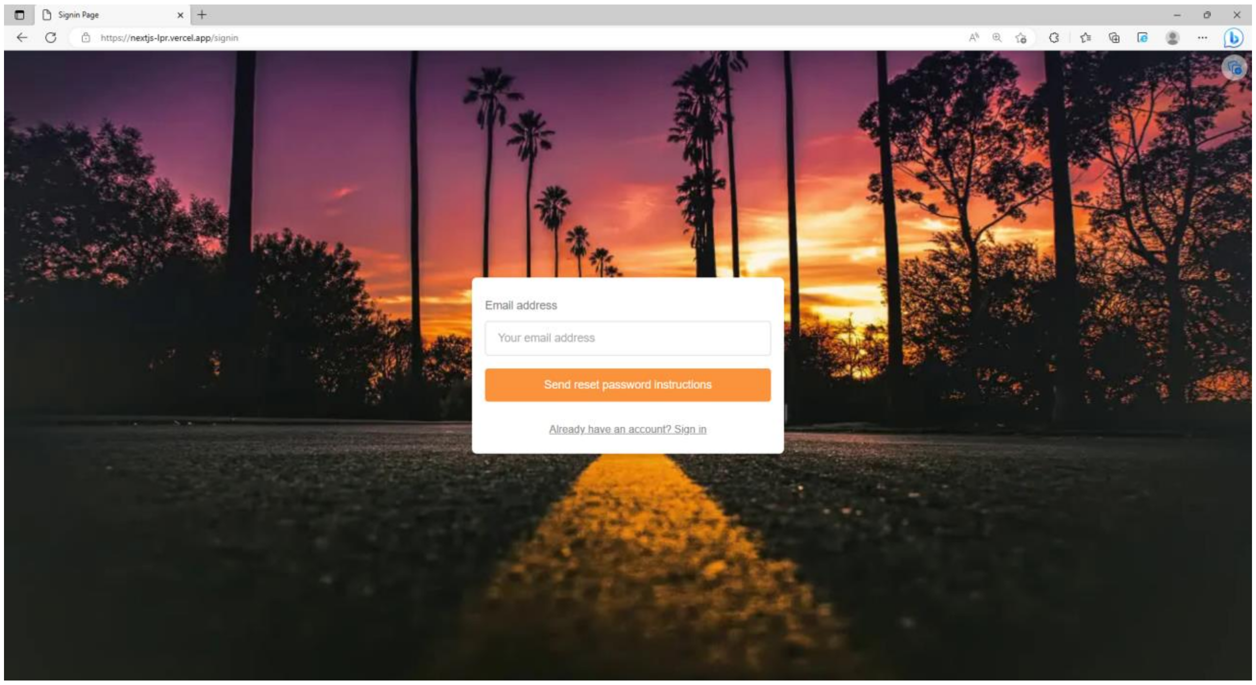
Signin Page

The first page displayed is the signin page. It is the mode of entry for accessing the web application.

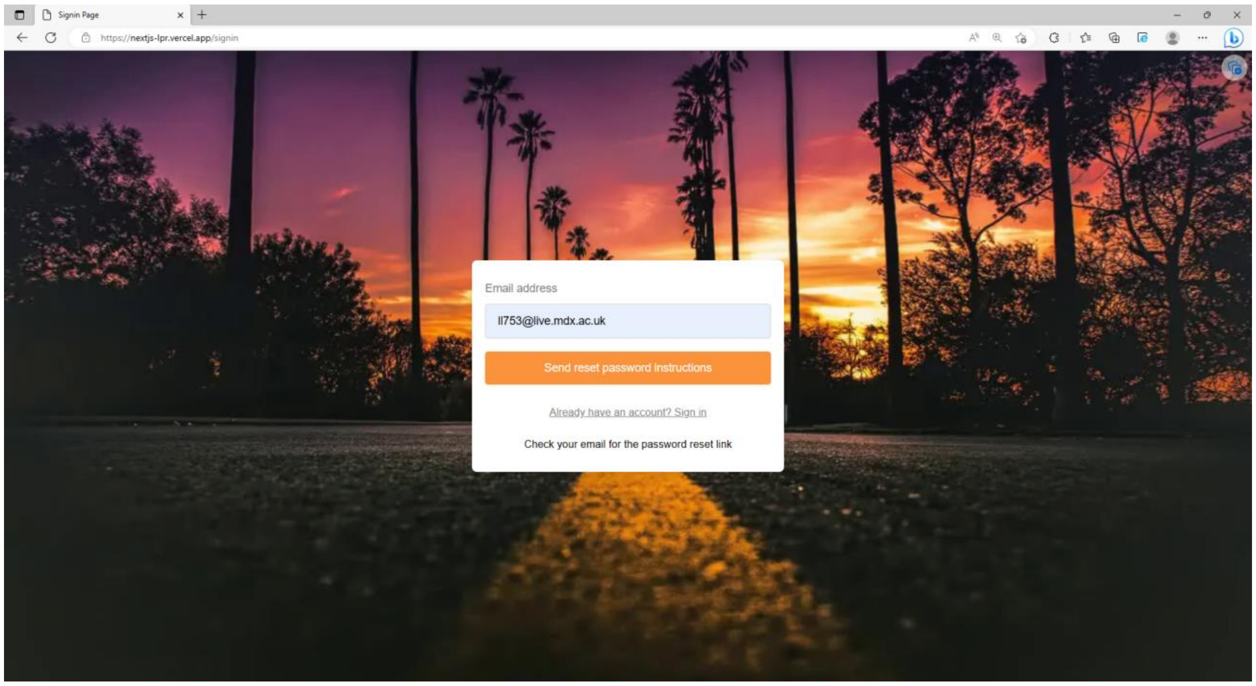


To signin, the correct Email Address and Password needs to be entered into the field. The user can then press 'Enter' or click on the 'Sign in' button to be authenticated. If the given credentials are correct, the user will be redirected to the license page.

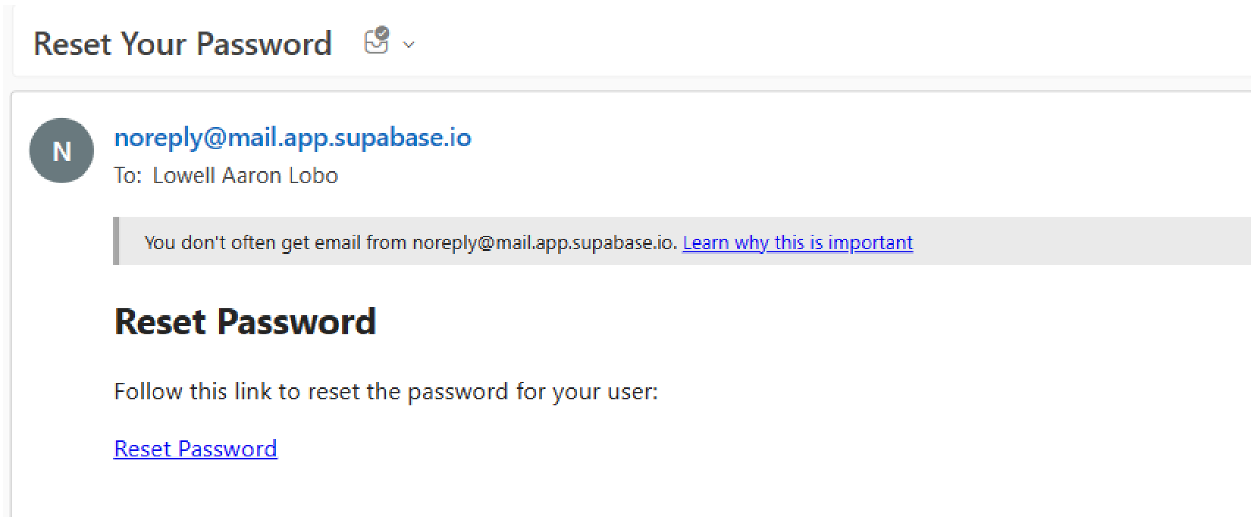
The signin page also possesses a 'Forgot you password?' feature. To be able to use the feature, the user can click on the link below the 'Sign in' button.


Signin Page
https://nextjs-lpr.vercel.app/signin
Email address
Your email address
Send reset password instructions
Already have an account? Sign in

The forgot password page allows the user to enter their registered email address and the instructions to change their password will be sent. There is also a link to return back to the signin page.


Signin Page
https://nextjs-lpr.vercel.app/signin
Email address
ll753@live.mdx.ac.uk
Send reset password instructions
Already have an account? Sign in

Check your email for the password reset link


Reset Your Password

noreply@mail.app.supabase.io
To: Lowell Aaron Lobo

You don't often get email from noreply@mail.app.supabase.io. Learn why this is important

Reset Password

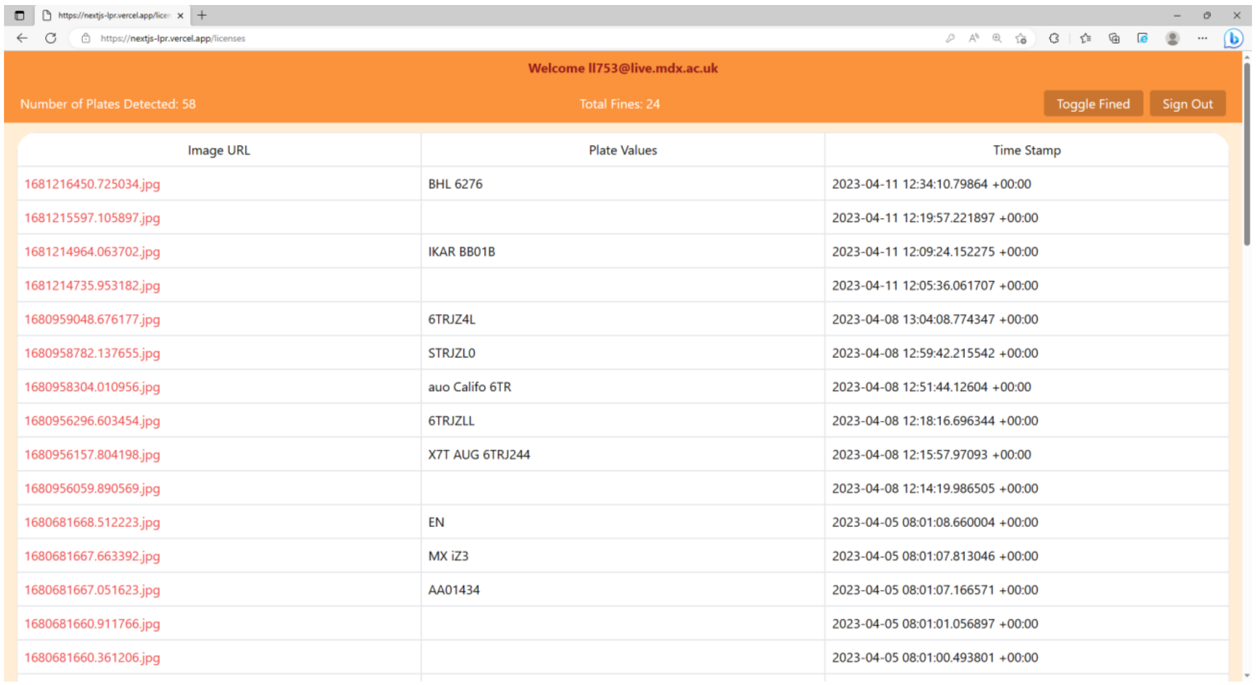
Follow this link to reset the password for your user:

Reset Password

The user will receive an email and can reset their password by clicking on the ‘Reset Password’ link.

<u>Licenses Page</u>

After signin, the user will be sent to the licenses page. This page consists of information about the total number of license plates detected over the whole lifetime, the total number of fines levied and displays a list of all the license plates detected.

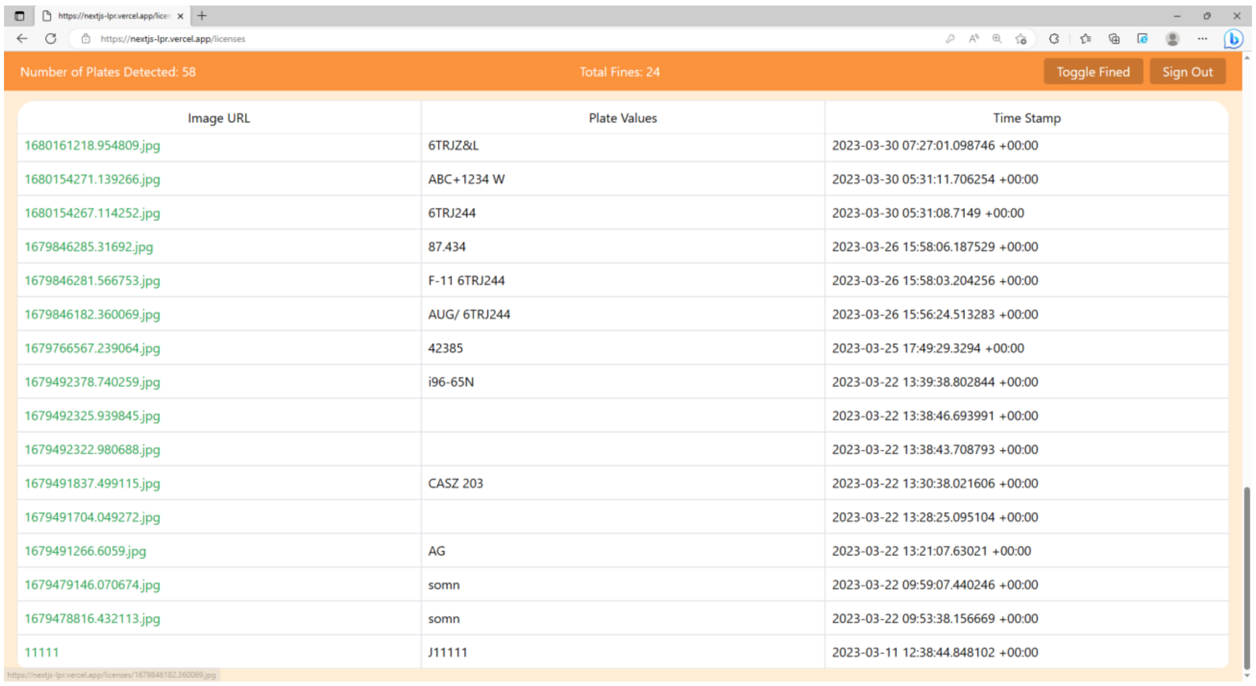


Welcome ll753@live.mdx.ac.uk

Number of Plates Detected: 58

Total Fines: 24

Image URL	Plate Values	Time Stamp
1681216450.725034.jpg	BHL 6276	2023-04-11 12:34:10.79864 +00:00
1681215597.105897.jpg		2023-04-11 12:19:57.221897 +00:00
1681214964.063702.jpg	IKAR BB01B	2023-04-11 12:09:24.152275 +00:00
1681214735.953182.jpg		2023-04-11 12:05:36.061707 +00:00
1680959048.676177.jpg	6TRJZ4L	2023-04-08 13:04:08.774347 +00:00
1680958782.137655.jpg	STRJZL0	2023-04-08 12:59:42.215542 +00:00
1680958304.010956.jpg	auo Califo 6TR	2023-04-08 12:51:44.12604 +00:00
1680956296.603454.jpg	6TRJZLL	2023-04-08 12:18:16.696344 +00:00
1680956157.804198.jpg	X7T AUG 6TRJ244	2023-04-08 12:15:57.97093 +00:00
1680956059.890569.jpg		2023-04-08 12:14:19.986505 +00:00
1680681668.512223.jpg	EN	2023-04-05 08:01:08.660004 +00:00
1680681667.663392.jpg	MX iZ3	2023-04-05 08:01:07.813046 +00:00
1680681667.051623.jpg	AA01434	2023-04-05 08:01:07.166571 +00:00
1680681660.911766.jpg		2023-04-05 08:01:01.056897 +00:00
1680681660.361206.jpg		2023-04-05 08:01:00.493801 +00:00



Number of Plates Detected: 58

Total Fines: 24

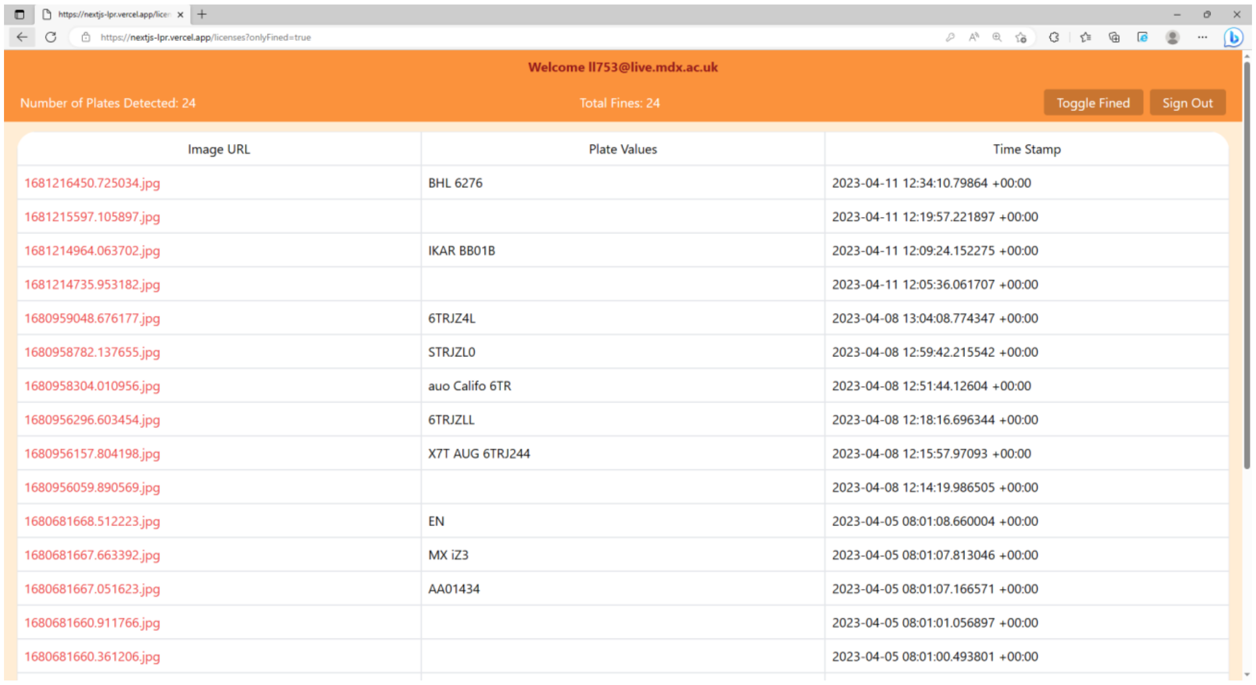
Image URL	Plate Values	Time Stamp
1680161218.954809.jpg	6TRJZ&L	2023-03-30 07:27:01.098746 +00:00
1680154271.139266.jpg	ABC+1234 W	2023-03-30 05:31:11.706254 +00:00
1680154267.114252.jpg	6TRJ244	2023-03-30 05:31:08.7149 +00:00
1679846285.31692.jpg	87.434	2023-03-26 15:58:06.187529 +00:00
1679846281.566753.jpg	F-11 6TRJ244	2023-03-26 15:58:03.204256 +00:00
1679846182.360069.jpg	AUG/ 6TRJ244	2023-03-26 15:56:24.513283 +00:00
1679766567.239064.jpg	42385	2023-03-25 17:49:29.3294 +00:00
1679492378.740259.jpg	i96-65N	2023-03-22 13:39:38.802844 +00:00
1679492325.939845.jpg		2023-03-22 13:38:46.693991 +00:00
1679492322.980688.jpg		2023-03-22 13:38:43.708793 +00:00
1679491837.499115.jpg	CASZ 203	2023-03-22 13:30:38.021606 +00:00
1679491704.049272.jpg		2023-03-22 13:28:25.095104 +00:00
1679491266.6059.jpg	AG	2023-03-22 13:21:07.63021 +00:00
1679479146.070674.jpg	somn	2023-03-22 09:59:07.440246 +00:00
1679478816.432113.jpg	somn	2023-03-22 09:53:38.156669 +00:00
11111	J11111	2023-03-11 12:38:44.848102 +00:00

https://nextjs-lpr.vercel.app/licenses/1679846182.360069.jpg

The list is scrollable and is ordered such that the most recent plates are displayed first. The list/table displays information like the image URL, the OCR plate value and the time stamp of when the license plate was captured. The user can click on the image URL to be redirected to a page with in-depth information about the captured license plate. The image URLs are coloured green or red based on the fine status, where red indicates that a fine has been imposed and the green indicates no fine.

The license plate page has two button, one for sign out and the other is a 'Toggle Fined' button. The 'Toggle Fined' works such that, if the button is clicked the list will display only those license plate entries which have been fined.

View displays only fined plates

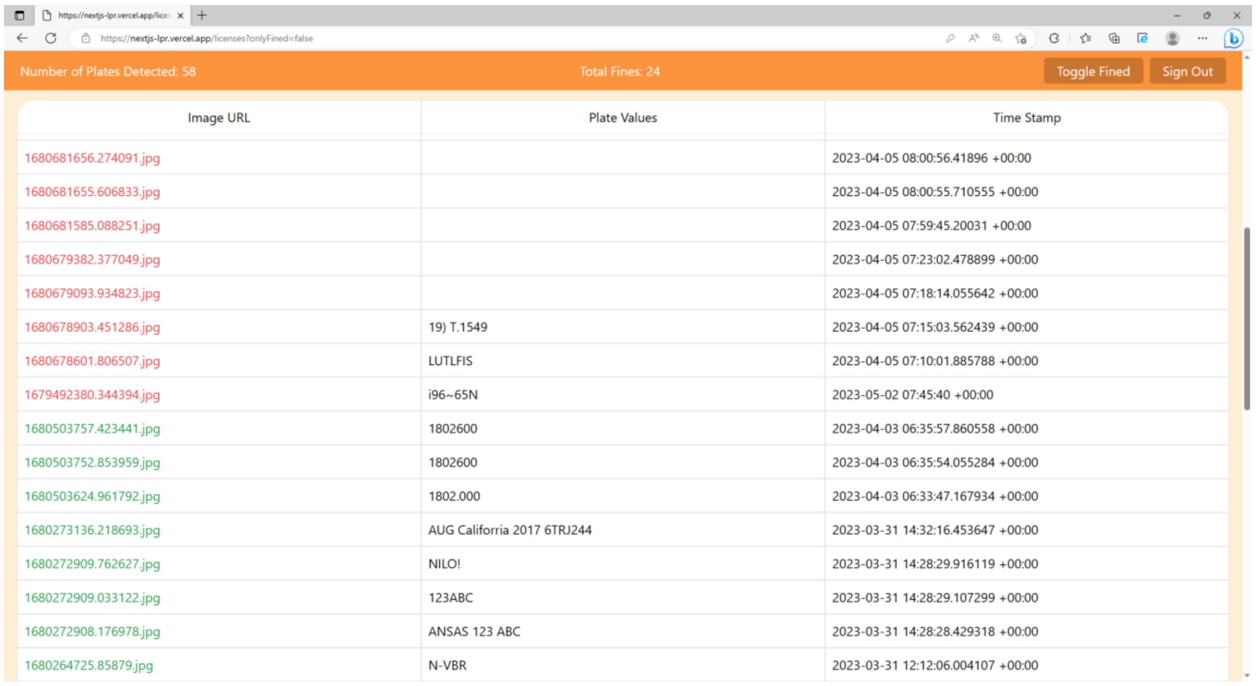


Welcome ll753@live.mdx.ac.uk

Number of Plates Detected: 24   Total Fines: 24   Toggle Fined   Sign Out

Image URL	Plate Values	Time Stamp
1681216450.725034.jpg	BHL 6276	2023-04-11 12:34:10.79864 +00:00
1681215597.105897.jpg		2023-04-11 12:19:57.221897 +00:00
1681214964.063702.jpg	IKAR BB01B	2023-04-11 12:09:24.152275 +00:00
1681214735.953182.jpg		2023-04-11 12:05:36.061707 +00:00
1680959048.676177.jpg	6TRJZ4L	2023-04-08 13:04:08.774347 +00:00
1680958782.137655.jpg	STRJZL0	2023-04-08 12:59:42.215542 +00:00
1680958304.010956.jpg	auo Califo 6TR	2023-04-08 12:51:44.12604 +00:00
1680956296.603454.jpg	6TRJZLL	2023-04-08 12:18:16.696344 +00:00
1680956157.804198.jpg	X7T AUG 6TRJ244	2023-04-08 12:15:57.97093 +00:00
1680956059.890569.jpg		2023-04-08 12:14:19.986505 +00:00
1680681668.512223.jpg	EN	2023-04-05 08:01:08.660004 +00:00
1680681667.663392.jpg	MX iZ3	2023-04-05 08:01:07.813046 +00:00
1680681667.051623.jpg	AA01434	2023-04-05 08:01:07.166571 +00:00
1680681660.911766.jpg		2023-04-05 08:01:01.056897 +00:00
1680681660.361206.jpg		2023-04-05 08:01:00.493801 +00:00

View displays all plates



Number of Plates Detected: 58   Total Fines: 24   Toggle Fined   Sign Out

Image URL	Plate Values	Time Stamp
1680681656.274091.jpg		2023-04-05 08:00:56.41896 +00:00
1680681655.606833.jpg		2023-04-05 08:00:55.710555 +00:00
1680681585.088251.jpg		2023-04-05 07:59:45.20031 +00:00
1680679382.377049.jpg		2023-04-05 07:23:02.478899 +00:00
1680679093.934823.jpg		2023-04-05 07:18:14.055642 +00:00
1680678903.451286.jpg	19) T.1549	2023-04-05 07:15:03.562439 +00:00
1680678601.806507.jpg	LUTLFIS	2023-04-05 07:10:01.885788 +00:00
1679492380.344394.jpg	i96~65N	2023-05-02 07:45:40 +00:00
1680503757.423441.jpg	1802600	2023-04-03 06:35:57.860558 +00:00
1680503752.853959.jpg	1802600	2023-04-03 06:35:54.055284 +00:00
1680503624.961792.jpg	1802.000	2023-04-03 06:33:47.167934 +00:00
1680273136.218693.jpg	AUG Califorria 2017 6TRJ244	2023-03-31 14:32:16.453647 +00:00
1680272909.762627.jpg	NILO!	2023-03-31 14:28:29.916119 +00:00
1680272909.033122.jpg	123ABC	2023-03-31 14:28:29.107299 +00:00
1680272908.176978.jpg	ANSAS 123 ABC	2023-03-31 14:28:28.429318 +00:00
1680264725.85879.jpg	N-VBR	2023-03-31 12:12:06.004107 +00:00

The list of license plate have been pre-fetch at request execution time and are displayed on the screen by accessing each item in the list of the destructured prop using the map function.

License Plate Details Page

If a user wishes to view more information of a listed license plate. The user can click on the image URL and the information will be provided.

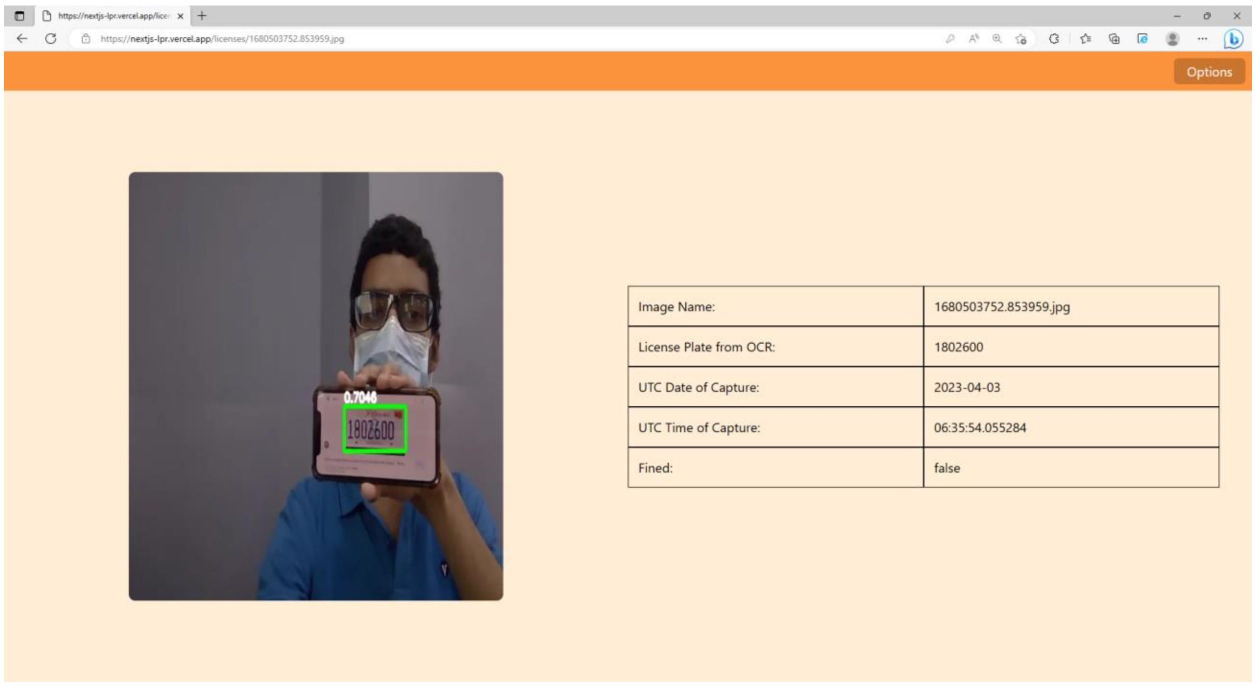

https://nextjs-lpr.vercel.app/licenses/1680503752.853959.jpg
Options
0.7046
1802600

Image Name:	1680503752.853959.jpg
License Plate from OCR:	1802600
UTC Date of Capture:	2023-04-03
UTC Time of Capture:	06:35:54.055284
Fined:	false

The page displays the image on the left hand side and the details on the right. The image also has the bounding box of the detected license plate and the confidence value of the detection. The table of details on the other hand displays information which has been extracted from the image.

The image is rendered at load time while the table of details have been pre-fetched at request execution time. Thus if the image has not been cached, the table of details will be displayed first and the image will be rendered after a few seconds.

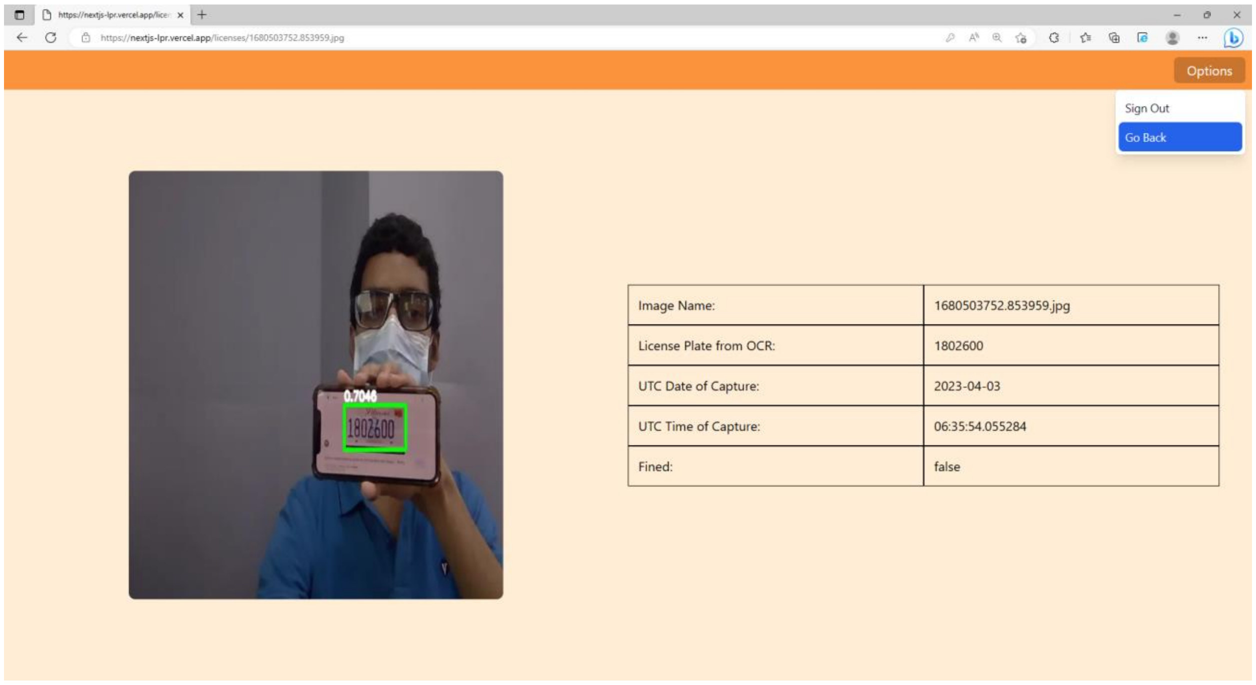
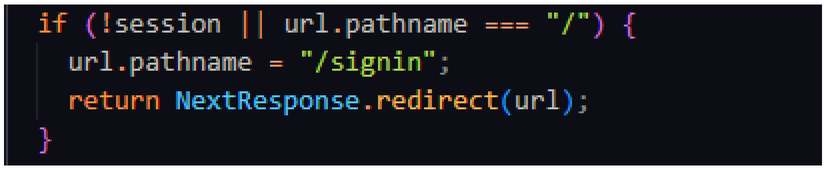

https://nextjs-lpr.vercel.app/licenses/1680503752.853959.jpg
Options
Sign Out
Go Back
0.7046
1802600

Image Name:	1680503752.853959.jpg
License Plate from OCR:	1802600
UTC Date of Capture:	2023-04-03
UTC Time of Capture:	06:35:54.055284
Fined:	false

The page also contains an ‘Options’ button which is a dropdown menu where the user can choose to either sign out or return back to the licenses page.

Middleware

Middleware is used in authentication, where if the user is invalid then the page redirects to the signin page.



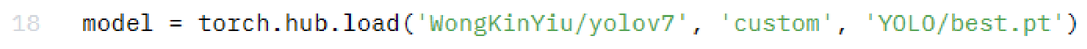
```
if (!session || url.pathname === "/") {
  url.pathname = "/signin";
  return NextResponse.redirect(url);
}
```

If a user session does not exist the page will redirect to the signin page, prompting the user to sign in.

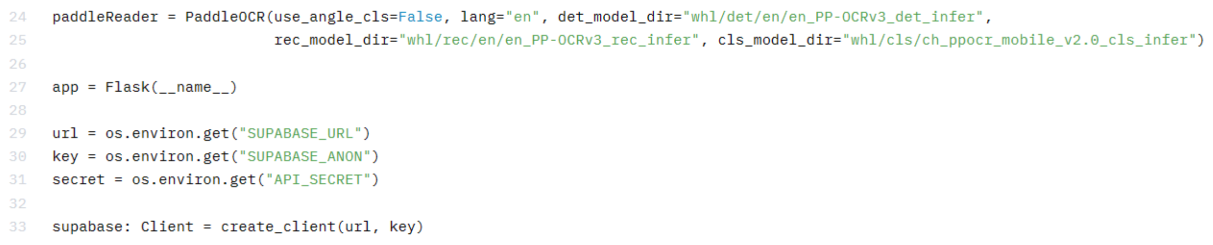
Flask Backend API

The Backend API is a hosted Python Script whose API is called by the Device.

In the script, first all necessary libraries are imported for use.



```
18   model = torch.hub.load('WongKinYiu/yolov7', 'custom', 'YOLO/best.pt')
```



```
24   paddleReader = PaddleOCR(use_angle_cls=False, lang="en", det_model_dir="whl/det/en/en_PP-OCRv3_det_infer",
25                            rec_model_dir="whl/rec/en/en_PP-OCRv3_rec_infer", cls_model_dir="whl/cls/ch_ppocr_mobile_v2.0_cls_infer")
26
27   app = Flask(__name__)
28
29   url = os.environ.get("SUPABASE_URL")
30   key = os.environ.get("SUPABASE_ANON")
31   secret = os.environ.get("API_SECRET")
32
33   supabase: Client = create_client(url, key)
```

In line 18, the YOLOv7 model is loaded using the torch.hub.load function provided by the Torch library in Python.

Line 24, is the initialisation of the OCR object. The object paddleReader will be used for OCR in the following script. The PaddleOCR class is linked to the paddleReader object and a few parameters have been passed to help in the OCR process.

In line 27, the Flask application has been initialised, this enables access to method like run and debug which is needed during development and to put the application into production.

Lines 29 to 31 are used to fetch the environment variables which will be used later in the script. Immediately in line 33 the Supabase Client is created for access to the Supabase database server.

The model used for complete detection was the YOLOv7 model using the PyTorch inference as this model was the most accurate in terms of license plate detection. This YOLOv7 model, unlike the rest, was even able to detect license plates that were tiny.

PaddleOCR was selected as the method for OCR in this implementation.

```
164        if request.method == 'POST': #change to POST
165            try:
166                fileimage = request.get_json()['image']
167                getSecret = request.get_json()['secret']
168                number = request.get_json()['number']
169
```

The first two verification come from checking the request method in line 164 and the try catch exception in line 165. If the request was not a POST request, the API will quit any further action and give a try again response. The try catch exception is used to check if all three required data, image, secret and minimum number, have been provided. If not the API wil quit any further action and give a try again response.

```
170                if secret == getSecret:
```

Finally in line 170, the received secret key and the stored secret will be compared. If they are not equivalent, the API will quit any further action and give a try again response.

After verification the received image, which is in the string or bytes format, will be converted into the ndarray format,

```
171                    photo = base64.b64decode(fileimage)
172                    np_data = np.frombuffer(photo, np.uint8)
173                    image = cv2.imdecode(np_data, cv2.IMREAD_UNCHANGED)
174                    image_np = np.array(image)
```

```
176                    results = model(image_np)
177                    detections = results.pandas().xyxy[0]
178                    if len(detections) == 0:
179                        return "No Detections"
180                    elif detections.iloc[0].confidence < 0.5:
181                        return "Detection confidence too low"
```

Next the converted frame date will be passed into the model for detection and based on the detection confidence the script will proceed further.

If the detection confidence is above the predefined threshold, the detections will be looped over one by one to draw the bounding box and run OCR on each detected license plate.

```
                        cropped = image_np[ymin:ymax, xmin:xmax, ...]
```

The image is cropped based on the bounding box values and then pre-processing is performed.

```
                        test_img = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
                        th3 = cv2.bilateralFilter(test_img, 11, 17, 17)
                        roi_image = cv2.threshold(th3, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
                        roi_image = clear_border(roi_image)
                        roi_image = cv2.cvtColor(roi_image, cv2.COLOR_BGR2RGB)
                        test_img1 = cv2.fastNlMeansDenoisingColored(roi_image, None, 20, 20, 7, 21)
                        test_img1 = cv2.fastNlMeansDenoisingColored(test_img1, None, 20, 7, 21)
```

In pre-processing, the cropped image is first converted into its grayscale format and a bilateral filter is applied. Next OTSU thresholding is performed to receive a binary image i.e. a black and white image of the license plate. The ‘clear_border’ function is then used remove any extra details that are away from the license plate characters. Finally the noise in the image is removed twice using different thresholds. After this the pre-processed image is passed into the PaddleOCR model for image-to-text conversion.

```
                        result = paddleReader.ocr(test_img1, cls=False)
```

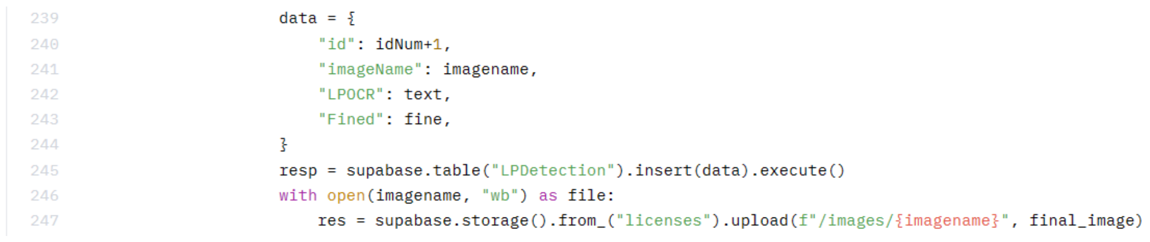
Next the extracted license plate number is compared with existing plate numbers in Supabase.

```
218                         idNum = list(supabase.table("LPDetails").select("*").eq('LP', text).execute())[0][1]
219                         try:
220                             thing = idNum[0]
221                             dt = datetime.datetime.now(timezone.utc)
222                             utc_time = dt.replace(tzinfo=timezone.utc)
223
224                             test = thing["PaymentTime"].replace("T", " ").replace("+00:00", "").split(".")[0]
225                             fullDate = test.split(" ")
226                             date = fullDate[0].split("-")
227                             actualDate = date[2]+"/"+date[1]+"/"+date[0][-2]+date[0][-1]+" "+fullDate[1]
228                             initial = datetime.datetime.strptime(actualDate, '%d/%m/%y %H:%M:%S')
229                             initial = initial.replace(tzinfo=timezone.utc)
230                             final = initial + datetime.timedelta(hours=thing["PaymentHours"])
231
232                             if final < utc_time:
233                                 fine = True
234                         except:
235                             fine = True
```

The extracted text is used to query the database. If the response list is empty, the script will jump to line 235 and immediately fine the car owner. This is because since the response list is empty that means that there is no information about the car in the parking payment database, which can only mean that the payment was not done.

Lines 221 till 230 is the logic used to convert the data from Supabase into a Python datetime format for easy comparison with the current time. The comparison is done in line 232, where if the final time or the end time of parking is lesser than the current time, the car owner must be fined.

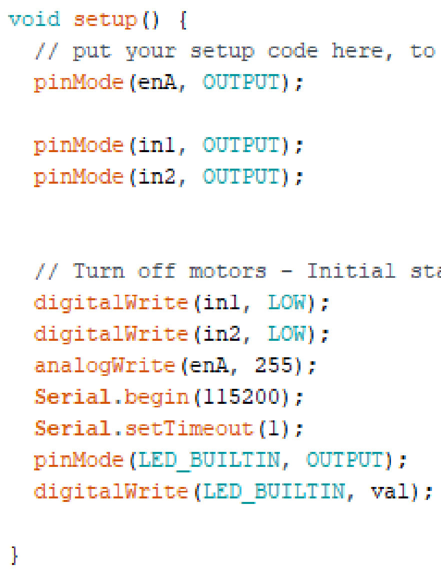
The extracted and analysed information is then stored on Supabase along with the full image with the bounding box of the detected license plate.



```python
                data = {
                    "id": idNum+1,
                    "imageName": imagename,
                    "LPOCR": text,
                    "Fined": fine,
                }
                resp = supabase.table("LPDetection").insert(data).execute()
                with open(imagename, "wb") as file:
                    res = supabase.storage().from_("licenses").upload(f"/images/{imagename}", final_image)
```

Arduino Code

The code running on the Arduino UNO is fairly simple.



```cpp
void setup() {
  // put your setup code here, to run once:
  pinMode(enA, OUTPUT);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);


  // Turn off motors - Initial state
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  Serial.begin(115200);
  Serial.setTimeout(1);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, val);

}
```

First all the pins and the Serial port is initialised in the setup function. All peripherals are set to be in the off state.

```
void loop() {

while (!Serial.available());
  num = Serial.readString().toInt();
  if (num == 1){
    val = 1;
    digitalWrite(LED_BUILTIN,val);
   analogWrite(enA, 255);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  }
  else if (num == 2){
    val = 0;
    digitalWrite(LED_BUILTIN,val);
    analogWrite(enA, 175);
    digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  }
  else if (num == 3){
    val = !val;
    digitalWrite(LED_BUILTIN,val);
    delay(250);
    val = !val;
    digitalWrite(LED_BUILTIN,val);
    delay(250);
    val = !val;
    digitalWrite(LED_BUILTIN,val);
    delay(250);
    val = !val;
    digitalWrite(LED_BUILTIN,val);
    delay(250);
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
  else{
    digitalWrite(LED_BUILTIN,0);
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
}
```

In the loop function, the values being sent to the Serial port by the Raspberry Pi are constantly read. Once a change has been detected the corresponding code is run.

Case1: the internal LED is switched on and the motor runs at full speed

Case2: the internal LED is switched off and the motor is slowed down

Case3: the internal LED blinks 2 times and the motor comes to a halt

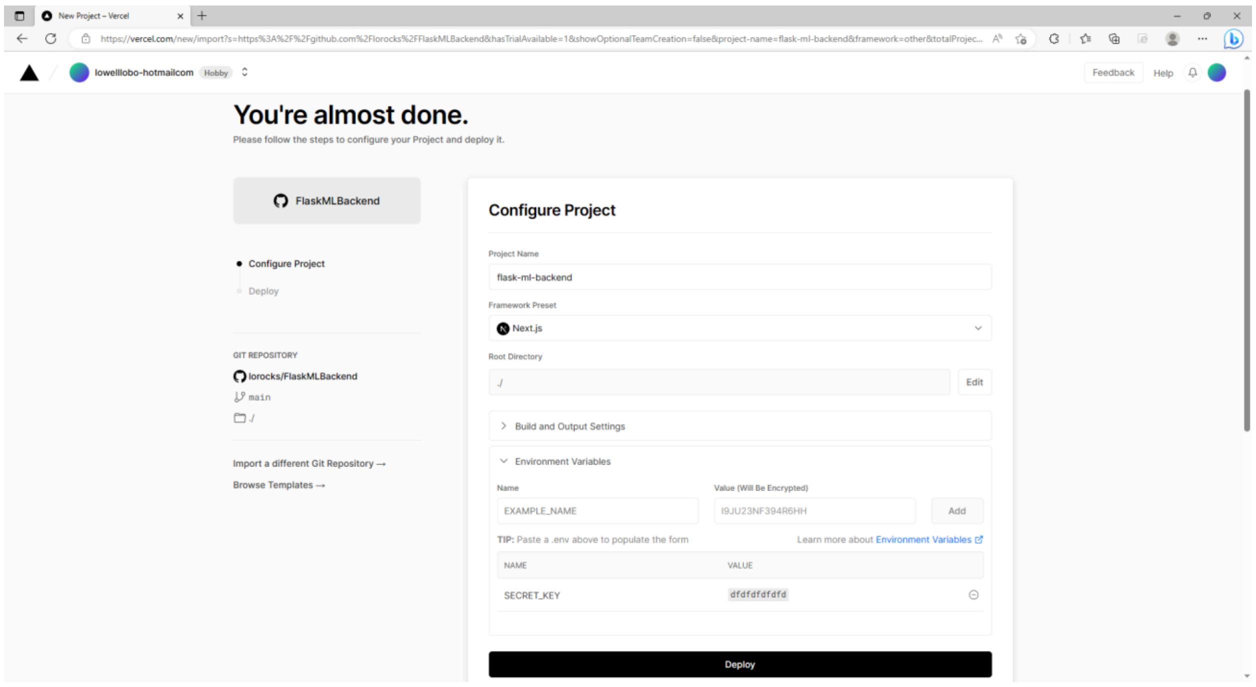
Case4: all peripheral are set to the off state

Hosting

Web Application Hosting

The Next.js application was hosted on Vercel, which is the Next.js recommended hosting platform. The process of hosting a Next.js application on Vercel is straightforward. The steps are,

- Create a Vercel Account
- Upload the Next.js application code to GitHub
- Add a new project in Vercel by clicking the ‘Add New’ button
- Link the Vercel account to GitHub
- Import the Next.js Repository from GitHub
- Select Next,js as the framework and add the Environment Variables
- Press Deploy



New Project – Vercel

https://vercel.com/new/import?s=https%3A%2F%2Fgithub.com%2Florocks%2FFlaskMLBackend&hasTrialAvailable=1&showOptionalTeamCreation=false&project-name=flask-ml-backend&framework=other&totalProjec...

lowelllobo-hotmailcom Hobby

You're almost done.

Please follow the steps to configure your Project and deploy it.

FlaskMLBackend

- Configure Project
- Deploy

GIT REPOSITORY
lorocks/FlaskMLBackend
main
./

Import a different Git Repository →
Browse Templates →

Configure Project

Project Name: flask-ml-backend

Framework Preset: Next.js

Root Directory: ./

Build and Output Settings

Environment Variables

Name: EXAMPLE_NAME
Value (Will Be Encrypted): I9JU23NF394R6HH

TIP: Paste a .env above to populate the form

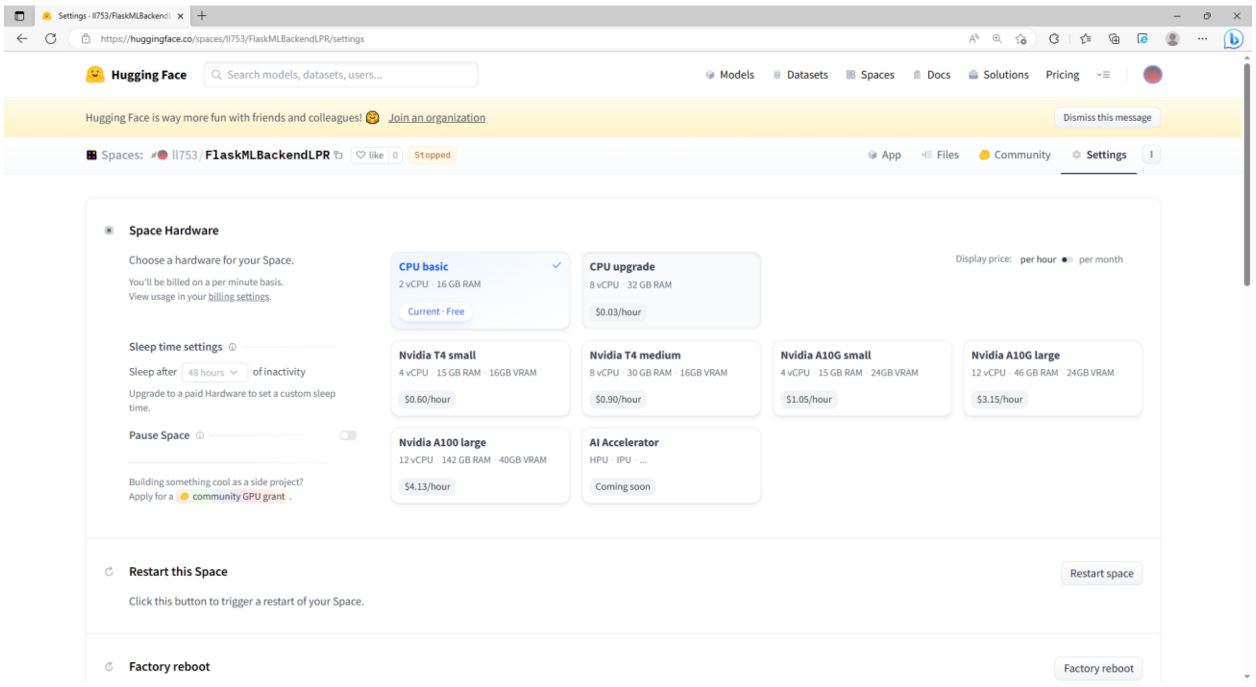
Learn more about Environment Variables

NAME	VALUE
SECRET_KEY	dfdfdfdfdfd

Vercel will automatically start deploying the application and will display live deployment updates. After the application has been deployed, Vercel will provide the URL to access the application.

ML Backend Hosting

The Flask application was hosted on Hugging Face Spaces, which has a large community focused on machine learning deployment. Hugging Face provides very good resources to deploy machine learning and deep learning application for free. At a low cost the resources can be upgraded to decrease inference time.


Settings - ll753/FlaskMLBackendL
https://huggingface.co/spaces/ll753/FlaskMLBackendLPR/settings

Hugging Face — Search models, datasets, users... — Models — Datasets — Spaces — Docs — Solutions — Pricing

Hugging Face is way more fun with friends and colleagues! Join an organization — Dismiss this message

Spaces: ll753/FlaskMLBackendLPR — like 0 — Stopped — App — Files — Community — Settings

Space Hardware

Choose a hardware for your Space.

You'll be billed on a per minute basis.
View usage in your billing settings.

Display price: per hour / per month

CPU basic 2 vCPU · 16 GB RAM Current · Free	CPU upgrade 8 vCPU · 32 GB RAM $0.03/hour		
Nvidia T4 small 4 vCPU · 15 GB RAM · 16GB VRAM $0.60/hour	Nvidia T4 medium 8 vCPU · 30 GB RAM · 16GB VRAM $0.90/hour	Nvidia A10G small 4 vCPU · 15 GB RAM · 24GB VRAM $1.05/hour	Nvidia A10G large 12 vCPU · 46 GB RAM · 24GB VRAM $3.15/hour
Nvidia A100 large 12 vCPU · 142 GB RAM · 40GB VRAM $4.13/hour	AI Accelerator HPU · IPU · ... Coming soon		

Sleep time settings

Sleep after 48 hours of inactivity

Upgrade to a paid Hardware to set a custom sleep time.

Pause Space

Building something cool as a side project?
Apply for a community GPU grant.

Restart this Space

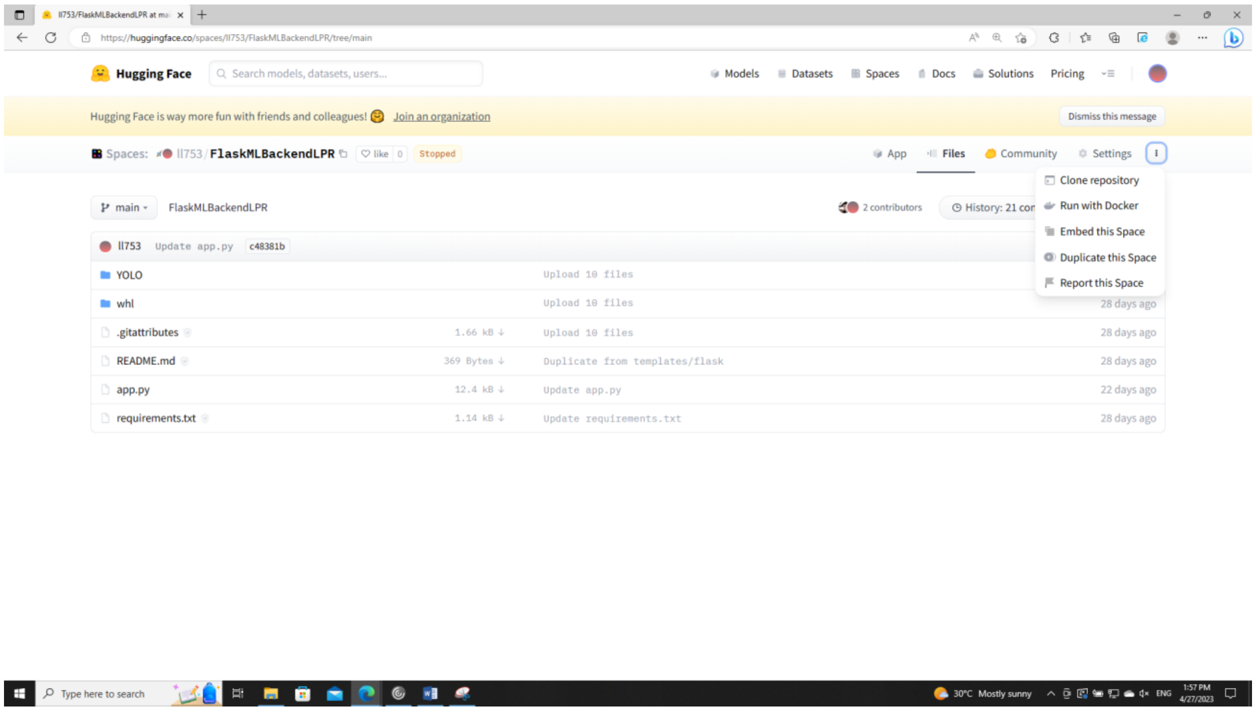
Click this button to trigger a restart of your Space. — Restart space

Factory reboot — Factory reboot

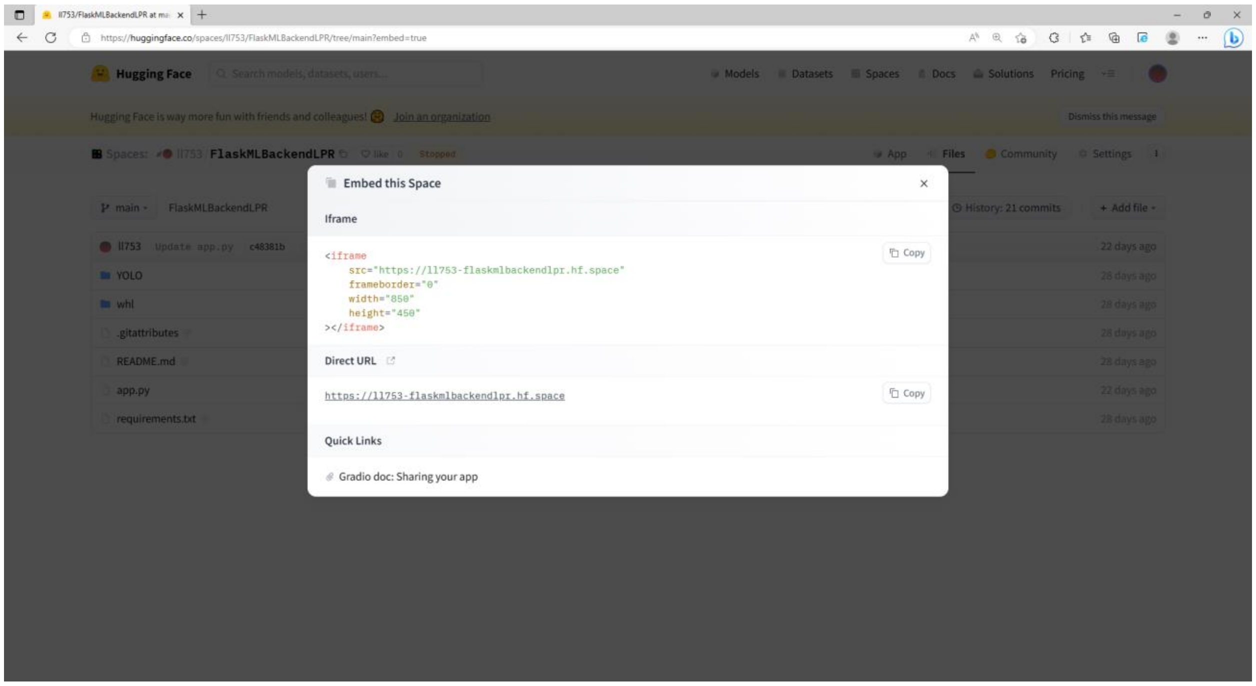
Hugging Face support various types of application deployments, but Flask development is not so straight forward. The best way to host a Flask application is to clone an existing Flask Demo application and modify it.

Hugging Face Spaces acts as a Repository where files can be uploaded. Hugging Face will automatically execute any Python script that is named ‘app.py’ which is stored in the Repository. The Flask script should be named as ‘app.py’ and uploaded on to Hugging Face Spaces. Immediately Hugging Face will start building the application.

Once the application has been built, the access URL can be found by click on the 3 dots beside the ‘Settings’ tab which will display a dropdown.



ll753 Update app.py c48381b			
YOLO		Upload 10 files	
whl		Upload 10 files	28 days ago
.gitattributes	1.66 kB	Upload 10 files	28 days ago
README.md	369 Bytes	Duplicate from templates/flask	28 days ago
app.py	12.4 kB	Update app.py	22 days ago
requirements.txt	1.14 kB	Update requirements.txt	28 days ago



https://huggingface.co/spaces/ll753/FlaskMLBackendLPR/tree/main?embed=true
Embed this Space
Iframe
Copy
Direct URL
Copy
Quick Links
Gradio doc: Sharing your app

```
<iframe
    src="https://ll753-flaskmlbackendlpr.hf.space"
    frameborder="0"
    width="850"
    height="450"
></iframe>
```

https://ll753-flaskmlbackendlpr.hf.space

Clicking on the ‘Embed this Space’ option will provide the URL for the Hugging Face Spaces hosted application.

GitHub

Most of the code was constantly uploaded to GitHub to keep a safe backup of the project progress. For installation of machine learning libraries on the Raspberry Pi, a GitHub repository which details the steps required in installation was created.

Recap of Project Flow

Main Project

- Raspberry Pi detects a License Plate
- Raspberry Pi motions the Arduino to perform some action based on License Plate detection and response from the Flask API
- Raspberry Pi script verifies if the License Plate is the full plate
- Raspberry Pi initiates a POST request to the Flask API
- Raspberry Pi sends the image frame via a JSON object in the POST request
- Flask API receives the POST request
- Flask API verifies if the request is valid or not
- Flask API detects the License Plate in the image frame
- Flask API crops and does pre-processing on the detected plate
- Flask API extracts the License Plate Number with OCR
- Flask API check for parking payment
- Flask API saves the image and inferred data to Supabase

User Interaction

- User goes to the Vercel hosted Next.js URL
- User enters their credentials for authentication
- User is redirected to the License Plate List page
- User views the list of all License Plates detected
- User can select to view only Fined License Plates
- User clicks on a License Plate image URL
- User is redirected to the License Plate Details page
- User views the extracted details of the license plate along with the image and inference
- User can Sign Out of the application

Testing and Analysis

Testing is a key aspects as part of any project development. A project that is released without testing cannot be considered as a project, similarly only a fully tested project is considered a complete project. Testing is essential to ensure the quality, verify the quality and identify errors in a project.

Testing Methods

Deep Learning Models

The three trained models were tested on a Google Colabs notebook environment. The models were tested using a CPU, GPU and TPU. 150 images were used in the testing and comparison was done between accuracy, speed of inference and number of plates detected. The code for testing is provided as part of the libraries or can be found in the documentation. To run tests, the directory of the images needs to be linked and the training script needs to be executed. It is important to note that all the testing images are not present in the training images. On a smaller subset of the testing images, grayscaling was performed and results were compared with inference from normal images.

Microprocessor Selection

The Raspberry Pi 2 and Raspberry Pi 3B were initially selected and then compared with each other based on OpenCV FPS, FPS with model inference, ease of installing libraries and overall specifications.

Deep Learning Inference on the Raspberry Pi

Twelve models were run on the selected Raspberry Pi and comparison was done based on Raspberry Pi’s FPS and accuracy of each inference method. This was followed by testing the models based on distance of plate from the camera, speed of plate entering the frame, illumination of the surrounding area and size of the license plate. Another aspect that was compared was the frame lag due to OpenCV in each inference method.

Camera

The three different camera devices are compared while testing the Deep Learning Inference models to find the best camera device for this implementation. It is important to remember that based on the camera’s resolution the results from the Deep Learning Inference can also vary.

Raspberry Pi Inference Script Functionality

The functionality of each and every section of the inference script is verified and tested. Mainly, the correct working of the non-edge plate detection, Arduino communication, conversion of frame format for sending, API POST request and license plate locking to not send the same license plate through the Backend API have been tested.

Arduino Serial Connectivity with the Raspberry Pi

The response of the Arduino UNO to the commands given by the Raspberry Pi is verified here. Also the serial port connectivity and the Arduino’s interaction at booting time is considered.

Web Application

All the pages in the Next.js application have common tests and page-specific tests that need to be performed. Common tests include routing, rendering, pre-rendering and UI/UX tests. For page-specific tests, authentication methods and sign in verification needs to be looked at for the signin page. Field placement, toggling fines and correct fetching of data is assessed in the licenses page. Finally, image rendering, button actions and correct fetching of data is evaluated in the license plate details page. Also, authentication using Middleware was analysed.

Backend API

A wide variety of test were performed on the Flask API, these include, verification of the request, conversion of image to a format for inference, precise license plate inference, retrieval of correct bounding box coordinates, checking for valid parking, saving inferred data to Supabase and saving image to Supabase.

Pre-processing for OCR

As part of pre-processing, accurate cropping of license plate and comparison of the three different pre-processing methods was determined.

OCR Models

Overall four different model for OCR were looked at and comparison was drawn between each after performing image-to-text conversion on a non-pre-processed image and the three pre-processed images. Next, false prediction, no prediction and omission of characters after prediction was assessed for each model and compared.

Testing Results

Deep Learning Models

Processor Inference Testing

	MobileNet SSDv2	YOLOv7	YOLOv7 Tiny
CPU	1x speed	2.5x speed	1.2x speed
GPU	1x speed	2.4x speed	1.2x speed
TPU	1x speed	2.4x speed	0.9x speed

The speed displayed in the table are purely comparative for the row, thus 1x speed of CPU cannot be compared to the 1x speed on GPU.

It was noticed that if the CPU took a certain amount of time for inference, the GPU would be 3-4 times faster than the CPU and the TPU would be 4 times faster than the GPU.

Metrics Testing

	MobileNet SSDv2	YOLOv7	YOLOv7 Tiny
Accuracy	Moderate accuracy	Very good accuracy	Good accuracy
Speed	Very fast	Slow	Fast
Number of Plates	Least found	Highest found	Moderate found
Plates with Tilt Angle	Least found	Same as YOLOv7 Tiny	Same as YOLOv7

Comparing the models with each other, it can be seen that the YOLOv7 model is the best model but it is slow. The MobileNet SSDv2 model works well only if the plate are upright, thus although the speed is high, the accuracy is lacking. The YOLOv7 Tiny on the other hand is an average model which is able to accurately find license plates as long as they are not smaller than a certain index and inferences pretty fast.

Image Variation Testing

For image variation testing, two types of images were used, one with many license plates in the same image and another with only singular license plate.

MobileNet SSD – the number of license plates found and the bounding boxes were exactly similar

YOLOv7 – the bounding box region was similar, but the number of license plates found in the grayscaled image was less than the normal image

YOLOv7 Tiny – some license plate which were found in the normal image were not found in the grayscaled image, instead the plates that were not found on the normal image were found on the grayscaled image

Microprocessor Selection

	Raspberry Pi 2	Raspberry Pi 3B
OpenCV FPS	10 FPS	25 FPS
FPS with Inference	0.33 FPS	0.6 FPS
Ease of Library Installation	Very hard	Somewhat hard
Hardware Specifications	32 bit with 1 GB RAM	64 bit with 1 GB RAM

Overall it can be inferred that the Raspberry Pi 3B is better than the Raspberry Pi 2.

Deep Learning Inference on the Raspberry Pi

The accuracy is measure on a scale of 0 – 5, where 0 implies no plates detected while 5 implies very good accuracy.

FPS and Accuracy with Different Cameras

<u>Pi NoIR Camera Module v2</u>

	Normal Inference		Threads Inference	
	FPS	Accuracy	FPS	Accuracy
TensorFlow Saved Model	Crashes the Raspberry Pi		Crashes the Raspberry Pi	
TensorFlow Lite	1.4 FPS	2	1.4 FPS	2
YOLOv7 PyTorch	Crashes the Raspberry Pi		Crashes the Raspberry Pi	
YOLOv7 ONNX	0.7 FPS	3	0.35 FPS	3
YOLOv7 Tiny PyTorch	0.5 FPS	4	0.2 FPS	4
YOLOv7 Tiny ONNX	4.5 FPS	3	2 FPS	3

Webcam

	Normal Inference		Threads Inference	
	FPS	Accuracy	FPS	Accuracy
TensorFlow Saved Model	Crashes the Raspberry Pi		Crashes the Raspberry Pi	
TensorFlow Lite	1.4 FPS	2	1.4 FPS	2
YOLOv7 PyTorch	Crashes the Raspberry Pi		Crashes the Raspberry Pi	
YOLOv7 ONNX	0.7 FPS	3	0.65 FPS	3
YOLOv7 Tiny PyTorch	0.5 FPS	4	0.5 FPS	4
YOLOv7 Tiny ONNX	4 FPS	3	4 FPS	3

Environmental Factors

Speed

If the license plate enters and exits the frame at high speed the license plate is not detected because the plate that enters is too blurry. At extremely slow speed the models are able to easily detect plates. License plate can also be detected at a moderate speed but the actual speed values have not been noted down.

Medium size is 7cm x 4cm
Small size is 4cm x 2cm

Detections vary based on the tilt angle of the plate, but below listed are the max distances where detections were easily found.

TensorFlow Saved Model

Medium Size- 78cm
Small Size- 55cm
Medium Size with Low Illumination- 70cm

TensorFlow Lite:

Medium Size- 75cm
Small Size- 30cm
Medium Size with Low Illumination- 63cm

Torch YOLO:

Small Size- 75cm
Small Size with Low Illumination - 50cm

Here with bright illumination, a lot of license plates are found, but when there is low illumination not as many plates are found.

Torch YOLO Tiny:

Small Size- 70cm
Small Size with Dimmed Screen- 50cm

With bright illumination, a lot of license plates are found, but when there is low illumination not as many plates are found.

ONNX YOLO:

Medium Size- 110cm
Small Size- 25cm
Medium Size with Dimmed Screen- 80cm

ONNX YOLO Tiny:

Medium Size- 77cm
Small Size- 57cm
Medium Size with Dimmed Screen- 65cm
Small Size with Dimmed Screen- 57cm

OpenCV Lag

The frame lag is noticed in all normal inference methods except the YOLOv7 Tiny ONNX inference.
This might be because the YOLOv7 Tiny ONNX inference has a high enough FPS value.
The threads inference method is able to solve the OpenCV frame lag issue.

Camera

When comparing the different cameras with each other it can be found that,

- Best Resolution – Pi NoIR Camera Module v2
- Best Speed – Webcam
- Best Accuracy – Pi NoIR Camera Module v2

Raspberry Pi Inference Script Functionality

Test case	Expected outcome	Actual outcome	Pass/Fail
Arduino Communication	Send data to Arduino via Serial Port	Data is sent correctly	Pass
Non-Edge Plate Detection	Only license plate that are not near the edge of the screen should be sent to Backend	Works as expected	Pass
Converting Frame	Image frame should be converted from ndarray to string format	Image frame is converted to string format	Pass
License Locking	Should send a specific license plate to Backend only once	Works as expected	Pass
API POST Request	Should initiate a POST request	Works as expected	Pass

Arduino Serial Connectivity with the Raspberry Pi

The Arduino is able to connect to the Raspberry Pi and receives information in the Serial Port at baud rate 115200. The Arduino properly responds to commands send from the Raspberry Pi except at initial script execution time. During inference the Arduino works as intended, but during booting the first command is not executed properly. After code editing the bug was fixed and the Arduino now works as intended.

Web Application

Common Tests

Test case	Expected outcome	Actual outcome	Pass/fail
Page Routing	Should route the user to the intended page	Works as expected	Pass
Page Element Rendering	Should render all page elements	Works as expected	Pass
Pre-Rendering Data	Fetch data for rendering	Works as expected	Pass
UI/UX at First Glance	Should be easy to use and comprehend	The pages are designed with users in mind	Pass
UI at Different Zoom Levels	Should auto fit the element at different zoom levels	Works as expected as long as the zoom is below 300%	Pass for Computer Fail for Mobile

Page-Specific Tests

Signin Page

Test case	**Expected outcome**	**Actual outcome**	**Pass/fail**
Wrong Input Validation	If the email is not entered correctly, notify the user to enter valid email format	Works as expected	Pass
Invalid Credentials Validation	If the credentials are invalid, don't authenticate	User is notified that the authentication failed	Pass
Form Submission	The form should submit when user presses the 'Enter' key	Works as expected	Pass

Licenses List Page

Test case	**Expected outcome**	**Actual outcome**	**Pass/fail**
Field Placement	All page elements should be properly displayed with no overlap	Works as expected	Pass
Toggling Fines	Should toggle between displaying only fined plates and all plates when button is pressed	Works as expected	Pass

Fetch Data for List	Should fetch of all the license plate details	Works as expected	Pass
Fetch Data during Toggle Fines	Should fetch data based on the toggle state	Works sometimes and other times it fails	Fail
Fetch Data during Toggle Fines	Should fetch data based on the toggle state (after change in logic)	Works as expected	Pass

<u>License Plate Details Page</u>

Test case	**Expected outcome**	**Actual outcome**	**Pass/fail**
Image Rendering	Image should be rendered quickly and smoothly	Image takes too long to render and image pops up suddenly	Fail
Image Rendering using Next.js Image Element	Image should be rendered quickly and smoothly	Works as expected	Pass
Button Action	Buttons should work properly and redirect to correct location	Works as expected	Pass
Fetch Data	Should fetch the details of the selected license plate	Works as expected	Pass

Middleware

Middleware authentication works properly where users who are not authenticated are thus redirected to the signin page for authentication.

Backend API

Test case	Expected outcome	Actual outcome	Pass/fail
Verify POST Request	Allow only POST requests to be executed	Displays response "GET Request invalid"	Fail
Verify POST Request	Allow only POST requests to be executed (after code edited)	Allows only POST Request	Pass
Read JSON Data from Request	Read the data sent with the request	Data is read	Pass
Verify Secret	Grant access only if secret key from the request is correct	Secret key is identical to that in the API	Pass
Convert Image for Inference	Image should be converted from string to ndarray format	Doesn't convert	Fail
Convert Image for Inference	Image should be converted from string to ndarray format (after code edited)	String is converted to ndarray	Pass

Precise License Plate Detection	Should detect the license plate correctly	Works as expected	Pass
Retrieve Correct Bounding Box Values	Should be able to plot the bounding box correctly on the image	Works as expected	Pass
Check Payment Validity	The stored payment time must be compared with current time to check if fine should be levied	User is always fined because comparison is done with string and datetime format	Fail
Check Payment Validity after Conversion of String to Datetime	The stored payment time must be compared with current time to check if fine should be levied	Works as expected	Pass
Save Data to Supabase	The extracted data should be saved in Supabase	Shows id already exists error	Fail
Save Data to Supabase	The extracted data should be saved in Supabase (after querying the recent id and incrementing)	Works as expected	Pass
Save Image to Supabase	Should save the image correctly in Supabase	Image file is saved as a text file not an image file like ‘.jpg’	Fail

Save Image to Supabase	Should save the image correctly in Supabase (after code editing)	Image is saved as a bytes files not an image file like ‘.jpg’	Fail
Save Image to Supabase	Should save the image correctly in Supabase (after more code editing)	Works as expected	Pass

Pre-Processing for OCR

Cropping

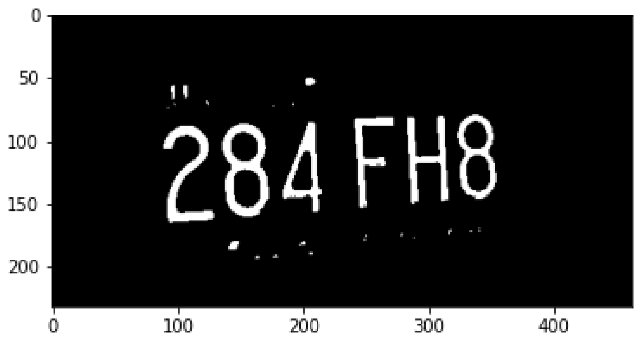
Cropping of the image is done based on the extracted bounding box values, but although the bounding box values were correct the image is not cropped properly. The error steps from different image sizes. When inference is performed the image is reformatted, but cropping was done on an unformatted image. After resizing the image, the cropping works as intended.

Pre-Processing Methods

Three different mathematical operations were implemented for pre-processing.

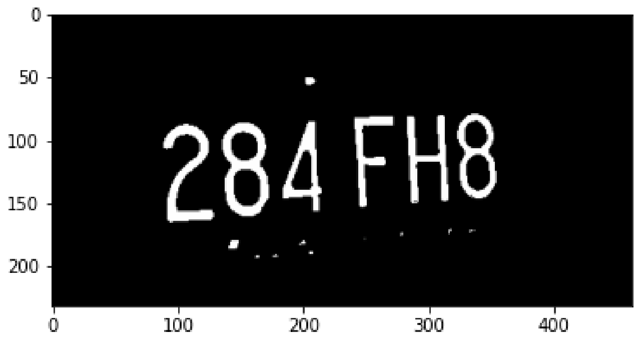
```
plt.imshow(license_image)
```

```
<matplotlib.image.AxesImage at 0x7f2e1645e7c0>
```



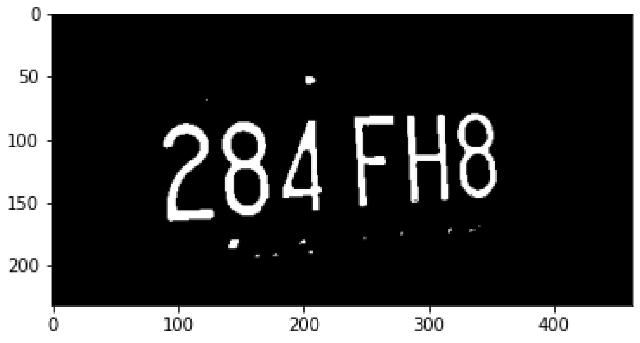
```
[60] plt.imshow(test_img1)
```

```
<matplotlib.image.AxesImage at 0x7f2e162c7ca0>
```



```
[45] plt.imshow(roi_image)
```

```
<matplotlib.image.AxesImage at 0x7f2e22094d00>
```



The third method gives the clearest image but the second method provides the least noise. Compared to the second and third, the first method produces a blurry image with lot of noise.

OCR Models

All four models were tested with the above mentioned pre-processing methods implemented.

	False Prediction	No Prediction	Omission of Characters	Speed of Inference
Tesseract	Always	Most of the time	N/A	N/A
Keras-OCR	Half the times	Never	Sometimes	Slow
EasyOCR	Sometimes	Never	Rarely	Fast
PaddleOCR	Sometimes but less than EasyOCR	Never	Rarely	Fast

Both EasyOCR and PaddleOCR, sometimes make false prediction because of the similarity between characters, for example { 5, S }. But EasyOCR is more prone to errors.

End-to-End Testing

Here the entire system's functionality and flow from start to end is tested. This is used to verify whether all project components harmoniously work together and meet the set requirements. The primary goal is to identify security flaws, performance issues and to verify if everything works after integration.

Post-Integration Testing

Test Method

<u>Device Inference</u>

In the test, the ML Activation script is booted and a license plate is placed in front of the camera. The license plate is slows slid into the frame and corresponding results are verified.

<u>Web Application</u>

The hosted web application is searched and email and password credentials are entered to authenticate. Next the most recent license plate entry is selected and the details are viewed.

Test Result

Device Inference

When the license plate is introduced in the frame, the motor speed reduces, but as long as the license plate is near the edge, nothing else happens. When the license plate is moved to the centre, the frame is send to the Backend for inference. After sometime, a response is printed on the terminal. The message, ‘Image has been saved’ is received from the Backend.

Web Application

Authentication happens properly and the page redirects to a page with a list of license plates. The top most license plate (most recent), is selected. The page redirect to another page, where the image which was captured is rendered and the OCR value from the Backend is displayed with other details.

Security Testing

Security check were made on the Web Application to ensure that the data cannot be accessed publically.

Performance Testing

The performance of the entire system was tested by using different license plate images and by varying the speed, distance and size of the license plate. The device was also tested to see if it would work properly over an extended duration of time.

Requirement Testing

Functional Requirements

- Device is able to capture images of cars with license plates
- Device is able to send the the images to a Backend Flask Server using an API call
- License plate detection and recognition is performed in the Backend with payment validation
- Information is stored in Supabase
- Information is displayed to the user using a Next.js App

Non-Functional Requirements

Security

- The Supabase is securely hosted with RLS enables
- The Next.js Application is hosted on a HTTPS and only authenticated users have access to data
- The Backend API cannot be used by anyone because of secret key validation

Reliability

- The device is able to detect license plates in varying environmental conditions
- The Backend API is able to perform better detections because of good resouce availability
- The payment validation for parking works as intended
- The information from Supabase is accurately displayed in the web application

Performance

- The Backend API is able to perform inference pretty fast because of good resouce availability
- The web application renders quickly and doesn't have excessive page flicker
- The ML Activation works with high FPS

Portability

- The ML Activation was tested on a Raspberry Pi 2 and a Raspberry Pi 3B. The script worked in both microprocessors.

Compatability

- The hosted web application can be accessed from any web browser
- The Backend API is able to detect license plate in varying environmental conditions

Serviceability

- The web application can be easily scaled and maintained because of technology provided by Vercel
- The Backend API can be easily scaled and maintained because of resources provided by Hugging Face Spaces
- The device's libraries can be easily updated and changes implemented by connecting to the Raspberry Pi via SSH

Conclusion

In conclusion, the project was successfully built and it met all set criteria. The project's main aim of creating a system to validate a car's parking payment was fulfilled along with other contributions. A systematic review about the existing license plate detection technologies and methods was drafted and can be used as a starting point by any researcher new to the field. Comparison between deep learning models, MobileNet SSDv2, YOLOv7 and YOLOv7 Tiny was performed. This is followed by implementing each model on a resource constrained device like a Raspberry Pi. Next, comparison of four different OCR libraries for image-to-text conversion is performed and all of this was implemented on a hosted Flask API. All the captured and extracted data is stored online and can be viewed by an authorised user using a custom built Next.js app.

Contribution

A newer deep learning models, YOLOv7, has been implemented in the design and result show that the model quite good and can be used in object detection scenarios. The YOLOv7 model was released quite a time before but not many researchers have used it in any application.

The YOLOv7 model was then used or inference on a resource constrained device like the Raspberry Pi and a method for inference using the ONNX Runtime is put forward to increase FPS and reduce strain on the Raspberry Pi.

PaddleOCR is a method for image-to-text conversion which has been hidden from the eyes of researchers. PaddleOCR gives superb released when compared with three other OCR libraries and is implemented as part of this system.

The images stored after capturing can be used for retraining, which makes this system a good method of collecting annotated images to build machine learning datasets.

A GitHub Repository was build to ease the process of library installation on any 64 bit Raspberry Pi system that runs Python version 3.9.

Finally, a robotic system is proposed and shown to be possible by implementation of a motor that responds according to the current state of the detection. The Arduino experimental motor shows that it is possible to build an autonomous robotic system which will exhibit motions based on data received from the Raspberry Pi. It is not highly necessary for the robotic device to be fully governed by the Raspberry Pi as even a line sensing robot can be implemented in this scenario.

Limitations

The Raspberry Pi is a resource constrained device and contains only 1 GB RAM. This restricts the device to be able to only run light weight models and even then exhibits low FPS.

The OCR model implemented has a low, but not zero, chance of false detection. This occurs because of image blur induced while cropping the image and from similar looking characters,

The OCR model also is unable to detect the license plate number if the image is too blurry. This can happen when the car or device is moving too fast and at times when the illumination is extremely low.

The Hugging Face hosted Flask API is quite fast using the free tier, but can be much faster using the paid tiers which provide higher RAM utilisation and GPU utilisation.

Future Scope

Coral Edge TPU is a machine learning ASIC that is used to accelerate deep learning inference on edge devices. This is just one of the many commercially available products that can boost FPS on resource constrained devices.

Instead of the Raspberry Pi, a NVIDIA Jetson could be used as a microprocessor. The NVIDIA Jetson provides usage of CUDA for inferencing which can be used to reduce inference times.

More images could be used in training to help boost accuracy of the deep learning model.

Also, instead of using an OCR library, the OCR model can be custom built for the specific scenario and a better camera can be used reduce errors from OCR.

Finally, a robotic device could be built to autonomously capture license plate images and surveil a set area.

References

Al-Hmouz, R. and Aboura, K. (2014) ‘License plate localization using a statistical analysis of Discrete Fourier Transform signal’, *Computers & Electrical Engineering*, 40(3), pp. 982–992. Available at: https://doi.org/10.1016/j.compeleceng.2014.01.001.

Ali, S.T.A., Usama, A.H., Khan, I.R., Khan, M.M. and Siddiq, A. (2021) ‘Mobile Registration Number Plate Recognition Using Artificial Intelligence’, in *2021 IEEE International Conference on Image Processing (ICIP). 2021 IEEE International Conference on Image Processing (ICIP)*, Anchorage, AK, USA: IEEE, pp. 944–948. Available at: https://doi.org/10.1109/ICIP42928.2021.9506699.

Aljelawy, Q.M. and Salman, T.M. (2022) ‘Detecting License Plate Number Using OCR Technique and Raspberry Pi 4 With Camera’, in *2022 2nd International Conference on Computing and Machine Intelligence (ICMI). 2022 2nd International Conference on Computing and Machine Intelligence (ICMI)*, Istanbul, Turkey: IEEE, pp. 1–5. Available at: https://doi.org/10.1109/ICMI55296.2022.9873776.

Al-Shemarry, M.S. and Li, Y. (2020) ‘Developing Learning-Based Preprocessing Methods for Detecting Complicated Vehicle Licence Plates’, *IEEE Access*, 8, pp. 170951–170966. Available at: https://doi.org/10.1109/ACCESS.2020.3024625.

Al-Shemarry, M.S., Li, Y. and Abdulla, S. (2018) ‘Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images’, *Expert Systems with Applications*, 92, pp. 216–235. Available at: https://doi.org/10.1016/j.eswa.2017.09.036.

Al-Shemarry, M.S., Li, Y. and Abdulla, S. (2023) ‘Identifying License Plates in Distorted Vehicle Images: Detecting Distorted Vehicle Licence Plates Using a Novel Preprocessing Methods With Hybrid Feature Descriptors’, *IEEE Intelligent Transportation Systems Magazine*, 15(2), pp. 6–25. Available at: https://doi.org/10.1109/MITS.2022.3210226.

Ashrafee, A., Khan, A.M., Irbaz, M.S. and Nasim, M.A.A. (2022) ‘Real-time Bangla License Plate Recognition System for Low Resource Video-based Applications’, in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW). 2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Waikoloa, HI, USA: IEEE, pp. 479–488. Available at: https://doi.org/10.1109/WACVW54805.2022.00054.

Asif, M.R., Qi, C., Wang, T., Fareed, M.S. and Raza, S.A. (2019) 'License plate detection for multi-national vehicles: An illumination invariant approach in multi-lane environment', *Computers & Electrical Engineering*, 78, pp. 132–147. Available at: https://doi.org/10.1016/j.compeleceng.2019.07.012.

Awan, S.M., Khattak, S., Khan, G.Z. and Mahmood, Z. (2019) 'A Robust Method to Locate License Plates under Diverse Conditions', in *2019 International Conference on Applied and Engineering Mathematics (ICAEM). 2019 International Conference on Applied and Engineering Mathematics (ICAEM)*, Taxila, Pakistan: IEEE, pp. 92–98. Available at: https://doi.org/10.1109/ICAEM.2019.8853773.

Bhagat, T. and Thakur, R. (2021) 'Automatic Recognition of License Plates', in *2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI). 2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, Hyderabad, India: IEEE, pp. 118–122. Available at: https://doi.org/10.1109/ICETCI51973.2021.9574072.

Bzdok, D., Krzywinski, M. and Altman, N. (2017) 'Machine learning: a primer', *Nature Methods*, 14(12), pp. 1119–1120. Available at: https://doi.org/10.1038/nmeth.4526.

Chao Gou, Wang, K., Li, B., and Fei-yue Wang (2014) 'Vehicle license plate recognition based on class-specific ERs and SaE-ELM', in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC). 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, China: IEEE, pp. 2956–2961. Available at: https://doi.org/10.1109/ITSC.2014.6958164.

Das, S. and Kumari, R. (2021) 'Application of Extended Hough Transform Technique for Stationary Images in Vehicle License Plate', in *2021 6th International Conference for Convergence in Technology (I2CT). 2021 6th International Conference for Convergence in Technology (I2CT)*, Maharashtra, India: IEEE, pp. 1–4. Available at: https://doi.org/10.1109/I2CT51068.2021.9417944.

Dhar, P., Abedin, Md.Z., Karim, R., Fatema-Tuj-Johora and Hossain, M.S. (2019) 'Bangladeshi License Plate Recognition Using Adaboost Classifier', in *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR). 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Spokane, WA, USA: IEEE, pp. 342–347. Available at: https://doi.org/10.1109/ICIEV.2019.8858580.

Domingos, P. (2012) ‘A few useful things to know about machine learning’, *Communications of the ACM*, 55(10), pp. 78–87. Available at: https://doi.org/10.1145/2347736.2347755.

Dwyer, B., Nelson, J. and Solawetz, J. (2022) ‘Roboflow (Version 1.0)’. Available at: https://roboflow.com/.

Elhadi, Y., Abdalshakour, O. and Babiker, S. (2019) ‘Arabic-Numbers Recognition System for Car Plates’, in *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Khartoum, Sudan: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICCCEEE46830.2019.9071288.

El-Latief, A.A., Hemayed, E.E., Helal, D. and Rehan, M. (2017) ‘Automatic Multi-Style License Plate Detection Using a Biologically Inspired Classifier’, in *2017 13TH INTERNATIONAL COMPUTER ENGINEERING CONFERENCE (ICENCO)*. Cairo Univ,Faculty Eng, Comp Eng Dept, pp. 67–72.

Gosthipaty, A. (2021) *License Plates*. Available at: https://www.kaggle.com/datasets/aritrag/license.

Gou, C., Wang, K., Yu, Z. and Xie, H. (2014) ‘License plate recognition using MSER and HOG based on ELM’, in *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics. 2014 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Qingdao, China: IEEE, pp. 217–221. Available at: https://doi.org/10.1109/SOLI.2014.6960724.

Haddaway, N.R., Page, M.J., Pritchard, C.C. and McGuinness, L.A. (2022) ‘PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis’, *Campbell Systematic Reviews*, 18(2), p. e1230. Available at: https://doi.org/10.1002/cl2.1230.

Henry, C., Ahn, S.Y. and Lee, S.-W. (2020) ‘Multinational License Plate Recognition Using Generalized Character Sequence Detection’, *IEEE Access*, 8, pp. 35185–35199. Available at: https://doi.org/10.1109/ACCESS.2020.2974973.

Hsieh, C.-F., Lin, C.-Z., Li, Z.-Z. and Cho, C.-H. (2022) ‘Automatic Vehicle License Plate Recognition Based on YOLO v4 for Smart Parking Management System’, in *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE). 2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*, Osaka, Japan: IEEE, pp. 905–906. Available at: https://doi.org/10.1109/GCCE56475.2022.10014165.

ibaiGorordo (2023) *ONNX YOLOv7 Object Detection*, *ibaiGorordo/ONNX-YOLOv7-Object-Detection*. Available at: https://github.com/ibaiGorordo/ONNX-YOLOv7-Object-Detection.

Kassm, G.A. and Achkar, R. (2017) 'LPR CNN Cascade and Adaptive Deskewing', *Procedia Computer Science*, 114, pp. 296–303. Available at: https://doi.org/10.1016/j.procs.2017.09.043.

Kessentini, Y., Besbes, M.D., Ammar, S. and Chabbouh, A. (2019) 'A two-stage deep neural network for multi-norm license plate detection and recognition', *Expert Systems with Applications*, 136, pp. 159–170. Available at: https://doi.org/10.1016/j.eswa.2019.06.036.

Kim, P., Lim, K.-T. and Kim, D. (2019) 'Learning Based Character Segmentation Method for Various License Plates', in *2019 16th International Conference on Machine Vision Applications (MVA)*. *2019 16th International Conference on Machine Vision Applications (MVA)*, Tokyo, Japan: IEEE, pp. 1–6. Available at: https://doi.org/10.23919/MVA.2019.8757905.

Kumar, G., Barman, A. and Pal, M. (2019) 'License Plate Tracking using Gradient based Segmentation', in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*. *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Kochi, India: IEEE, pp. 1737–1740. Available at: https://doi.org/10.1109/TENCON.2019.8929688.

L. Qin, W. Wei, and Z. Ma (2021) 'License Plate Image Recognition Based on Deep Learning under Complex Conditions', in *ICMLCA 2021; 2nd International Conference on Machine Learning and Computer Application*. *ICMLCA 2021; 2nd International Conference on Machine Learning and Computer Application*, pp. 1–4.

Larxel (2020) *Car License Plate Detection*. Available at: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection.

Li, Y., Niu, D., Chen, X., Li, T., Li, Q. and Xue, Y. (2019) 'Vehicle License Plate Recognition Combing MSER and Support Vector Machine in A Complex Environment', in *2019 Chinese Control Conference (CCC)*. *2019 Chinese Control Conference (CCC)*, Guangzhou, China: IEEE, pp. 7045–7050. Available at: https://doi.org/10.23919/ChiCC.2019.8865171.

Lien, C.-C., Chien, Y.-C., Teng, F.-Y. and Yang, C.-C. (2019) 'Deep License Plate Recognition in Ill-Conditioned Environments With Ill-Conditional Data Augmentation', in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*. *2019 International Conference on Machine Learning* *and Cybernetics (ICMLC)*, Kobe, Japan: IEEE, pp. 1–7. Available at: https://doi.org/10.1109/ICMLC48188.2019.8949248.

Liu, W.-C. and Lin, C.-H. (2017) ‘A hierarchical license plate recognition system using supervised K-means and Support Vector Machine’, in *2017 International Conference on Applied System Innovation (ICASI). 2017 International Conference on Applied System Innovation (ICASI)*, Sapporo, Japan: IEEE, pp. 1622–1625. Available at: https://doi.org/10.1109/ICASI.2017.7988244.

Lobo, L. (2023) *ML-Installation-for-RPi-64bit-3.9-Python, lorocks/ML-Installation-for-RPi-64bit-3.9-Python*. Available at: https://github.com/lorocks/ML-Installation-for-RPi-64bit-3.9-Python.

Lubna, Mufti, N. and Shah, S.A.A. (2021) ‘Automatic Number Plate Recognition:A Detailed Survey of Relevant Algorithms’, *Sensors*, 21(9), p. 3028. Available at: https://doi.org/10.3390/s21093028.

Lyasheva, M.M., Lyasheva, S.A. and Shleymovich, M.P. (2021) ‘Application of the Weight Model to Detect the State Registration Number of the Vehicle in the Image’, in *2021 International Russian Automation Conference (RusAutoCon). 2021 International Russian Automation Conference (RusAutoCon)*, Sochi, Russian Federation: IEEE, pp. 448–452. Available at: https://doi.org/10.1109/RusAutoCon52004.2021.9537382.

M. S. Al-Shemarry, Y. Li, and S. Abdulla (2022) ‘Detecting Distorted Vehicle Licence Plates Using Novel Preprocessing Methods With Hybrid Feature Descriptors’, *IEEE Intelligent Transportation Systems Magazine*, pp. 2–21. Available at: https://doi.org/10.1109/MITS.2022.3210226.

Maduri, P.K., Dhiman, P., Rathour, S., Dutt, E. and Gupta, A. (2021) ‘Smart Eye of Traffic Management & Control System’, in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India: IEEE, pp. 794–798. Available at: https://doi.org/10.1109/ICAC3N53548.2021.9725593.

Min, W., Li, X., Wang, Q., Zeng, Q. and Liao, Y. (2019) ‘New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification’, *IET IMAGE PROCESSING*, 13(7), pp. 1041–1049. Available at: https://doi.org/10.1049/iet-ipr.2018.6449.

N. Ramya, M. Annamalai, and K. Santhosh (2022) ‘An Automated Vehicle Tracking System Using Haar-Cascade Classifiers and Optical Character Recognition Engine’, in *2022 1st International Conference on*

Computational Science and Technology (ICCST). 2022 1st International Conference on Computational Science and Technology (ICCST), pp. 958–962. Available at: https://doi.org/10.1109/ICCST55948.2022.10040346.

Neto, E.C., Gomes, S.L., Rebouças Filho, P.P. and de Albuquerque, V.H.C. (2015) 'Brazilian vehicle identification using a new embedded plate recognition system', *Measurement,* 70, pp. 36–46. Available at: https://doi.org/10.1016/j.measurement.2015.03.039.

NEXT.js (2022) *Middleware*. Available at: https://nextjs.org/docs/advanced-features/middleware.

Nguyen, D.-L., Putro, M.D., Vo, X.-T. and Jo, K.-H. (2021) 'Triple Detector based on Feature Pyramid Network for License Plate Detection and Recognition System in Unusual Conditions', in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE). 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE),* Kyoto, Japan: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ISIE45552.2021.9576487.

Nguyen, T.-N. and Nguyen, D.-D. (2018) 'A New Convolutional Architecture for Vietnamese Car Plate Recognition', in *2018 10th International Conference on Knowledge and Systems Engineering (KSE). 2018 10th International Conference on Knowledge and Systems Engineering (KSE),* Ho Chi Minh City, Vietnam: IEEE, pp. 7–12. Available at: https://doi.org/10.1109/KSE.2018.8573375.

Onim, Md.S.H., Nyeem, H., Roy, K., Hasan, M., Ishmam, A., Akif, Md.A.H. and Ovi, T.B. (2022) 'BLPnet: A new DNN model and Bengali OCR engine for Automatic Licence Plate Recognition', *Array*, 15, p. 100244. Available at: https://doi.org/10.1016/j.array.2022.100244.

Polishetty, R., Roopaei, M. and Rad, P. (2016) 'A Next-Generation Secure Cloud-Based Deep Learning License Plate Recognition for Smart Cities', in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA),* Anaheim, CA, USA: IEEE, pp. 286–293. Available at: https://doi.org/10.1109/ICMLA.2016.0054.

Pourhadi, N., Shafizadeh, B., Eshghi, F. and Kelarestaghi, M. (2022) 'YOLOv5-based ALPR Improvement using Selective SR-GAN', in *2022 2nd International Conference on Computing and Machine Intelligence (ICMI). 2022 2nd International Conference on Computing and Machine Intelligence (ICMI),* Istanbul, Turkey: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICMI55296.2022.9873675.

Rabindranath, S., Kv, P., Bj, D. and Guha, T. (2022) ‘License Plate Detection using Computer Vision technique with Artificial Intelligence’, in *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon). 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, Mysuru, India: IEEE, pp. 1–5. Available at: https://doi.org/10.1109/MysuruCon55714.2022.9972497.

Radha, R. and Viju, V.R. (2022) ‘Automated Vehicle Number Plate (VNP) Detection based on Optimized Segmentation and Machine Learning’, in *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS). 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India: IEEE, pp. 292–306. Available at: https://doi.org/10.1109/ICSCDS53736.2022.9760741.

‘Raspberry Pi Imager’ (2023). Available at: https://www.raspberrypi.com/software/.

Roboflow (2021) *Original License Plates*. Available at: https://public.roboflow.com/object-detection/license-plates-us-eu/3.

Rokonuzzaman, M., Abdullah Al Amin, M., Ahmed, M.H.K.M.U. and Rahman, M.T. (2017) ‘Automatic vehicle identification system using machine learning and robot operating system (ROS)’, in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE). 2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka: IEEE, pp. 253–258. Available at: https://doi.org/10.1109/ICAEE.2017.8255362.

Rosebrock, A. (2015) *Increasing Raspberry Pi FPS with Python and OpenCV*, *Increasing Raspberry Pi FPS with Python and OpenCV*. Available at: https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/.

Roy, S., Shivakumara, P., Roy, P.P., Pal, U., Tan, C.L. and Lu, T. (2015) ‘Bayesian classifier for multi-oriented video text recognition system’, *Expert Systems with Applications*, 42(13), pp. 5554–5566. Available at: https://doi.org/10.1016/j.eswa.2015.02.030.

Saranya, D., Kanagavalli, N. and Priyaradhikadevi, T. (2021) ‘The Proficient ML method for Vehicle Detection and Recognition in Video Sequence’, in *2021 International Conference on System, Computation, Automation and Networking (ICSCAN). 2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*, Puducherry, India: IEEE, pp. 1–5. Available at: https://doi.org/10.1109/ICSCAN53069.2021.9526504.

Sarif, Md.M., Pias, T.S., Helaly, T., Tutul, Md.S.R. and Rahman, Md.N. (2020) ‘Deep Learning-Based Bangladeshi License Plate Recognition System’, in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Istanbul, Turkey: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ISMSIT50672.2020.9254748.

Sasi, A., Sharma, S. and Cheeran, A.N. (2017) ‘Automatic car number plate recognition’, in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). 2017 4th International Conference on Innovations in Information, Embedded and Communication Systems*, Coimbatore, India: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICIIECS.2017.8275893.

Sathya, K.B., Vasuhi, S. and Vaidehi, V. (2020) ‘Perspective Vehicle License Plate Transformation using Deep Neural Network on Genesis of CPNet’, *Procedia Computer Science*, 171, pp. 1858–1867. Available at: https://doi.org/10.1016/j.procs.2020.04.199.

Satsangi, M., Yadav, M. and Sudhish, P.S. (2018) ‘License Plate Recognition: A Comparative Study on Thresholding, OCR and Machine Learning Approaches’, in *2018 International Conference on Bioinformatics and Systems Biology (BSB). 2018 International Conference on Bioinformatics and Systems Biology (BSB)*, Allahabad, India: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/BSB.2018.8770662.

Selmi, Z., Ben Halima, M. and Alimi, A.M. (2017) ‘Deep Learning System for Automatic License Plate Detection and Recognition’, in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto: IEEE, pp. 1132–1138. Available at: https://doi.org/10.1109/ICDAR.2017.187.

Selmi, Z., Halima, M.B., Pal, U. and Alimi, M.A. (2020) ‘DELP-DAR system for license plate detection and recognition’, *Pattern Recognition Letters*, 129, pp. 213–223. Available at: https://doi.org/10.1016/j.patrec.2019.11.007.

Shafi, I., Hussain, I., Ahmad, J., Kim, P.W., Choi, G.S., Ashraf, I. and Din, S. (2022) ‘License plate identification and recognition in a non-standard environment using neural pattern matching’, *Complex & Intelligent Systems*, 8(5), pp. 3627–3639. Available at: https://doi.org/10.1007/s40747-021-00419-5.

Sunil, A., Samuel, A. and C V, P. (2022) ‘An Imperious Verdict for The Recognition of Vehicles Number-Plate Using an Innovative Methodology’, in *2022 Second International Conference on Interdisciplinary Cyber Physical Systems (ICPS). 2022 Second International Conference on Interdisciplinary Cyber Physical Systems (ICPS),* Chennai, India: IEEE, pp. 46–50. Available at: https://doi.org/10.1109/ICPS55917.2022.00016.

Tabrizi, S.S. and Cavus, N. (2016) ‘A Hybrid KNN-SVM Model for Iranian License Plate Recognition’, *Procedia Computer Science,* 102, pp. 588–594. Available at: https://doi.org/10.1016/j.procs.2016.09.447.

Tang, J., Wan, L., Schooling, J., Zhao, P., Chen, J. and Wei, S. (2022) ‘Automatic number plate recognition (ANPR) in smart cities: A systematic review on technological advancements and application cases’, *Cities,* 129, p. 103833. Available at: https://doi.org/10.1016/j.cities.2022.103833.

Trapeznikov, I., Priorov, A. and Volokhov, V. (2014) ‘Allocation of text characters of automobile license plates on the digital image’, in *Proceedings of 15th Conference of Open Innovations Association FRUCT. 2014 15th Conference of Open Innovations Association FRUCT and 3rd Regional Seminar on e-Tourism (FRUCT),* St. Petersburg, Russia: IEEE, pp. 144–149. Available at: https://doi.org/10.1109/FRUCT.2014.6872421.

ultralytics (2023) *yolov5, ultralytics/yolov5*. Available at: https://github.com/ultralytics/yolov5/.

Varma P, R.K., Ganta, S., B, H.K. and Svsrk, P. (2020) ‘A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition using Image Processing Techniques’, *Procedia Computer Science,* 167, pp. 2623–2633. Available at: https://doi.org/10.1016/j.procs.2020.03.324.

Wang, B., Xiao, H., Zheng, J., Yu, D. and Chen, C.L.P. (2022) ‘Character Segmentation and Recognition of Variable-Length License Plates Using ROI Detection and Broad Learning System’, *Remote Sensing,* 14(7), p. 1560. Available at: https://doi.org/10.3390/rs14071560.

Yang, R., Yin, H. and Chen, X. (2015) ‘License Plate Detection Based on Sparse Auto-Encoder’, in *2015 8th International Symposium on Computational Intelligence and Design (ISCID). 2015 8th International Symposium on Computational Intelligence and Design (ISCID),* Hangzhou, China: IEEE, pp. 465–469. Available at: https://doi.org/10.1109/ISCID.2015.151.

Yao, Z. and Yi, W. (2014) ‘License plate detection based on multistage information fusion’, *Information Fusion,* 18, pp. 78–85. Available at: https://doi.org/10.1016/j.inffus.2013.05.008.

Ying, T., Xin, L. and Wanxiang, L. (2018) 'License plate detection and localization in complex scenes based on deep learning', in *2018 Chinese Control And Decision Conference (CCDC). 2018 Chinese Control And Decision Conference (CCDC)*, Shenyang: IEEE, pp. 6569–6574. Available at: https://doi.org/10.1109/CCDC.2018.8408285.

Zhang, M., Yu, W., Su, J. and Li, W. (2019) 'Design of License Plate Recognition System Based on Machine Learning', in *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC). 2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, Xiamen, China: IEEE, pp. 518–522. Available at: https://doi.org/10.1109/ICIVC47709.2019.8981074.

Zhang, X., Ni, X., Deng, Y., Jiang, C. and Maleki, M. (2021) 'Chinese License Plate Recognition Using Machine and Deep Learning Models', in *2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML). 2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML)*, Chengdu, China: IEEE, pp. 342–346. Available at: https://doi.org/10.1109/PRML52754.2021.9520386.

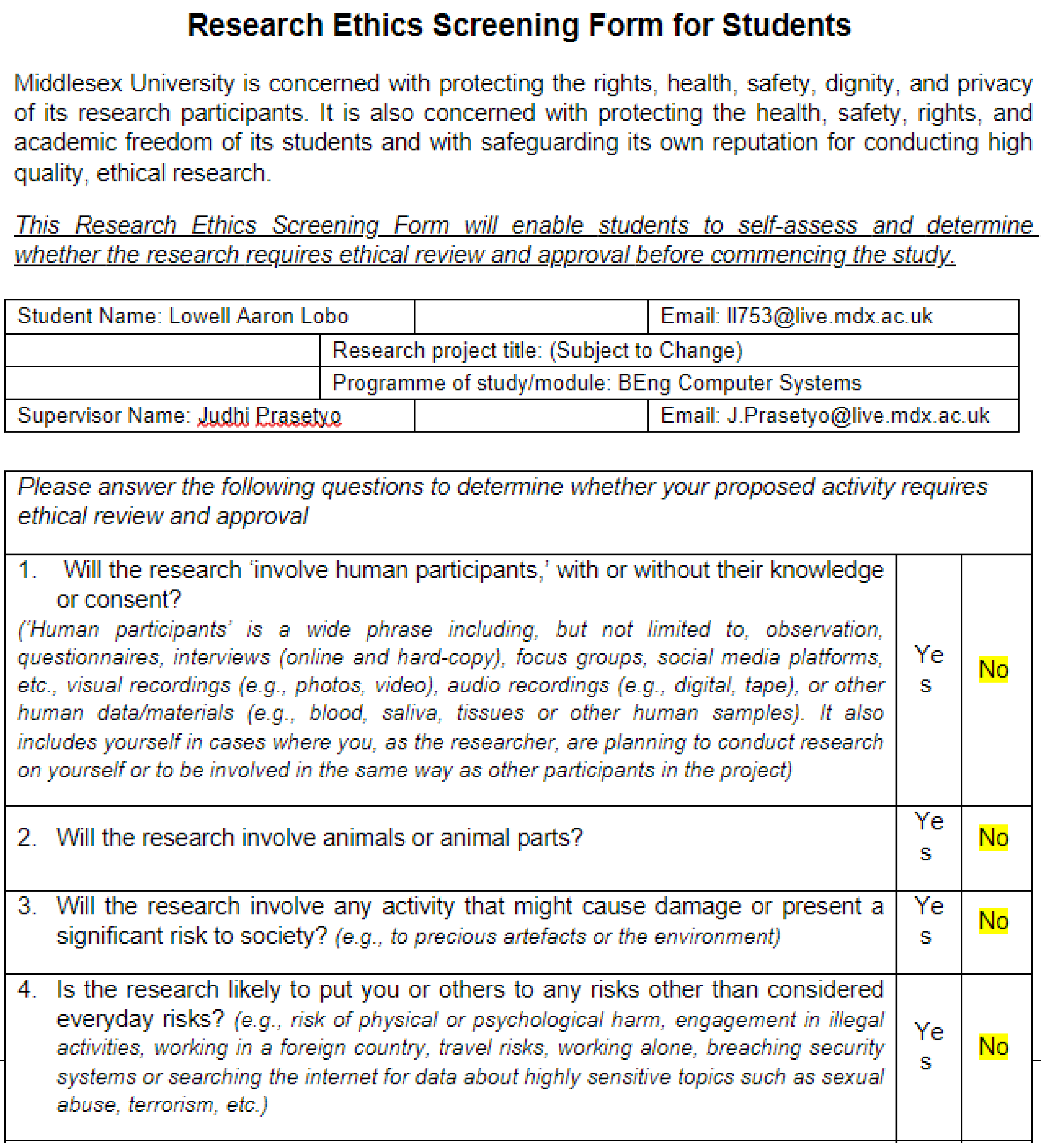
Appendix

Meeting Log

Date	Time	Discussion
17 January 2023	11:45 am	Initial greetings and discussion on how to do the project and how big should the project be
1 February 2023	11:50 am	Took advice about which project to do. Formalised the project outcome and gained feedback on which scenario to implement. Emphasis on testing all scenarios and environmental conditions.
2 February	05:00 pm	Ethics form discussion and how to submit
9 February	2:00 pm	Project timeline, Gantt chart and ethics form submission. Verifying if project is ethical or not.
23 February 2023	01:50 pm	Feedback on literature review and what all to include in it. Also got advise to start making the deep learning models.
3 March 2023	11:00 am	Mentioned frontend development is in progress and showed the blog updates. Got feedback to implement the

		project first on a PC and then port it on to a Raspberry Pi
24 March 2023	12:00 pm	Displayed the full project skeleton in the working condition and got advise to test all scenarios

Ethics Screening Form



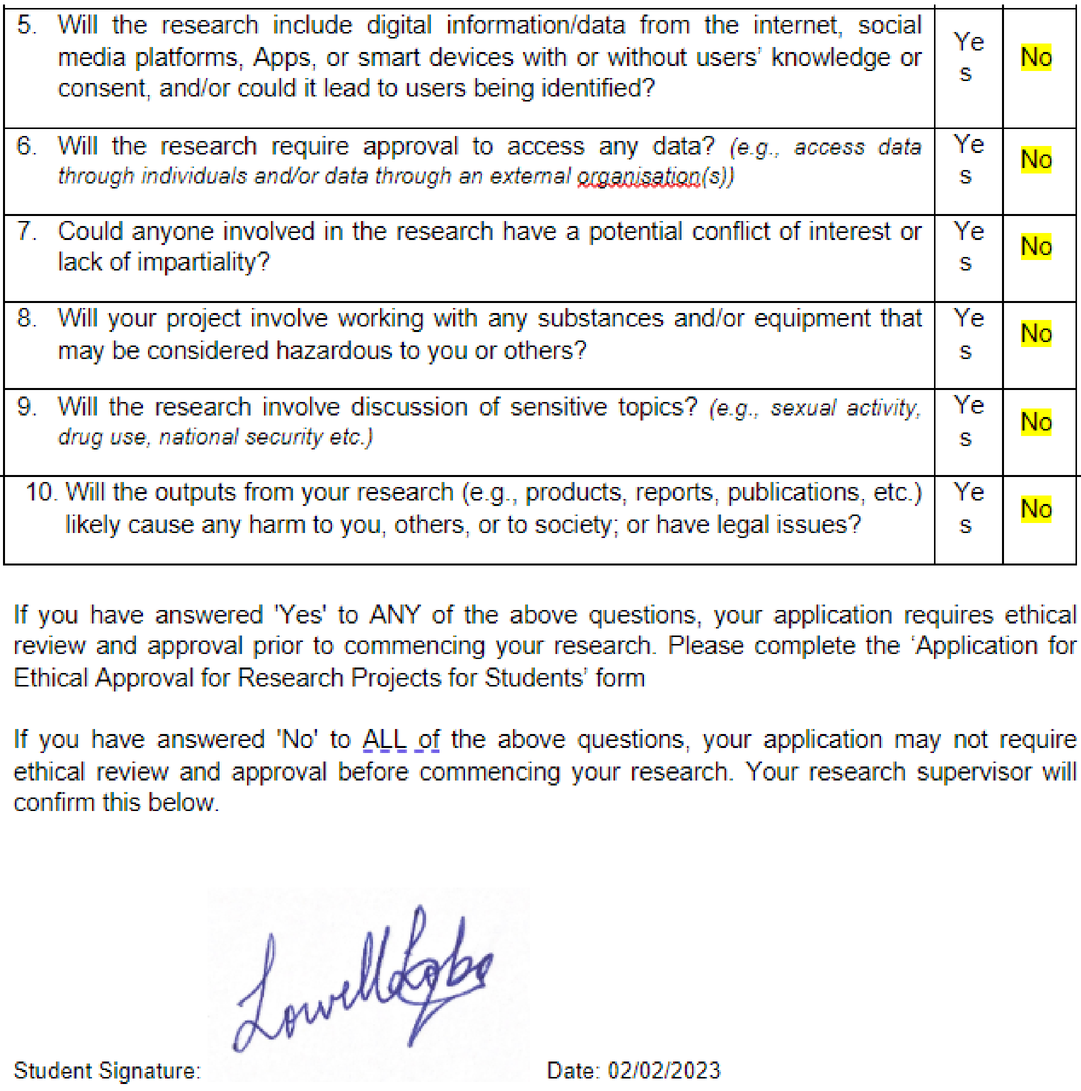
Research Ethics Screening Form for Students

Middlesex University is concerned with protecting the rights, health, safety, dignity, and privacy of its research participants. It is also concerned with protecting the health, safety, rights, and academic freedom of its students and with safeguarding its own reputation for conducting high quality, ethical research.

This Research Ethics Screening Form will enable students to self-assess and determine whether the research requires ethical review and approval before commencing the study.

Student Name: Lowell Aaron Lobo		Email: ll753@live.mdx.ac.uk
	Research project title: (Subject to Change)	
	Programme of study/module: BEng Computer Systems	
Supervisor Name: Judhi Prasetyo		Email: J.Prasetyo@live.mdx.ac.uk

Please answer the following questions to determine whether your proposed activity requires ethical review and approval		
1. Will the research 'involve human participants,' with or without their knowledge or consent? *('Human participants' is a wide phrase including, but not limited to, observation, questionnaires, interviews (online and hard-copy), focus groups, social media platforms, etc., visual recordings (e.g., photos, video), audio recordings (e.g., digital, tape), or other human data/materials (e.g., blood, saliva, tissues or other human samples). It also includes yourself in cases where you, as the researcher, are planning to conduct research on yourself or to be involved in the same way as other participants in the project)*	Yes	[✓ No]
2. Will the research involve animals or animal parts?	Yes	[✓ No]
3. Will the research involve any activity that might cause damage or present a significant risk to society? *(e.g., to precious artefacts or the environment)*	Yes	[✓ No]
4. Is the research likely to put you or others to any risks other than considered everyday risks? *(e.g., risk of physical or psychological harm, engagement in illegal activities, working in a foreign country, travel risks, working alone, breaching security systems or searching the internet for data about highly sensitive topics such as sexual abuse, terrorism, etc.)*	Yes	[✓ No]



5. Will the research include digital information/data from the internet, social media platforms, Apps, or smart devices with or without users' knowledge or consent, and/or could it lead to users being identified?	Yes	[✓ No]
6. Will the research require approval to access any data? *(e.g., access data through individuals and/or data through an external organisation(s))*	Yes	[✓ No]
7. Could anyone involved in the research have a potential conflict of interest or lack of impartiality?	Yes	[✓ No]
8. Will your project involve working with any substances and/or equipment that may be considered hazardous to you or others?	Yes	[✓ No]
9. Will the research involve discussion of sensitive topics? *(e.g., sexual activity, drug use, national security etc.)*	Yes	[✓ No]
10. Will the outputs from your research (e.g., products, reports, publications, etc.) likely cause any harm to you, others, or to society; or have legal issues?	Yes	[✓ No]

If you have answered 'Yes' to ANY of the above questions, your application requires ethical review and approval prior to commencing your research. Please complete the 'Application for Ethical Approval for Research Projects for Students' form

If you have answered 'No' to ALL of the above questions, your application may not require ethical review and approval before commencing your research. Your research supervisor will confirm this below.

Student Signature: [illegible] Date: 02/02/2023

To be completed by the supervisor:

Based on the details provided in the self-assesment form, I confirm that:	
The study does not require ethical review and approval	☒
The study requires ethical review and approval	☐

Superivsor Signature:[illegible]....... Date:..23-2-2023.........