Faculty of Engineering

Self-Admitting Parking System

Supervised by

Prof. Mona Ahmed Fahmy Ismail

Submitted by

Abdullah Ashraf Abdelaaty

Ahmed Montasser AbdulWahab Al-Azab

Hazem Khaled seif

Mohamed Samy mohamed

Mostafa Abdelraouf

Radwa Ahmed Mostafa

2021-2022

Acknowledgement

Our special thanks to our supervisor Prof. Mona Ahmed Fahmy Abd El Baky Ismail for providing her support and valuable suggestions regarding project work. We extend our sincere thanks to logitime Egypt Engineers, Eng. Mohamed Ragab and Eng. Rafeek Abadeer for providing valuable guidance at every stage, along with the necessary environment to test the project.

Abstract

With the rapid increase of car usage in the past few years, human management of parking areas is becoming more difficult and time consuming. This is about developing a good and efficient system that takes over the problem of identifying the driver ID, associating the driver with his vehicle and maintaining the record of vehicles parked accurately. Human interference is minimized at the parking area to a great extent such as authorizing the driver to enter the garage and the payment process are all done automatically. The various steps that this project comprises are face recognition, license plate recognition and interacting through web application. Both face recognition and number plate recognition use computer vision technology.

The technology adopted in this research can be utilized not only for security, but also in other fields such as parking fee collection, automatic speed control or tracking stolen cars.

Table of Contents

Chapter 1: Introduction

1.1 Overview

Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Machines can accurately identify and locate objects then react to what they “see” using digital images from cameras, videos, and deep learning models.

Face recognition is one of the much-studied biometrics technology and developed by different companies. Human face recognition procedure basically consists of two phases, first, face detection… The next is face recognition, which recognizes a face as individuals.

Automatic license plate recognition (ALPR) is the process of retrieving license plate information from a captured image or video frames from a sequence of videos.

Optical Character Recognition (OCR) is a field of artificial intelligence that converts images with text into machine processed text. OCR systems are available for several languages including Arabic, Arabic OCR has been developed and improved over decades, which ultimately causes a huge number of approaches with robust results. Using deep learning in Arabic OCR can result in 100% accuracy in a shorter time and less resources to process the image. The characteristics of Arabic text cause more errors than English text in OCR.

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign learnable weights and biases to various objects in the image and be able to differentiate one from the other.

1.2 Digital access modules for ticketless parking system

The ticketless parking solution has digital modules. Thanks to them, it allows access to the parking facility and identifies each person and its vehicle. Here are the parts of the project:

1. Face recognition camera

Ticketless parking systems depend on FR cameras. They recognize the face of every customer. It adds a layer of security to the system.

2. ANPR camera

Ticketless parking systems work with ANPR cameras. They recognize the LPN of every vehicle. For new parking customers, the cameras register new faces and LPNs and allow their access. For subscribed customers, the system allows immediate access, because the face and LPN is previously registered.

3. Web parking app

We provide a user-friendly interface to deal with the hidden modules.

1.3 Why use a smart parking system?

1.3.1 Benefits for customers:

- **Better customer experience and satisfaction**

Parking customers really like the ticketless parking system as well as they do not have to deal with the control device to get a ticket. That way, customers have minimal interaction with the parking management system. It is far more convenient!

- **Easy entry/exit**

The ticketless parking system makes the entry/exit process easier than ever! By installing that type of parking solution, customers will enjoy a convenient parking experience. Automatic barrier opens, no more lost time in waiting for getting or scanning a paper ticket.

- **Fast flow-through of cars**

Thanks to the easy entry/exit process, cars move faster without the need to get or insert a ticket. The ticketless parking system is dynamic and works mostly with recognizing the driver face and the car LPN.

- **Online payments**

This innovative parking management solution offers many modern methods for online payments. Customers can pay for their parking stay via the application.

Also pay stations can be installed in the parking space for the traditionalists. It will allow payment with cash or credit/debit cards.

In addition, parking customers can subscribe and pay for the service monthly.

- **Privacy**

Another benefit for parking customers is that their privacy is secure. The ticketless parking system does not record any personal information. The only data collected are the license plate numbers, times of entry and exit and the collected parking fees. That is necessary for the easier entry/exit ticketless process.

- **No need to press any button for a ticket**

Regarding the current situation of COVID 19, we have to be extremely careful with surfaces touched by many people. A suitable example of such a surface is the button for getting a paper ticket in the supermarket parking facilities. Hundreds of people a day push it. That makes this surface a potential place where COVID 19 can linger on. The ticketless parking system eliminates the risk of spreading the virus because you do not need to press any button.

1.3.2 Benefits for parking operators:

- **Less costs and more revenue**

The ticketless parking system reduces the equipment maintenance and the need of hardware such as ticket terminals. That way, parking operators can save money from reduced investment costs. In addition, this solution eliminates the costs for paper tickets, replacing them with virtual ones.

- **Reduced on-site staff**

This innovative parking management solution is autonomous. It works without or with less on-site stuff. Instead, it controls the access to the parking facility via different digital modules and cloud-based software. Also, the ticketless parking system can work 24/7, non-stop day and night.

- **Eco-friendly**

Ticketless parking system is a great eco-friendly solution! It means no paper tickets. No spare parts, wastage, litter to collect. No ticket jams.

- **Remote parking control**

One of the best advantages of the ticketless solution for parking operators is the option for remote control. You can easily manage your parking facility and your parking customers' booking and payments. All that from a distance only by a mobile or a computer.

1.4 Use cases

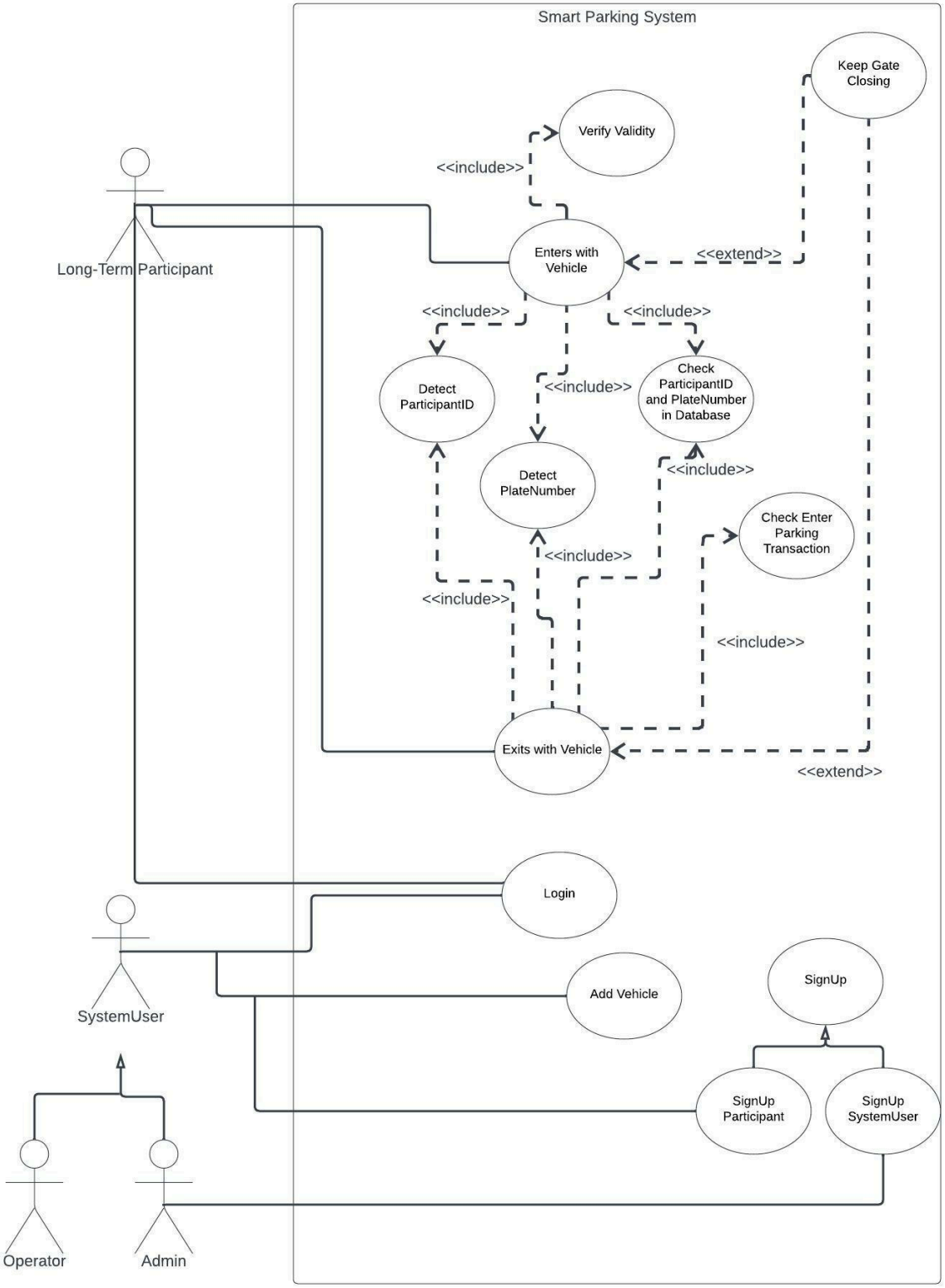

*Figure 1.1 UseCase Diagram for Smart Parking System (*subscriber-customer scenario)

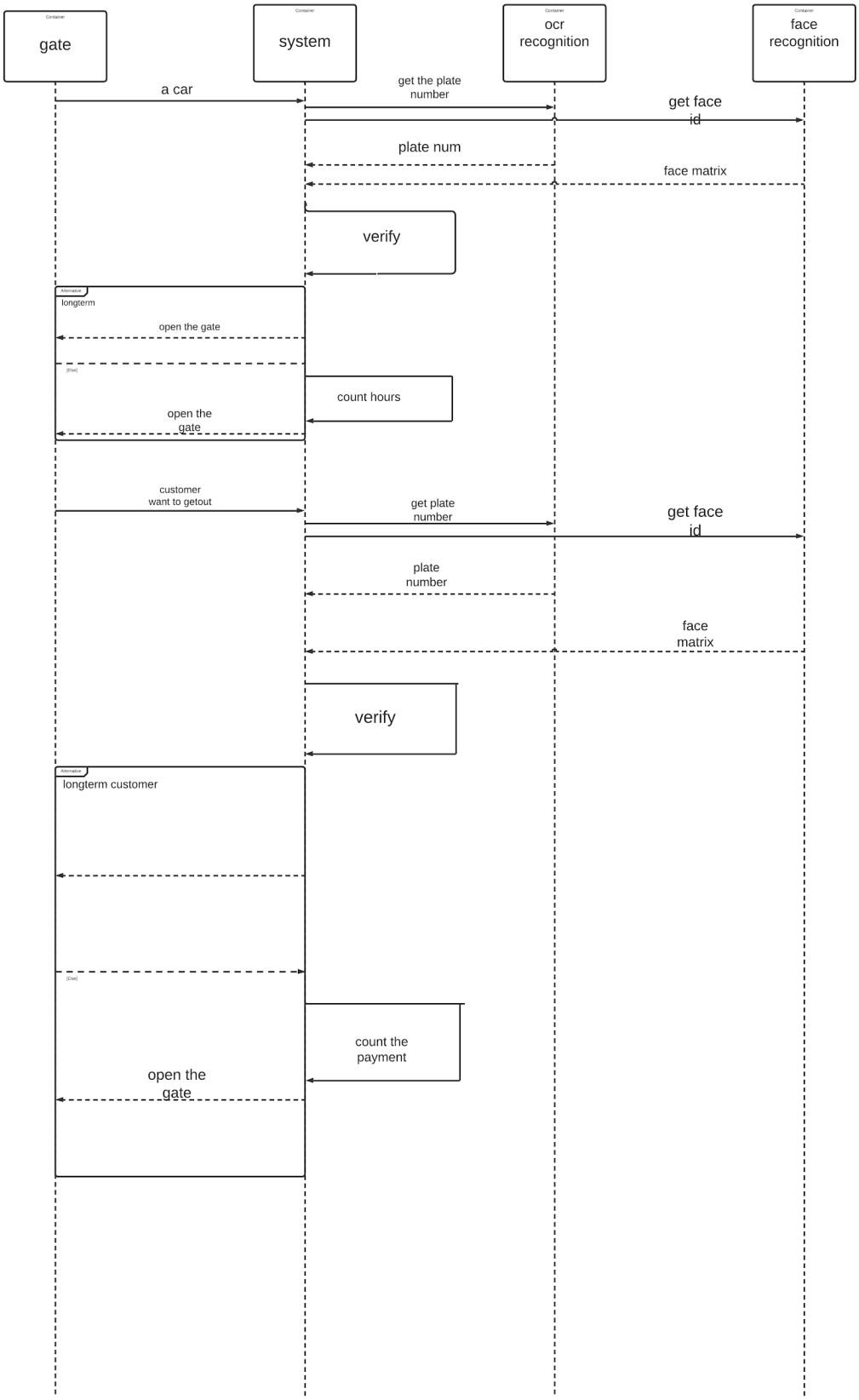

Figure 1.2 Sequence Diagram

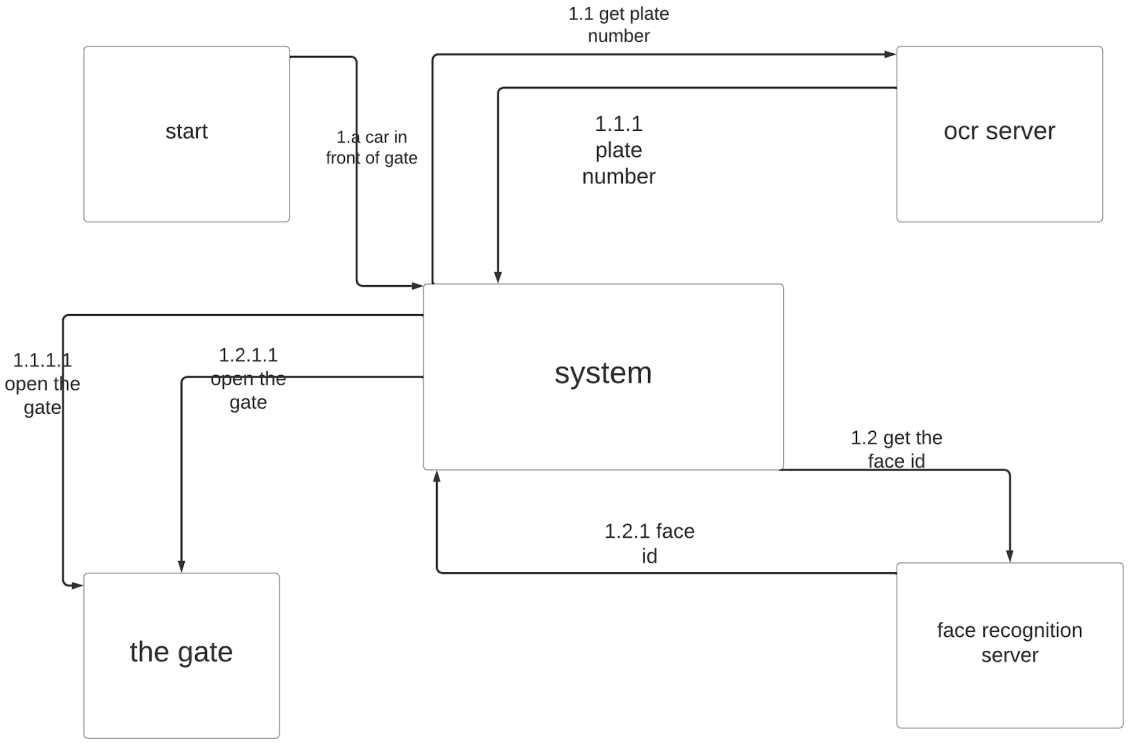

Figure 1.3 Communication Diagram

1.5 Who does this project target?

A project like ours mainly targets parking spaces in public places like malls, supermarkets, and parks. It is also suitable for big companies and hospitals to provide a convenient parking experience for their staff and employees.

A similar project can be utilized for governmental use. With the government authority, it would be easier to install a barcode on all cars. The barcode will hold all the necessary information to identify the driver and the vehicle. Scanning a barcode would be far easier, but it doesn't come up with the level of security that the face recognition module alone provides. In Chapter 2: Face Module, we will discuss more about Biometric Security and why it is a global trend.

Moreover, the information gathered via the implementation of the Smart Parking System can be exploited to predict future parking patterns. Collecting such statistical data is the key to reducing costs and increasing revenue. Not to forget that a huge benefit of the smart parking system as parking spaces can be fully utilized.

1.6 Agile software development

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Agile methodology can fit with our team.

- **Iterative, incremental, and evolutionary**

Agile development methods can minimize the amount of up-front planning and design using iterations. Each iteration involves a cross-functional team working in all functions: planning, analysis, design, coding, unit testing, and acceptance testing. It helps to release the product or new features quickly. Also it minimizes overall risk and allows the product to adapt to changes quickly with minimal bugs.
Agile's main principle is that the most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- **Efficient and face-to-face communication**

With the widespread adoption of remote working during the COVID-19 pandemic and changes to tooling, more studies have been conducted around co-location and distributed working, which shows that co-location is increasingly less relevant.

- **Very short feedback loop and adaptation cycle**

A common characteristic in agile software development is the daily stand-up meeting of about 15 minutes. During the meeting, team members review collectively how they are progressing toward their goal.

Chapter 2: Face Recognition Module

2.1 Biometric Security

Face recognition is a biometric solution designed to recognize a human face without any physical contact required. The solution runs through algorithms that match the facial nodes of a person into the database. Face recognition identifies the unique features of the human face and then makes a comparison based on the existing database of photographs. Sensors detect and identify face shapes by the color of the iris, nose shape, and so on. Identifying the human face includes concentrating on certain unique features.
The face recognition solution has been a major component in the field of security: criminal identification, surveillance, police authorities, bank services, online payment, and parking services.

Face recognition pros:

- Major security. Only your face unlocks the device.
- Convenience and simplicity.
- Enhance the organization of photographs.
- Flexible and powerful.

Face recognition cons:

- Access to sensitive data.
- Technology has its limits. Sometimes even the best systems can have a glitch.
- data. Recording and scanning with face recognition technology can make people feel like they are being constantly monitored and analyzed.
- Facial recognition is affected by lighting, makeup, and even sometimes by the user's natural skin tone.

2.2 Face identification and recognition

Face recognition is a computer vision task of identifying and verifying a person based on a photograph of their face. A face recognition system is expected to identify faces present in images and videos automatically. It has two steps:

1. Face detection which is the process of automatically locating faces in a photograph and localizing them by drawing a bounding box around their extent (e.g. is this the person?)
2. Face identification (or recognition) which is a one-to-many mapping for a given face against a database of known faces (e.g. who is this person?).

2.3 Current solutions

First, let's discuss approaches we tried for face detection:

2.3.1 Haar Cascades

- Haar Cascades is a very efficient and fast technique used for face detection.
- It uses Line detection features proposed by viola and jones in their paper [1] "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001.
- For example Haar Cascades Features can detect horizontal and vertical edges by detecting the light intensity change like in Figure(1)

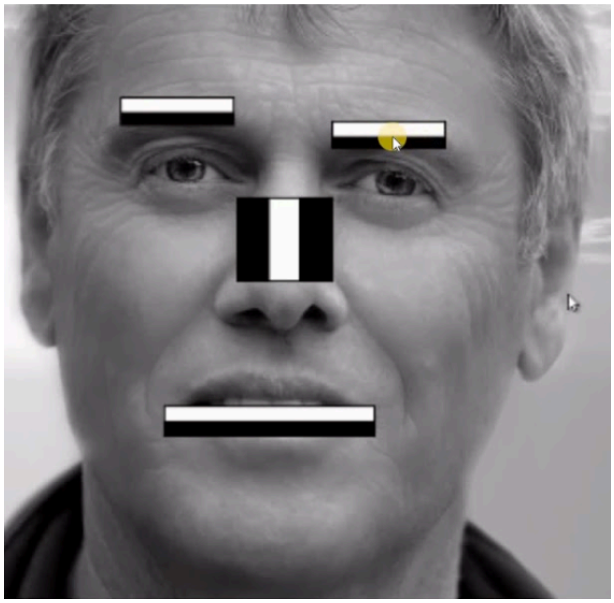

Figure 2.1 Haar Cascades identifying face features.

- We apply those features by passing them on the whole image which is not efficient.
- So we have Optimization Steps in this Algorithm:
- First technique is The integral image:

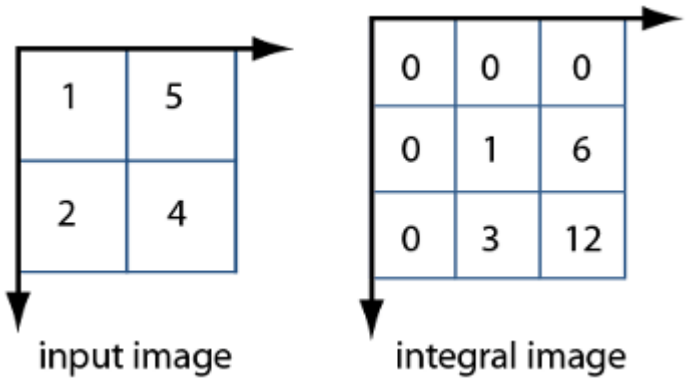

Figure 2.2 Integral image.

    - Each cell in the integral image is the summation of itself and all cells to the left and up.
    - Instead of applying the features on the whole pixels of the image the integral image makes the calculation independent of the image size and reduces them to only four calculations.

- Second Technique is the AdaBoost:
    - The feature set has 180000 features.
    - Majority of these features are not irrelevant to facial features.
    - AdaBoost technique is a selection technique to eliminate those irrelevant features.
    - We get 6000 features out of the 180000 after applying this technique.

- Third Step is Cascades Of Classifiers:
    - Instead of applying the whole 6000 features on each window of the image.
    - The features are organized In cascades.
    - Total of 38 stages of features.
    - The first 5 stages contain 1=>10=>25=>25=>50 features.
    - If a window fails in the first cascades it will be discarded.
    - On average only 10 features are applied on each non-face sub-window instead of the whole 6000 features.

2.3.2 (HOG) Histogram of Oriented Gradients

In their paper, Dalal and Triggs [2] describe HOGs as a feature descriptor that has been used for object and pedestrian detection. A HOG relies on the property of objects within an image to possess the distribution of intensity gradients or edge directions.
The distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and usually, edges and corners pack in a lot more information about object shape than flat regions.

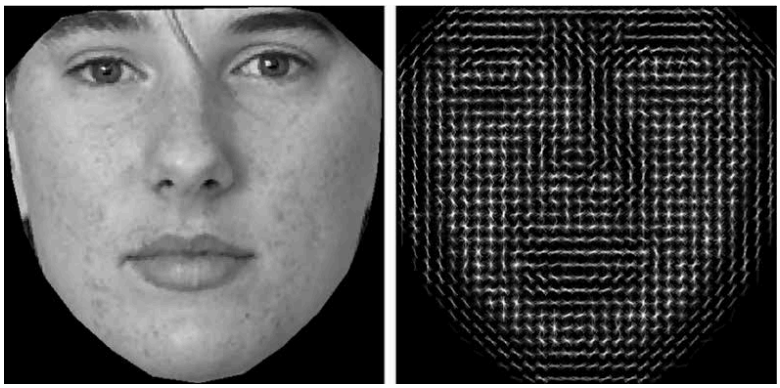

Figure 2.3 output of HOG face pattern for a sample image.

The basic idea of HOG is dividing the image into small connected cells like the left, then computes a histogram for each cell using following formulas:

$$g = \sqrt{g_x^2 + g_y^2} \qquad \theta = arctan\frac{g_y}{g_x}$$

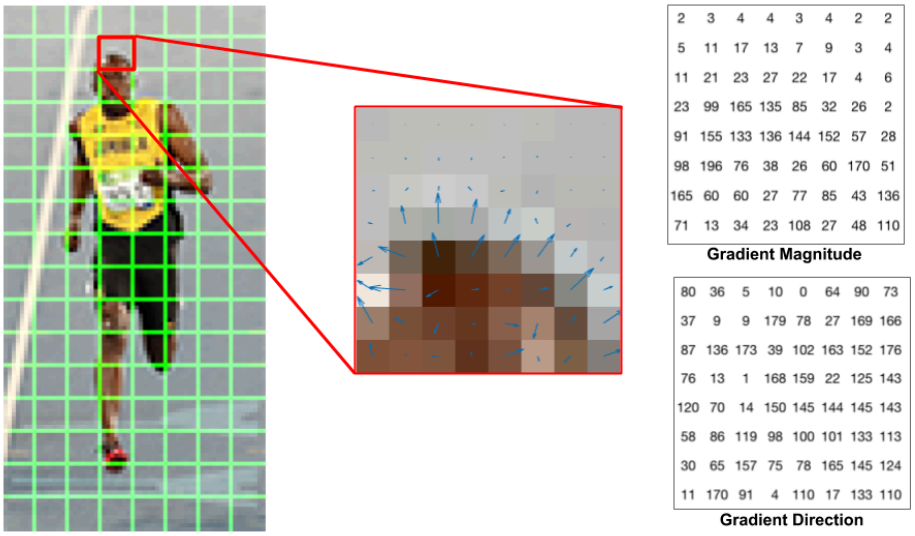

Figure 2.4 Center: The RGB patch and gradients represented using arrows. Right: The gradients in the same patch represented as numbers

The next step is to bring all histograms together to form a feature vector i.e., it forms one histogram from all small histograms which is unique for each face.

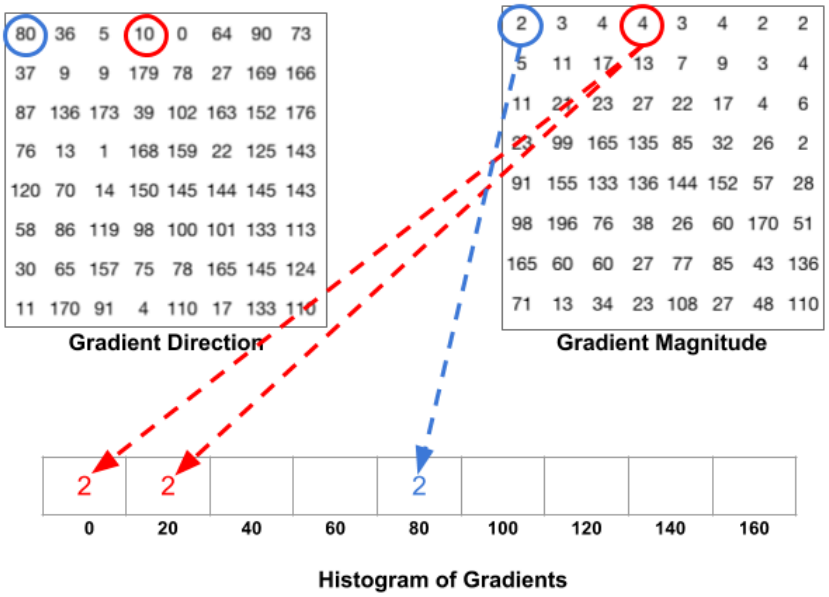

Figure 2.5 Histogram of features vector

Finally to visualize the HOG descriptor of an image, we plot the 9×1 normalized histograms in the 8×8 cells. See image on the side. You will notice that the dominant direction of the histogram captures the shape of the person, especially around the torso and legs.

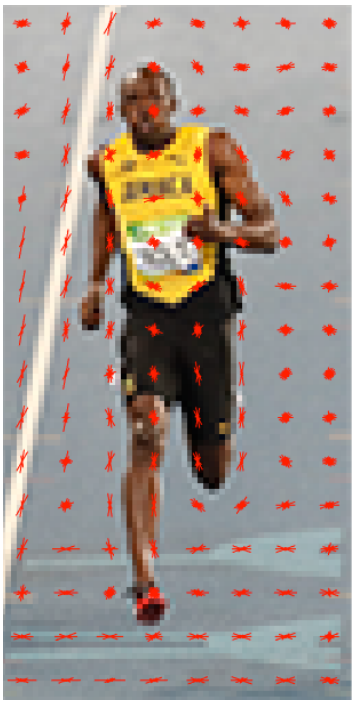

Figure 2.6 the direction of the histogram captures the shape of the person.

Up until now, we have discussed two approaches to classical face detection techniques. Let's see CNN based face detection, then summaries why it is better to use one.

2.3.3 MTCNN (Multi-task Cascaded Convolutional Neural Networks)

MTCNN is one of the most popular and most accurate face detection tools today. Based on the paper published in 2016 by (Zhang et al.) [3], it consists of three convolutional networks.

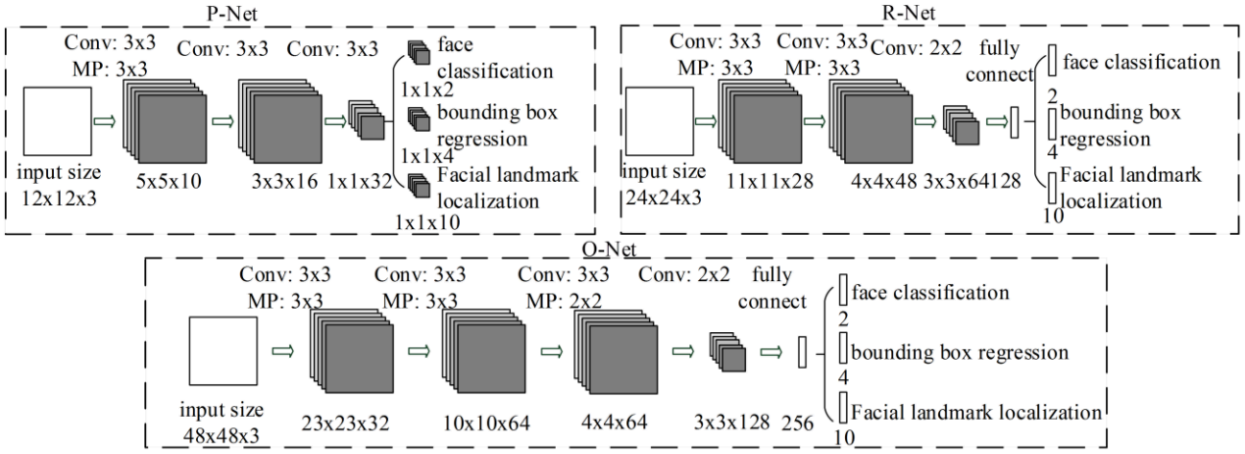

Figure 2.7 MTCNN Structure.

- First, we create the image pyramid for our input image to be able to detect all faces of different sizes in the input.
- This pyramid is the input for the following three-stage cascaded network.

1. Stage 1: P-Net

- This stage is used to obtain all candidate windows of faces in the previous image pyramid and their bounding box regression vector.
- After the bounding box regression is done we do some refinement to combine narrowly overlapped boxes to downsize the volume of candidates.

2. Stage 2: R-Net
    - In this stage we do more optimization by reducing the number of candidate windows obtained from the previous stage
    - Three outputs of this stage:
        - Face classification (if there is a face or not.)
        - Vector of Bounding Box coordinates.
        - Vector of Facial Landmarks localization.

3. Stage 3: O-Net

- This Stage is similar to R-Net but it has a more specific goal which is to get the five facial landmarks:
    - Left eye.
    - Right eye.
    - Nose.
    - Mouth left corner.
    - Mouth right corner.

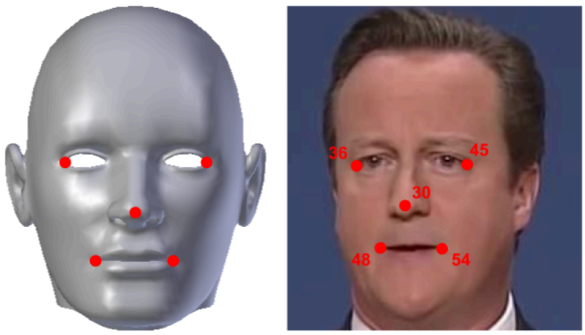

Figure 2.8 Five Facial Landmarks.

So we have Three outputs of this whole Stage:

1. Face Classification:
    - Binary Classification using binary cross-entropy loss.
2. Bounding box regression:
    - Euclidean loss is used.
3. Facial Landmarks:
    - Locating all Face landmarks and using Euclidean distance as a loss function.

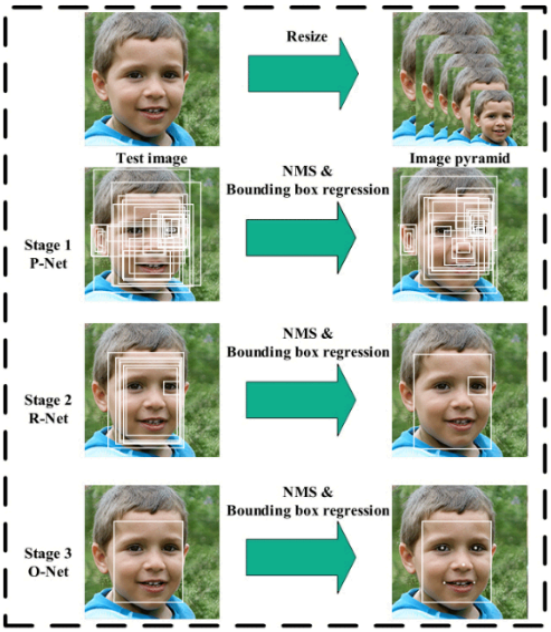

Figure 2.9 MTCNN Face Detection.

2.3.4 Comparison Of previous Face Detection Techniques

The next Numbers are based on a processor intel(R) core(TM) i7-7700HQ @ 2.80 GHz.

	Haar Cascades	HOG	MTCNN
FPS (frames/second)	25	13	20
Accuracy	85%	89%	99%~99.50%

Table(2.1)

To summarize, Classical approaches like haar and HOG don't work on faces at odd angles. It only works with straight and front faces, In addition to that the Haar technique results in a lot of false positive detections. It requires manually calibrating the parameters which is not practical at all. It is really useful if you use it to detect faces from scanned documents like driver's licenses and passports but not a good fit for real-time video. When working with real-time video streaming, CNN-based face detection is much better.

2.4 Face recognition

Second, let's tackle Face Recognition techniques:

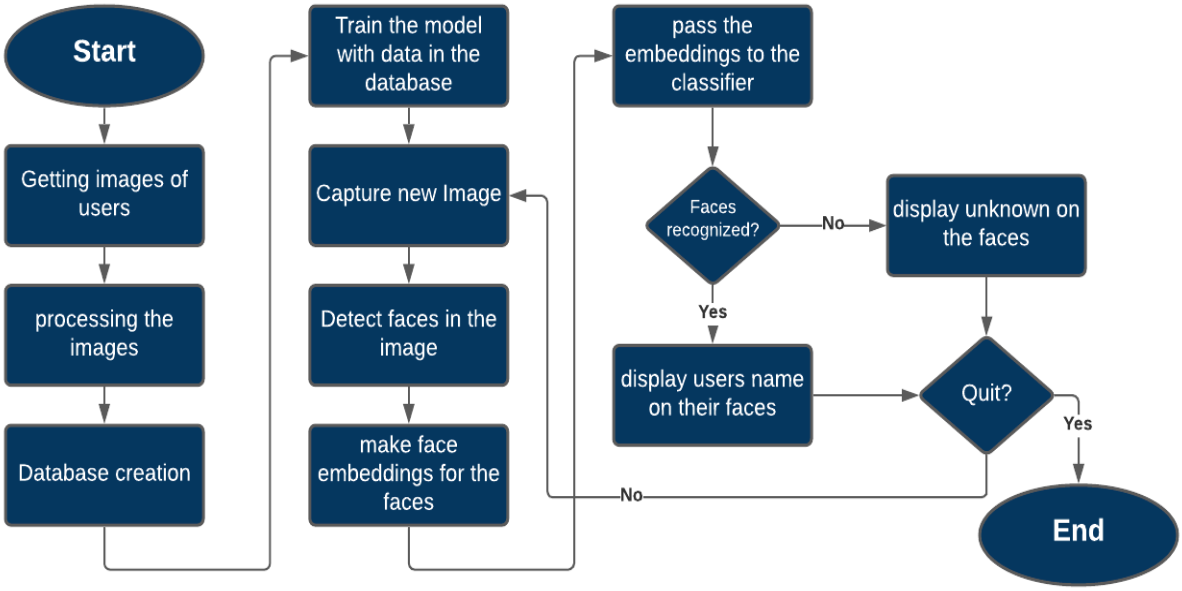

Figure 2.10 Face Recognition Pipeline.

KNN

- K-nearest neighbor belongs to classical machine learning under supervised classification.
- We used Haar cascades for detection and KNN for creating classifiers of each person in our database.

Dlib

- Lib is an open source C++ library implementing a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction.

2.4.1 FaceNet

- FaceNet is a face recognition system developed in 2015 by researchers at Google in their paper titled FaceNet: A Unified Embedding for Face Recognition and Clustering [4].

- The FaceNet system can be used to extract high-quality features from faces, called face embeddings. These face embeddings were then used as the basis for training classifier systems on standard face recognition benchmark datasets, achieving then-state-of-the-art results on a range of face recognition benchmark datasets.
- The focus on training a model to create embeddings directly (rather than extracting them from an intermediate layer of a model) was an important innovation in this work. The FaceNet system can be used broadly thanks to multiple third-party open source implementations of the model and the availability of pre-trained models.

- The model is a deep convolutional neural network trained via a triplet loss function that encourages vectors for the same identity to become more similar (smaller distance), whereas vectors for different identities are expected to become less similar (larger distance).
- The focus on training a model to create embeddings directly (rather than extracting them from an intermediate layer of a model) was an important innovation in this work.

2.4.2 Comparison of previous Face Recognition techniques

The next Numbers are based on a processor intel(R) core(TM) i7-7700HQ @ 2.80 GHz.

	KNN	Dlib	Facenet
FPS(Frames/second)	25	3	9
Accuracy	84%	97%	99%~99.8%

Table(2.2)

- To summarize using KNN was very fast but with a lot of false detection which leads to low accuracy, Using Dlib in my case was very slow, which isn't suitable for our application
- So Facenet was The reasonable choice for our purpose.

2.4.3 Further Optimization on Face recognition

- We faced a noticed lag while using the CCTV camera
- After encoding the .h264 to mjpeg we transferred the load on the network instead of the application so lag was reduced significantly.
- Our application also faced low frame rate while operating on cpu.
- We handled that by using CUDA Toolkit, So we were operating on GPU instead of CPU, which increased the number of frames to reach 15 fps.

Chapter 3: License Plate Detector

3.1 Introduction

In this section of the documentation, the problem of license plate recognition will be thoroughly discussed. Starting from the motives behind choosing an algorithm over the other, theory of operation, specification of the implemented algorithm, obtained results and limitations.

3.2 Algorithm choice

The recent surge in computer vision resulted in a myriad of object detection algorithms that rely on machine learning. That being said, the same task was achieved in the past using traditional image processing, however, this section is focused only on recent technologies and algorithms.

Any detection task should, conventionally, be divided into two sub-tasks; detection of possible object regions - it's also called object localization - and classification of these regions. For example, RCNNs demonstrate this concept clearly. They typically use a Region Proposal Network (RPN) that proposes regions of interest (ROIs) that might contain objects. The output from the RPN is then fed to a classifier which in turn classifies the regions into classes. While RCNNs, and many other algorithms of the same category, give very accurate results with high mean Average Precision (mAP), they possess high latency in real-time detection due to the fact that they're two-stage/proposal networks.

As a result, a greater attention was given to a more recent algorithm called *YOLO*. YOLO is a shorthand for You Only Look Once. Compared to the approaches taken by object detection algorithms before YOLO, which repurpose classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. Following a fundamentally different approach to object detection, YOLO achieves state-of-the-art results beating other real-time object detection algorithms by a large margin. In other words, YOLO only requires a single forward pass to be able to predict all possible bounding boxes and classify them with confidence to each class. The figure below shows the distinction between the two types of detectors discussed until now.

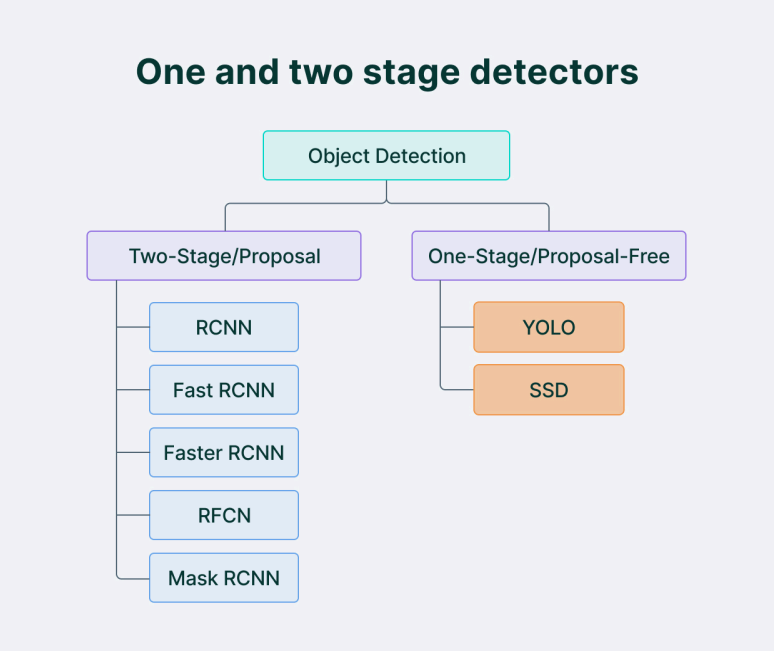

Figure 3.1
Comparison between one and two stage detectors. One-stage detectors, sometimes called proposal-free detectors owing to the lack of a proposal network, offer better performance, hence, are more preferable to real-time detection.

3.3 Theory behind YOLO

The first version of the YOLO algorithm works by dividing the image into *N* grids, each having an equal dimensional region of SxS. Each of these *N* grids is responsible for the detection and localization of the object it contains. Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and confidence of the object being present in the cell. Formally confidence is defined as $Pr(Object) * IoU^{Pred./truth}$, where IoU stands for Intersection over Union. The way IoU is computed is demonstrated in figures 3.2(a) and 3.2(b). Typically, an IoU value above 0.5 is considered true for a class prediction. This process greatly lowers the computation as both detection and recognition are handled by cells from the image as opposed to the previously discussed algorithms. [7]

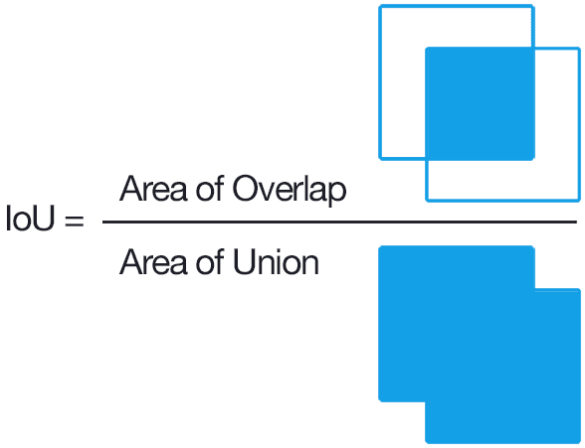

Figure 3.2(a) shows the formula used to compute the IoU value. It's the division of the area of overlap between the predicted bounding box and the ground truth over the area of union.

Figure 3.2(b) shows different bounding boxes alignments relative to the ground truth boxes yielding different values for IoU.

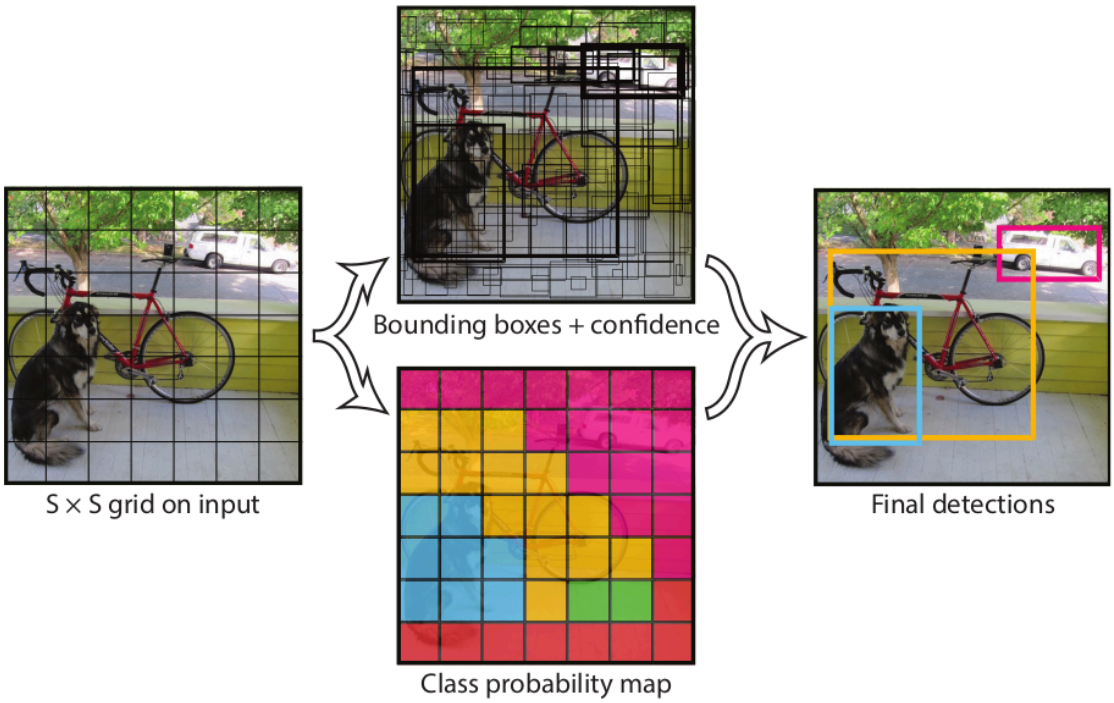

Figure 3.3
An image containing multiple objects is used to demonstrate how YOLOv1 works. The image to the left is the input image with an S x S grid drawn on top of it. Bounding boxes are refined according to IoU and class probabilities to obtain the final detection result shown on the right.

Since 2015 modifications to the original algorithm were introduced in a series of papers by the same authors. These modifications were necessary to tackle issues such as localization errors and low recall compared to competing technologies like Fast-RCNN.[8] The changes made from YOLOv1 until YOLOv4 resulted in substantial improvements in mAP and recall that even starting from YOLOv2 you could obtain better results than other state-of-the-art algorithms in much less time. The figure below shows the effect of applying batch normalization after each convolutional layer in YOLOv1, using anchor boxes for object detection aided with k-means clustering for optimal prior sizes and the utilization of Darknet-19.

Despite the huge improvement in YOLOv2 we decided to use YOLOv4 as our license plate detector due to further improvements in AP and FPS.

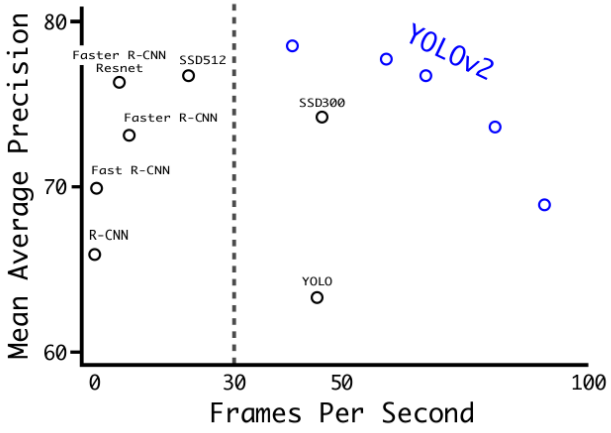

Figure 3.4 shows mAP versus FPS for various object detection methods. It also shows the quick improvement of YOLOv2 compared to its first version

3.4 Training the model

The different aspect ratio and details between foreign and Egyptian license plates affected the detection of local license plates using pre-trained models. When testing these models, the accuracy was below 50% rendering them useless to our use case.

Due to privacy concerns, it was very hard to find online datasets that match the pattern of local license plates to train the model on. In cooperation with Logitime Egypt, we were able to obtain 1000 images from installed parking systems in Egypt and 300 scrapped images from the internet. The size of this dataset is very tiny compared to other datasets used in computer vision literature, however, the detector shows impressive results on the testset and novel samples captured by the group individuals. Figures 3.5(a) and 3.5(b) show random samples from the dataset used to train the model.

Figure 3.5(a)
A random sample from the images obtained from installed parking systems in cooperation with Logitime Egypt

Figure 3.5(b)
A random sample from scrapped images

Using Darknet - we used the official implementation by Alexey Bochkovskiy - we successfully completed the training on YOLOv4. The model accepts input samples of size 416 x 416 as recommended by paper authors. Darknet weights were then converted to tensorflow format and the model ran with a frame rate of 5 FPS on CPU. Frame rate was improved after using CUDA on NVIDIA's GTX-1050 Ti to approximately 15 FPS. Deploying the model on the same machine running the face module and both the front-end and backend servers was not possible due to insufficient resources. Fortunately, the authors offered a lightweight version of YOLOv4 called YOLOv4-tiny. YOLOv4-tiny utilizes the same technology of YOLOv4 but with fewer learnable parameters, less inference time and inference memory usage.

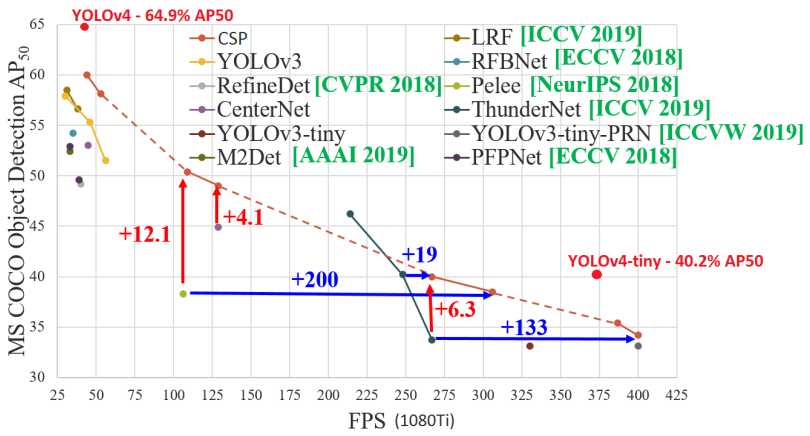

Figure 3.5
Performance metrics show that YOLOv4 tiny is roughly 8X as fast at inference time as YOLOv4 and roughly 2/3 as performant on MS COCO (a very hard dataset). On small custom detection tasks that are more tractable, you will see even less of a performance degradation. On the custom object detection model of ours, we see almost no degradation of performance as a result of decrease in model size. This image was published by Alexey Bochkovskiy on github.

Thanks to YOLO's augmentation methods, the model manages to generalize to novel input samples in almost every case. Although, one out of five frames can go undetected in some lighting conditions without majorly affecting the function of the detector.

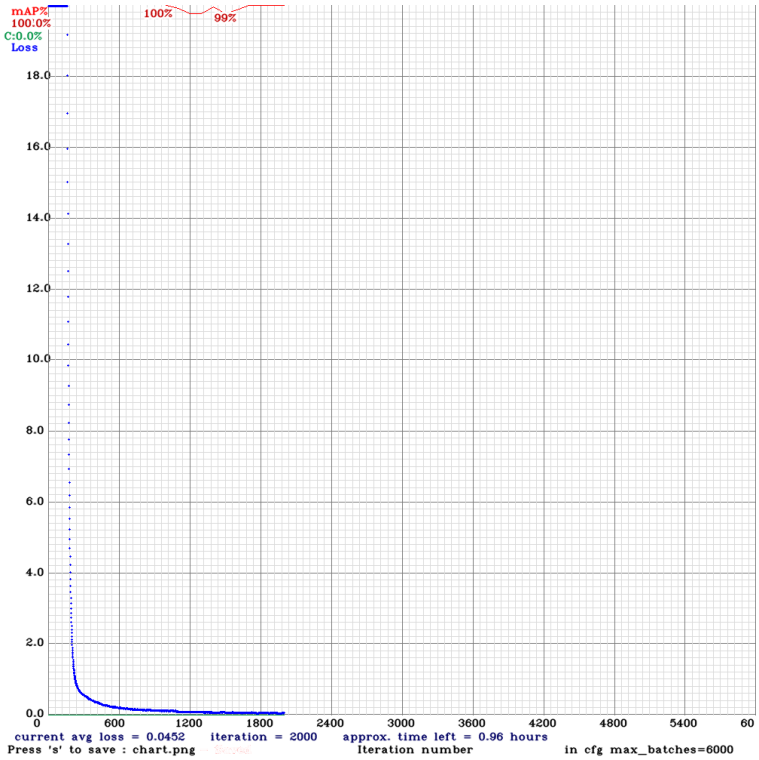

Figure 3.6 shows the mAP and model loss versus iteration count. Signs of quick convergence as well as overfitting are present in the graph. However, the detector behaves as expected most of the time

3.5 Assumptions for optimal OCR results

The distance between the camera and the front side of the car is one of the detrimental factors that affect the ability of the next component in the pipeline, a symbols extractors followed by a classifier, to function properly. As previously mentioned, the input video frame should be of dimensions 416 x 416 and in order to extract any usable information from the frame at hand the area of the license plate should be at least 2% of the frame area, i.e. of dimensions 68 x 34 taking into consideration that Egyptian license plates have an aspect ratio of 2:1.

This condition is met at a certain distance from the car front. Keeping a single device as the only source for sample videos captured at a resolution of 1280 x 720.[1] It's found that a distance of 1 meter is optimal for the rest of the pipeline to function as intended. It's also worth mentioning that a camera system has to be directly facing the car front, images that had skewed or rotated license plates were successfully detected by the detector but are not expected nor handled by the following components. This limitation stems from the adopted technique to extract symbols from images and the next section offers more in depth explanation.

Figure 3.7(a)
Detected license plate with the accepted conditions of dimensions and orientation. Note that slight tiltation do not really affect the symbols extraction stage

Figure 3.7(b)
Detected license plate that doesn't meet the conditions stated above. Despite being detected and classified correctly it's not possible to go forward with it

[1] Each frame of the input video is cropped and resized to reach the target input size

Chapter 4: Optical Character Recognition

4.1 OCR Pipeline

OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible. The subprocesses are:

- Preprocessing of the Image
- Text detection
- Character Segmentation
- Character Recognition
- Post Processing

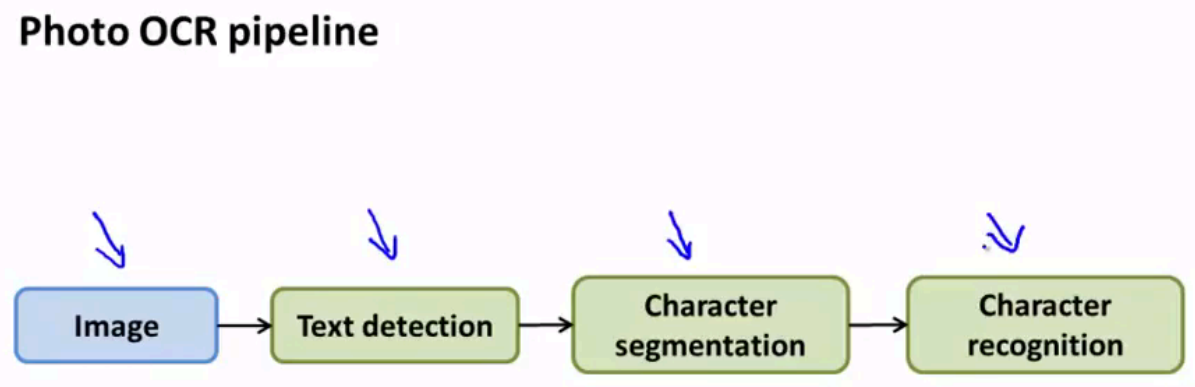

Figure 4.1 OCR pipeline

The sub-processes in the list above of course can differ, but these are roughly steps needed to approach automatic character recognition. OCR faces many challenges like complex backgrounds, diverse fonts and handwritten texts.

4.2 Image preprocessing

Before recognizing the individual characters, the plate obtained from the detector has to go through multiple preprocessing steps.

First, we discuss the objective of image preprocessing:

1. **Eliminate different lightning effects:**

An image captured in daylight will have different brightness and contrast levels than one taken at night in artificial lights. Though the installed hardware has a substantial role in ensuring as good lighting conditions as possible for the system, we also need to make sure the software module can deal with different lighting conditions.

2. **Crop the exact characters region:**

Each plate has a color code indicating the vehicle type. We are not interested in this part for now, but it can be a part of more future work. We have tried two approaches to crop the character region only.

   a. **Canny edge detection approach:**
   This approach gave 53% accuracy. It was greatly affected by the noise. Any sharp noise outside the plate (in the car frame) is detected as an edge.

Figure 4.2(a) Original image

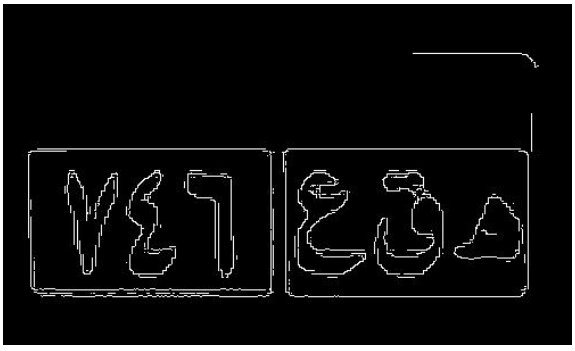

Figure 4.2(b) Edge detection by canny.

Figure 4.2(c) Output of edge detection.

Figure 4.2(d) Characters region.

Figure(4.2) Edge detection by canny.

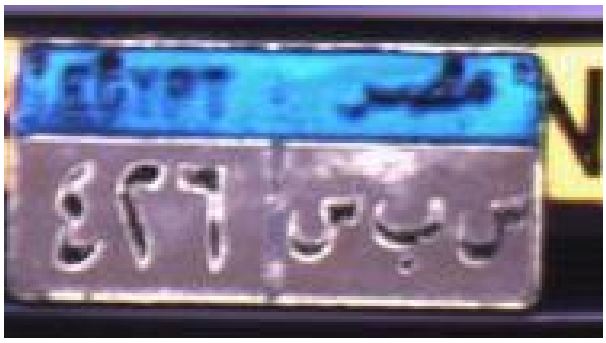

Figure 4.3(a) Original image.

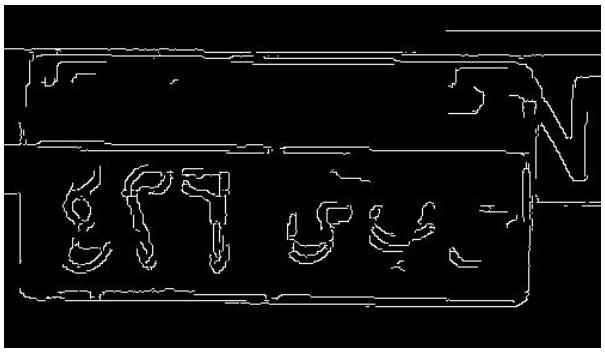

Figure 4.3(b) How noise affects edge detection.

Figure 4.3(c) Output of edge detection under noise.

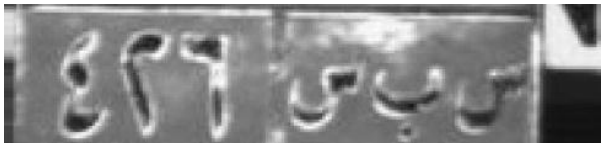

Figure 4.3(d) Characters region under noise.

Figure(4.3) How noise affects edge detection.

b. Find contours approach:

This approach gave more accurate results. We tested this approach on 300 frames. We need to mention that it didn't work on all frames and far frames were discarded. Up to 95% of the undiscarded frames were cropped correctly.

Figure 4.4(a) Original image.

Figure 4.4(b) Applying find contours.

Figure 4.4(c) Find contours output.

3. **Clean the noise:**

The detected plate won't always be perfect. It can be distorted, dirty or has partially erased characters. Applying back-to-back operations of opening (erosion, then dilation) and closing (dilation, then erosion) can be useful in sharpening edges.

Figure 4.5 Partially erased characters before and after opening and closing operations.

4.3 Segmentation

Segmentation as defined in the original article published in 2013 [6] is one of the most important phases of the OCR system. It is an intermediate step between detecting the plate and OCRing the character. Simply, segmentation is applied on the ROI obtained from the previous step (image preprocessing) to break the whole image into subparts to process them further. For our application, we're going to use the Vertical Project Profile technique.

Vertical Project Profile

Once the colored image is converted to the binary image, only black and white pixels are present in the image. In a binary image, pixels representing useful information are called Foreground pixels, and the pixels that are not foreground pixels are called Background pixels. We choose whether a foreground pixel should be white or black while binarizing the image. We either apply this projection horizontally or vertically.

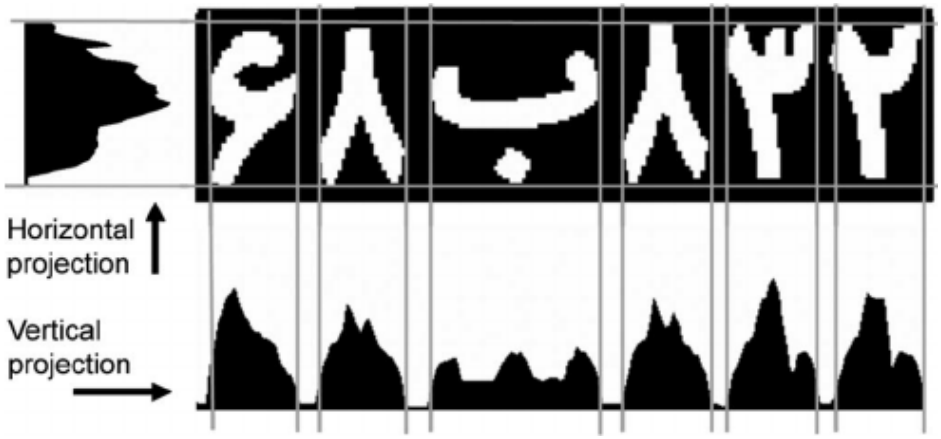

Figure 4.6 example of horizontal and vertical projections

To separate license plate characters, we only care for the vertical histogram projection. In this method, we count the No.of foreground pixels along the image columns. The result is an array of the size equal to No.of columns in the image (Width of the image).

In the above image:

- Foreground pixels are black pixels, and background pixels are white.
- Columns that represent the text have high No.of foreground pixels, which correspond to higher peaks in the histogram.
- Columns that represent the gaps in-between the words have high No.of background pixels, which correspond to lower peaks in the histogram.
- Columns that correspond to lower peaks in the histogram can be selected as the segmenting lines to separate the words.

Results

Figure 4.7(a) original image

Figure 4.7(b) crop & resize

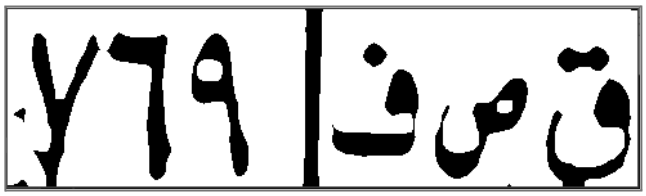

Figure 4.7(c) Blur, Threshold, Dilation and binarization

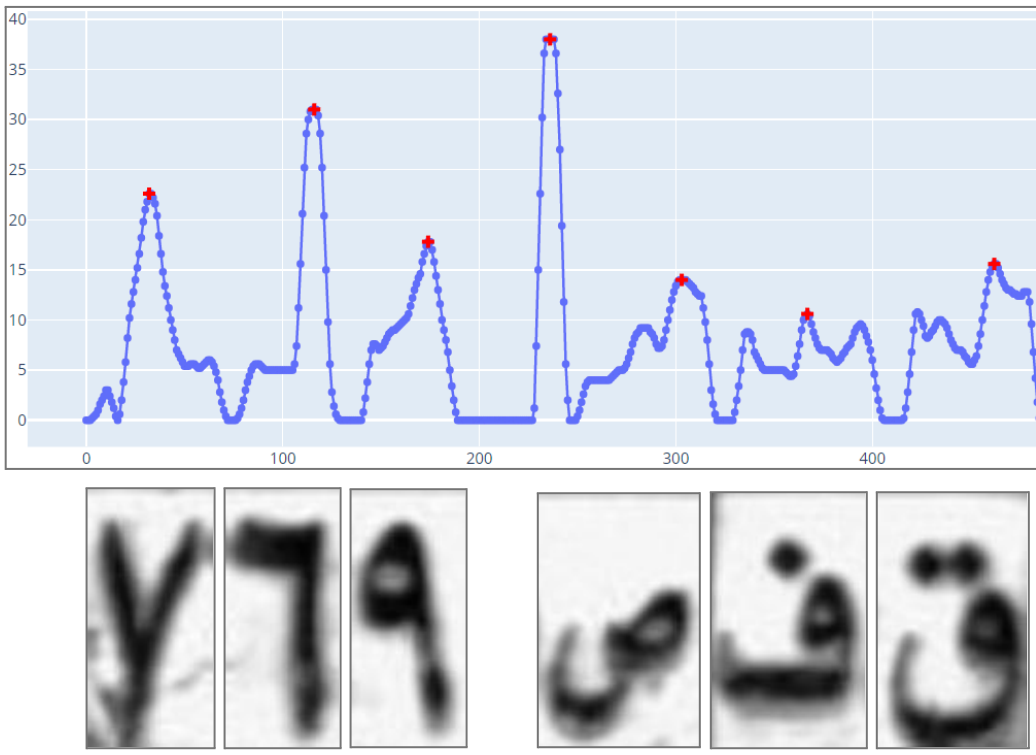

Figure 4.7(d) Finally, Draw vertical projection and crop the original image based on the peaks.

We tested the segmentation on 129 different plates with different angels in various lightning conditions. These plates contained 776 characters. Accuracy of segmentation was 92% with a chance of 4% that a character got recognised twice. This problem can be solved with further smoothing to the histogram.

4.4 Character Recognition

4.4.1 Tesseract

We started our work by testing a widely used model for OCR text documents, Tesseract OCR [5]. Tesseract gained popularity as there weren't too many free and powerful OCR alternatives on the market for the longest time. Tesseract began as a Ph.D. research project in HP Labs, Bristol. It was developed by HP between 1984 and 1994. In 2005 HP released Tesseract as an open-source software. Since 2006 it has been developed by Google. Tesseract is a lightweight library that can recognize more than 100 languages, including Arabic. Being an open-source project, Tesseract promised an easy-to-retrain process, which by experiment, it wasn't entirely true.

Results
Using Tesseract to recognise individual Arabic characters as in the license plates, gave 60% accuracy for letters and 73% for Indic-Arabic numbers. But this accuracy was obtained by preprocessing each plate individually to clean the noise as much as possible, while automating the process significantly decreases the accuracy and makes the tesseract unreliable.

Figure 4.8 results of tesseract on individual Arabic characters

Limitations of tesseract
Tesseract works best when there is a clean segmentation of the foreground text from the background such as textbooks which tesseract is mainly trained for. The better the image quality (size, contrast, lightning) the better the recognition result. It requires a bit of preprocessing to improve the OCR results, images need to be scaled appropriately, have as much image contrast as possible, and the text must be horizontally aligned. In practice, we can't guarantee to get an image clean enough for tesseract to work on.

Tesseract limitations summed in the list:

- Doesn't do well with images affected by distorted perspective and complex background.
- It may find gibberish and report this as OCR output.
- It is not always good at analyzing the natural reading order of documents. For example, it may fail to recognize that a document contains separate characters, and may try to join these characters.

This disappointing result shifted our decision to build an OCR model from scratch customized for our problem of recognising individual Arabic characters.

4.4.2 Classifier

The success in building a functional classifier capable of predicting the symbol class from segmented images depends on two factors. First, an adequate dataset that contains diverse examples and suitable distribution in terms of number of samples per each class. Second, a good model architecture. We'll discuss these two factors thoroughly in the section below

4.4.2.1 Dataset collection and preparation

In our application we are only interested in 26 characters:

- 17 letters: أ ب ج د ر س ص ط ع ف ق ل م ن هـ و ى
- 9 numbers: ١٢٣٤٥٦٧٨٩

At first we used a very tiny dataset of 1,300 sample images. Part of this dataset was manually extracted from the dataset used to train the license plate detector and the other part was obtained from the segmentation part discussed above. The reader might be wondering why we didn't use the entire dataset used in training the license plate detector to obtain sample images for this part. Most of the images used to train the license plate detector are not suitable to run them through the segmentation component to extract useful data.

Afterwards, a dataset of approximately 14,000 sample images was provided to us by a colleague of ours. This dataset came along with another idea to increase the number of training samples – that is data augmentation. Each sample image underwent 9 augmentation methods. These methods included random image rotation, distortion, brightness, contrast and saturation manipulation. The augmented dataset contained 53,437 sample images.

4.4.2.2 Building the classifier model

We used two different classifiers throughout the development of this project. A SVM was used first with the initial dataset. When a larger dataset was obtained, we decided to train a CNN to better handle the problem.

SVM

This model worked fine whenever the prediction was made on a sample image that the SMV has seen before, but when testing how the model generalizes, it didn't give very promising results. This is due to the tiny dataset it was trained on.

CNN classifier

Model architecture was inspired by other CNN classifiers used to classify handwritten greek letters. Here's a quick overview on the architecture

1. An input image of dimensions 75 x 25 is fed to the first convolutional layer
2. The first convolutional layer has a kernel of 3 x 3 and 32 filters. Batch normalization is applied after the first layer
3. The second convolutional layer has a kernel of 3 x 3 and 64 filters. Batch normalization and 2D Max Pooling are applied after the second layer
4. The third convolutional layer has a kernel of 4 x 4 and 128 filters. Batch normalization and 2D Max Pooling are applied after the third layer
5. Output is flattened and inputted to a fully connected dense layer of 128 nodes followed by 26 nodes that represent the final prediction.
6. The nonlinearity used throughout the entire network is *tanh* except for the final dense layer that uses softmax.
7. L1 and L2 regularization were applied on the last convolutional layer and the first dense one

The model has a total of 940,954 trainable parameters. A schematic diagram visualizing model architecture is shown below.

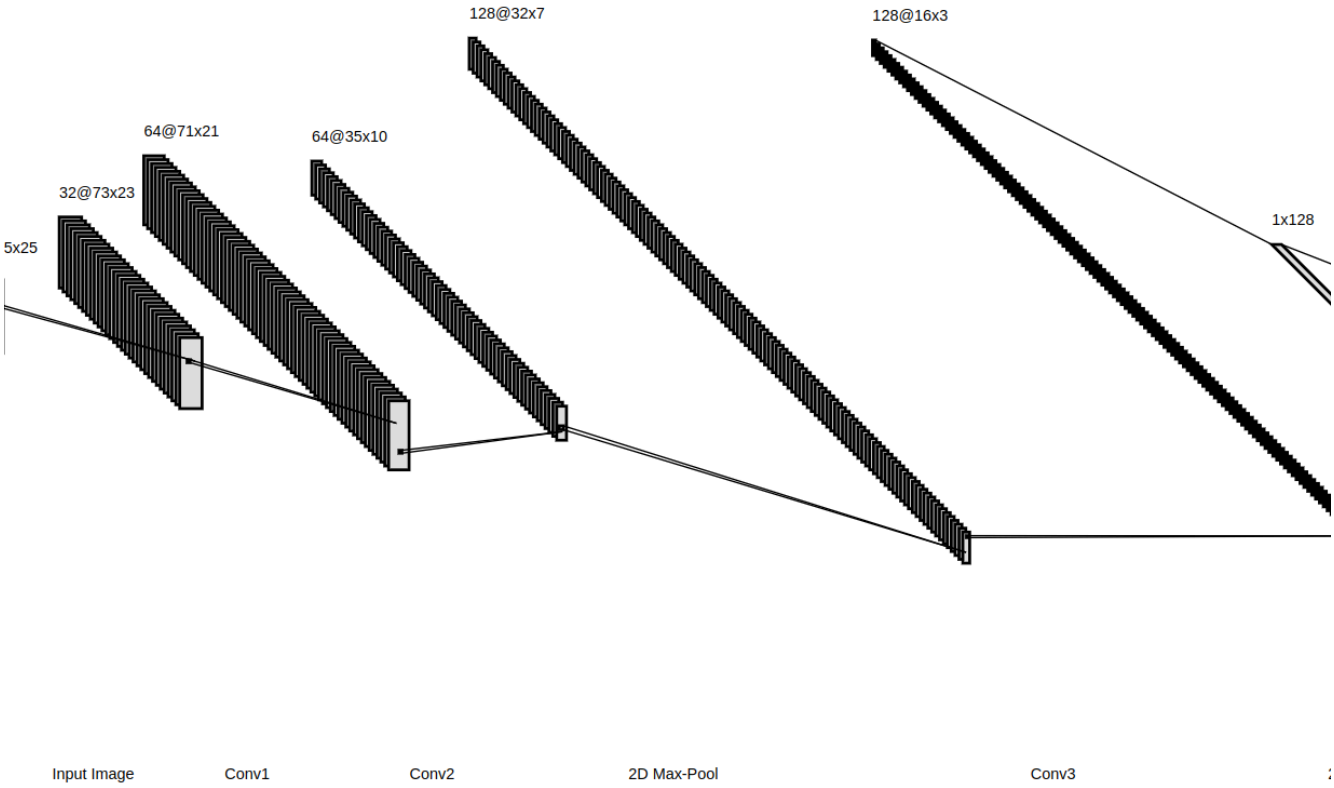

Figure 4.9
CNN model architecture used

4.4.2.3 Training results

The model was trained for 10 epochs. In the two training metrics visualized below, you will notice the following:

- Training loss decreases greatly on the first epoch only.
- Validation loss diverges at the end of the 7th epoch as a sign of overfitting.
- After the 7th epoch, validation accuracy drops greatly as a sign of overfitting.

We used the weights produced after the end of the 7th epoch. On a test set the model shows a fair accuracy level of 85%.

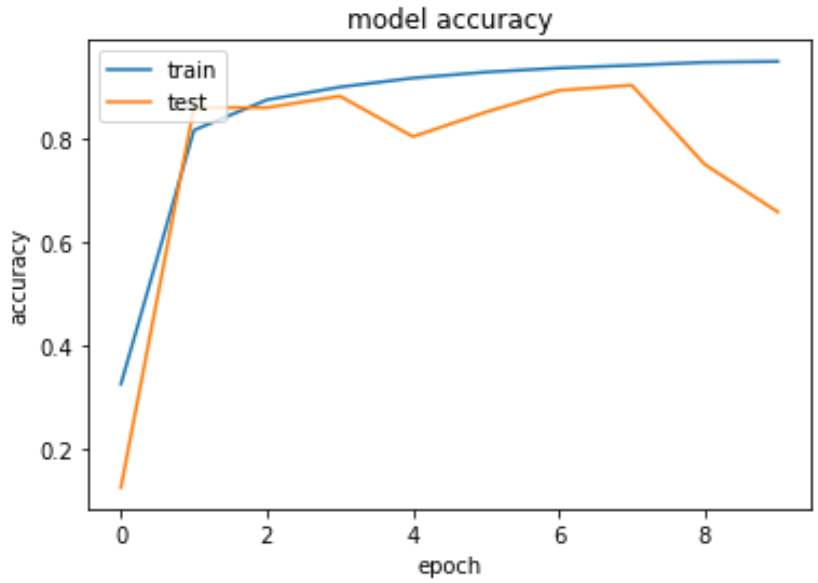

Figure 4.10(a) shows training and validation loss in blue and orange lines respectively

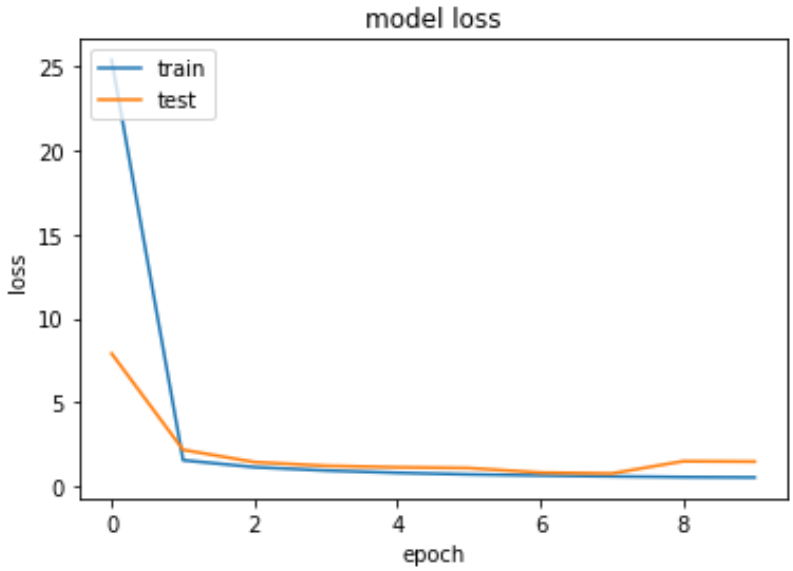

Figure 4.10(b) shows training and validation accuracy in blue and orange lines respectively

Chapter 5: Our website

5.1 Why choose a web application over a desktop one?

- For the user, it saves the hassle of downloading and installing updates as we can add new features or fix bugs without the user even noticing. He will always open the website to find the latest stable version.
- Web applications are not reliant on the hardware or system specifications to run.
- Cross-platform availability and mobility: it gives us the convenience to switch to a mobile app.
- Light on system resources: web services consume less processing power compared to desktop apps.
- A desktop app is downloaded once, which means there's no waiting for the web pages to load. However, using Single Page Approach compensated for this lag.

5.2 Frontend

In this part we will discuss components and stack of our front-end part. We will talk about the following:

- Tech stack.
- Admin panel architecture.
- High level architecture.

Tech stack

Our tech stack was chosen to be flexible and reliable. We are using React 17 with Material UI from Google. We are also using Axios for asynchronous requests over RESTful API. This allows us to build a continuous experience without reloading pages and pressuring the server.

Admin pages

1. Profile
    - Contains profile information for the current user as his Email, Role, etc..
2. System Users
    - Contains a list of all system users and the ability to add new users.
3. Customers
    - Contains a list of registered customers and ability to register new ones.
4. Cars
    - Contains a list of all cars registered in the system with their information and ability to register new cars.
5. Control
    - Includes adding cameras and hardware configuration required to connect with each of them.

5.2.1 High level architecture

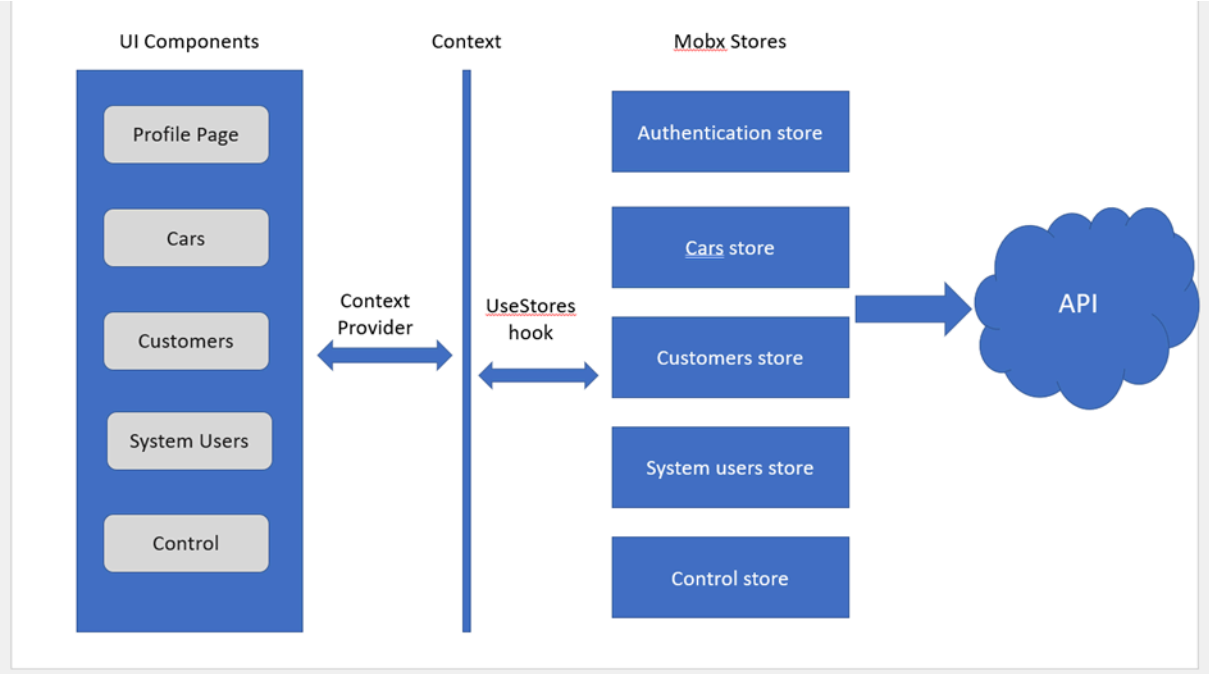

Figure 5.1 High level architecture

According to the previous figure our app is separated into different UI components to have our ui decoupled from business logic.

Business logic in our app is centered in the “Mobx store” which provides reactivity to our components based on any changes in the data, also responsible for communication between the UI and the services where requests to the API are made.

Till the moment we have 5 stores as follows:

1. Authentication Store
    - Responsible for handling the logic for Login, Logout, Signup, users’ data.
2. Car’s store
    - Handling the current list of cars and any logic related to it.
3. Customers store
    - Handling the current list of customers and any logic related to it.
4. System users store
    - Handling the current list of system users and any logic related to it.
5. Control store
    - Handling all hardware connections and configs (Cameras).

5.2.2 Interface

- **System User Sign-In Interface**

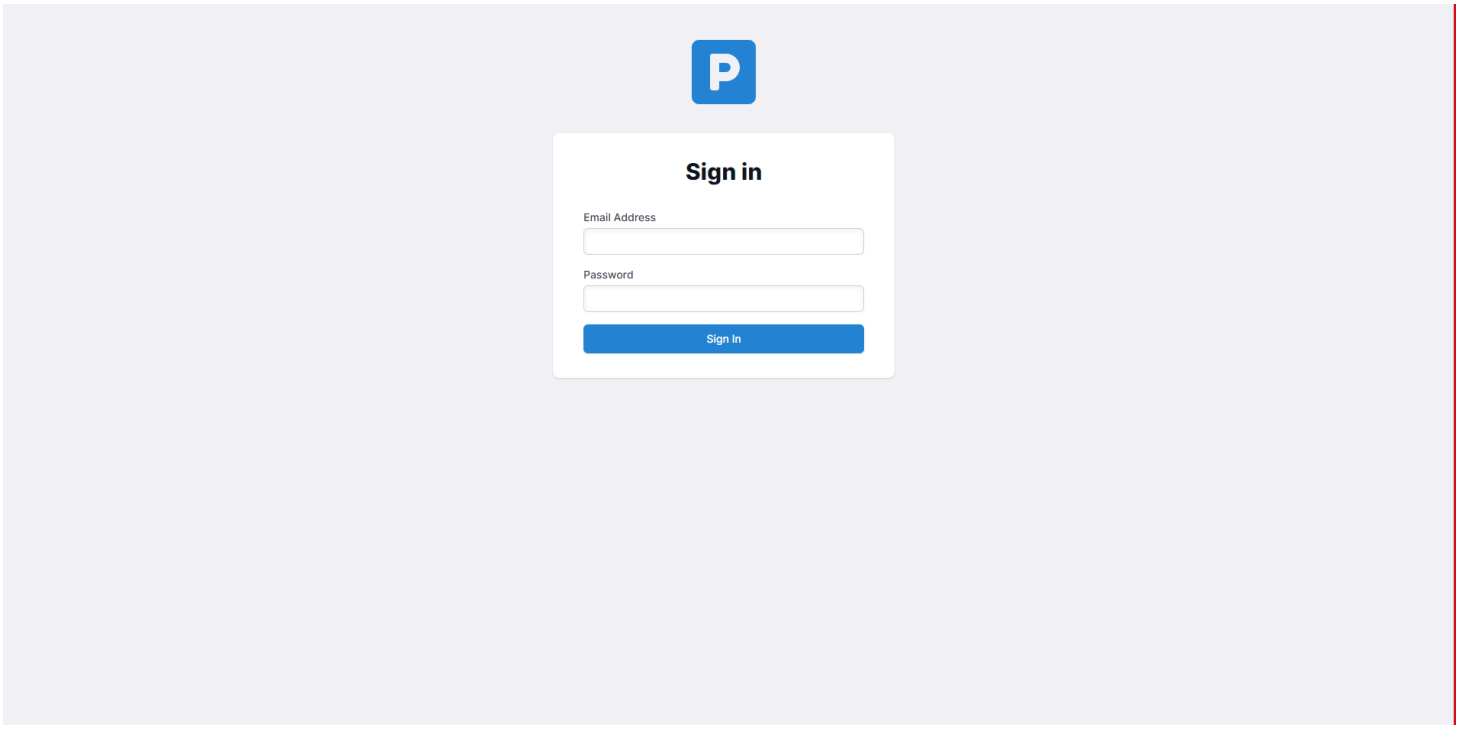

Figure 5.2 SystemUser Sign-In Page

A system user who manages and operates the Parking System should provide his credentials to sign in. Therefore, he can have access to the platform functionality.

- **System User Profile interface**

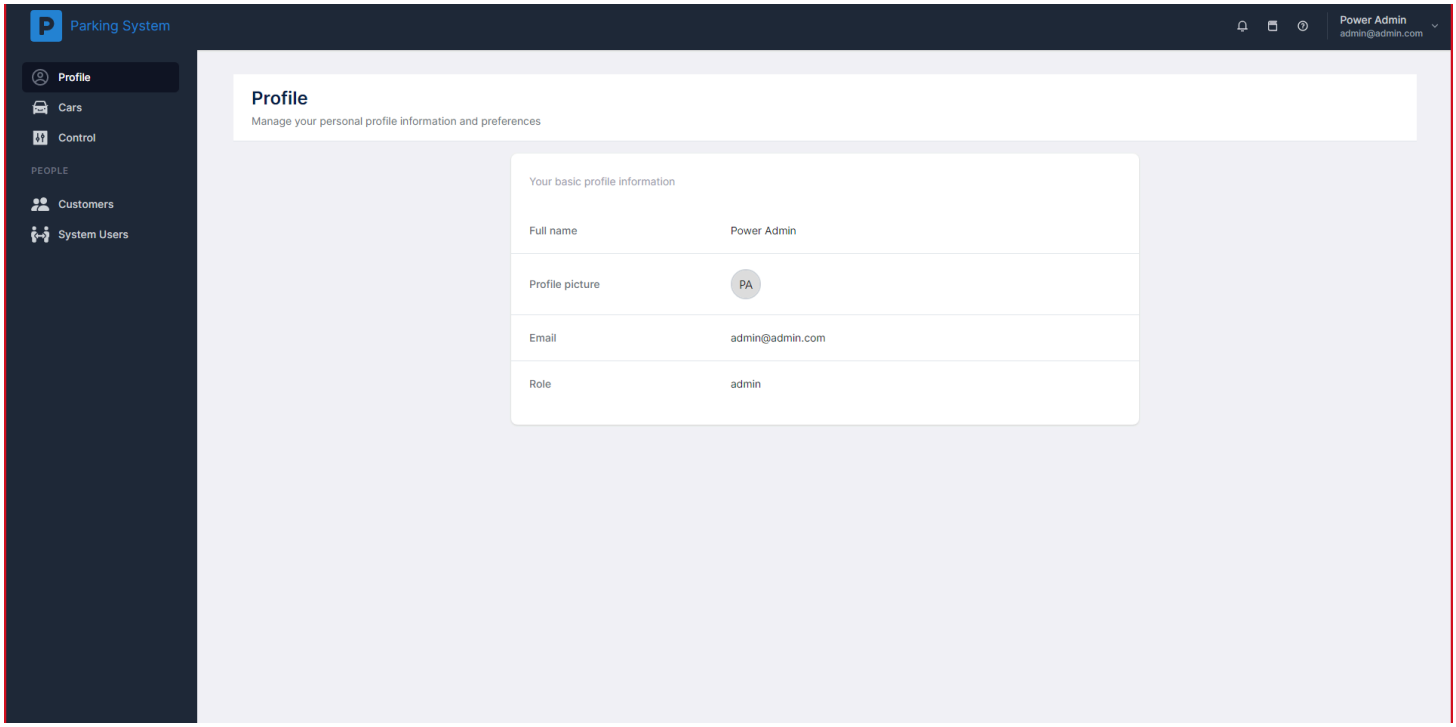

Figure 5.3

This interface shows details of a System User (the admin who manages and operates the Parking System).

- **SystemUser interface**

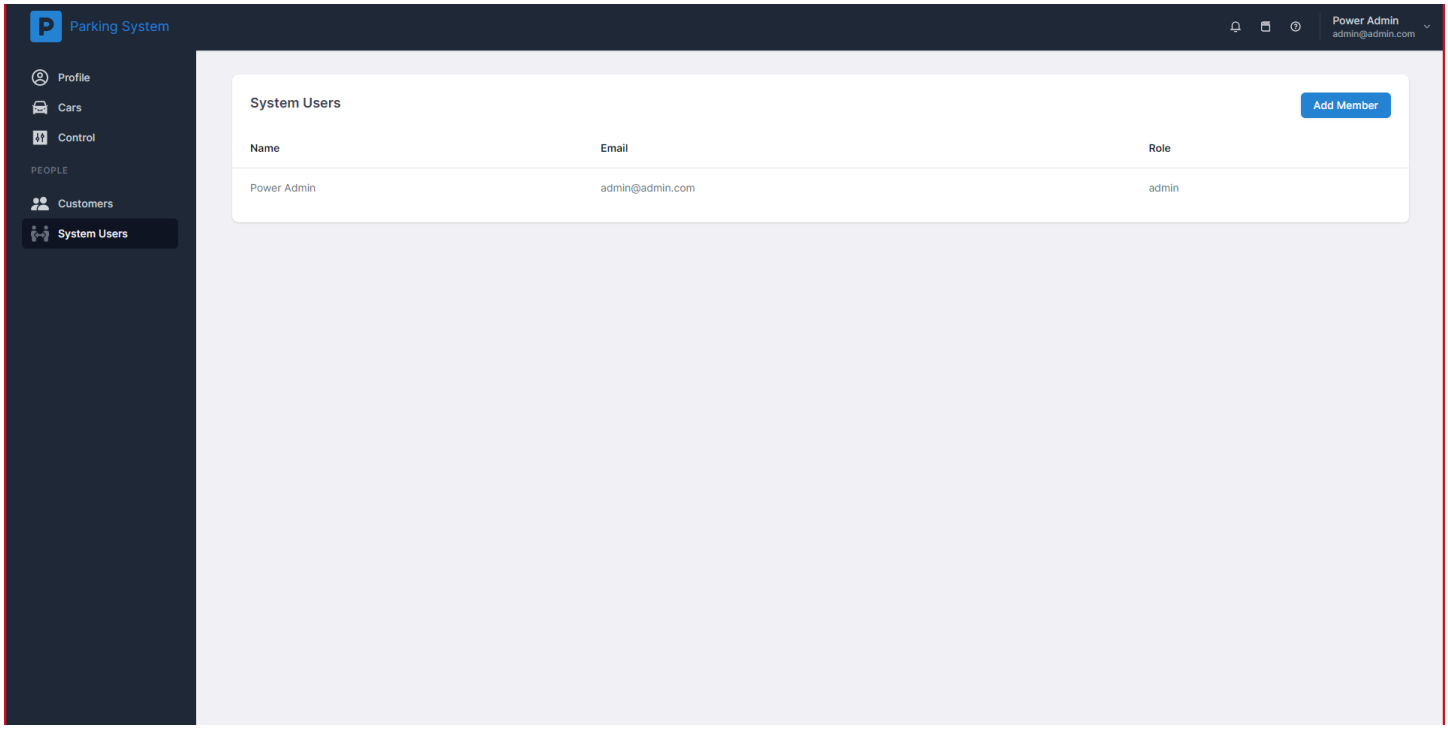

Figure 5.4

This interface shows the list of all System Users (admins who manage and operate the parking system), that are held by the system.

- **Add new customer Interface**

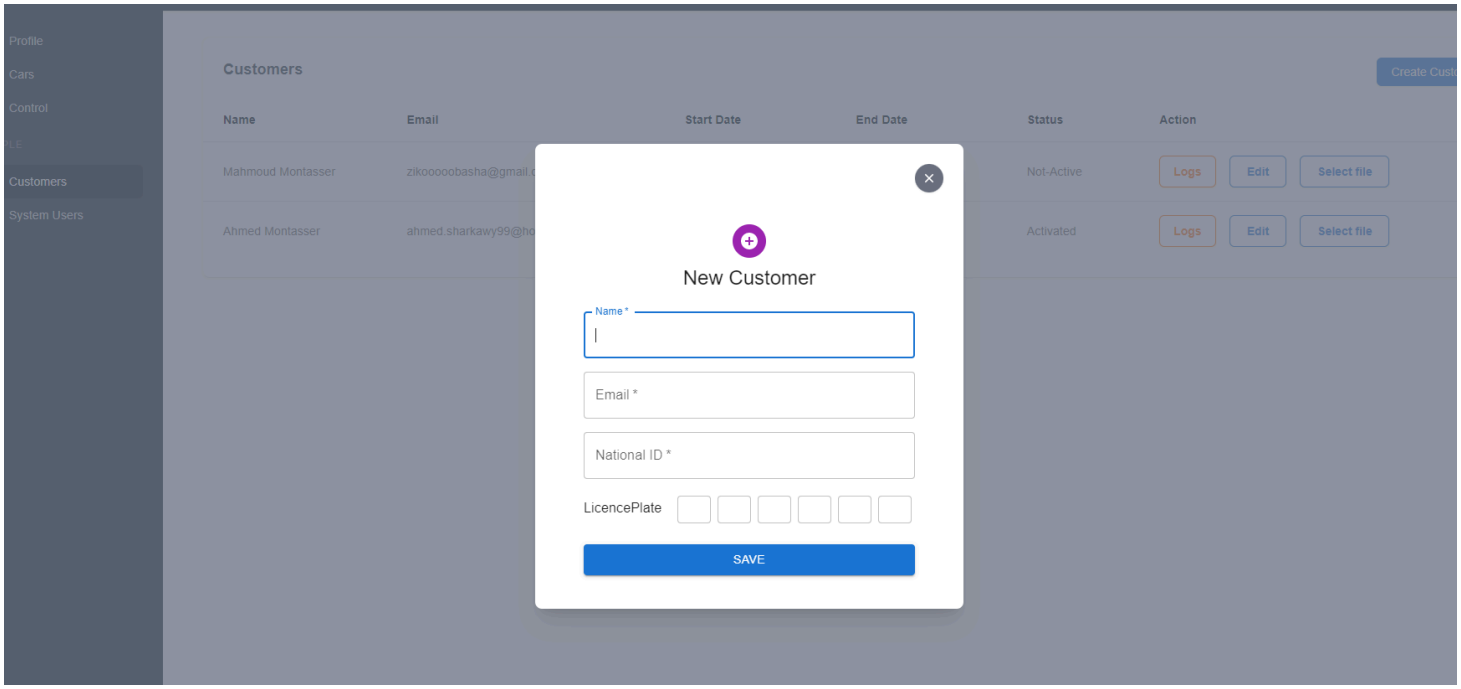

Figure 5.5

A System User (who manages and operates the parking system) can add new customers through this interface. To add a new customer, the customer's name, email, national ID, and the list of license plates of vehicles should be provided.

- **Customers list**

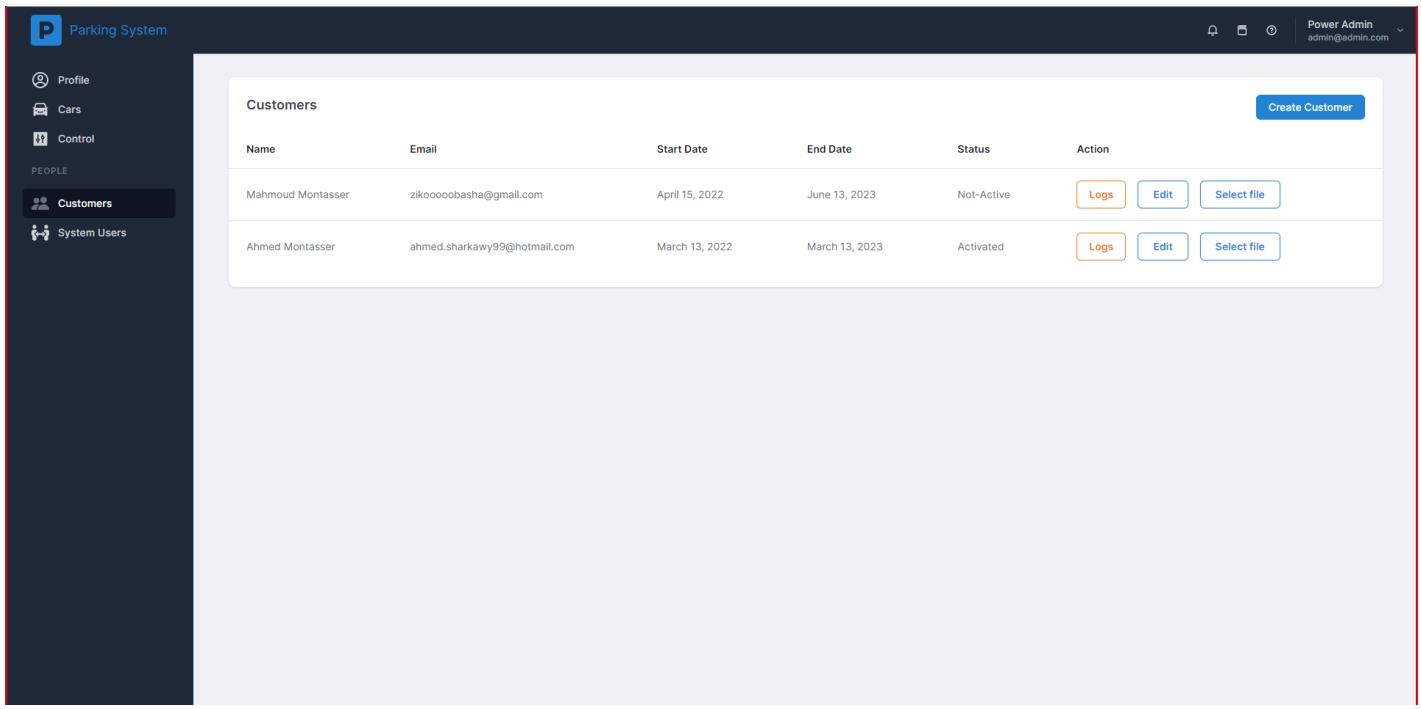

Figure 5.6

This Interface shows the list of customers held by the system.

- **Add video of new customer**

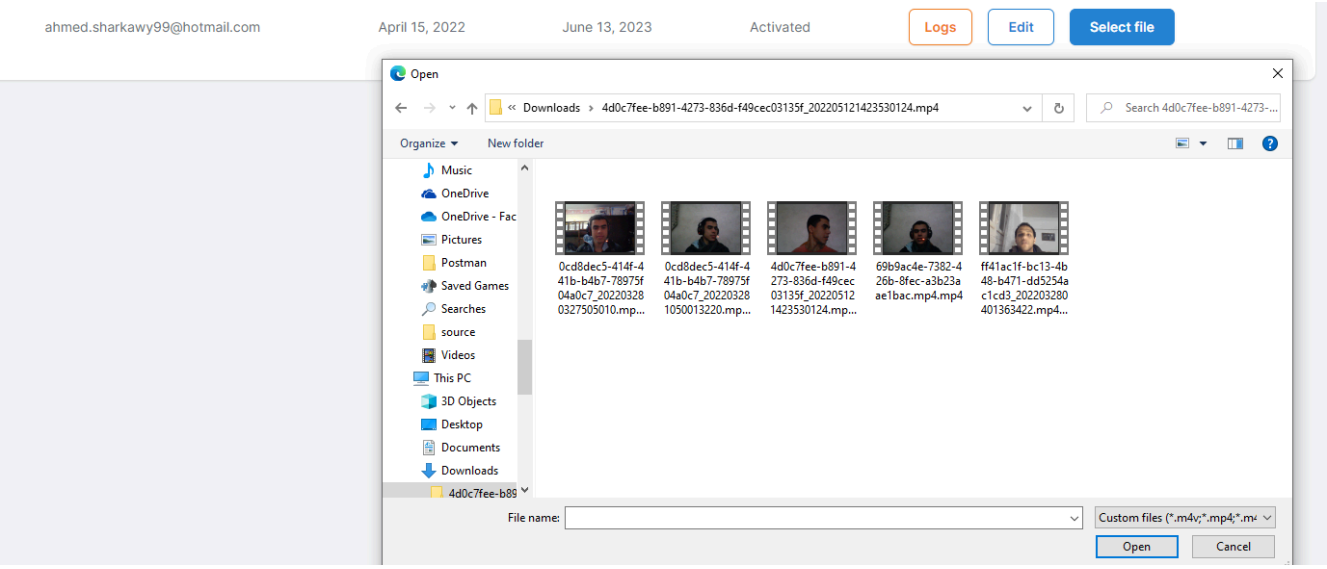

Figure 5.7

By pressing the "Select file" button displayed for each customer in the customers list, we can select a video to train the model on.

- **Add new vehicle**

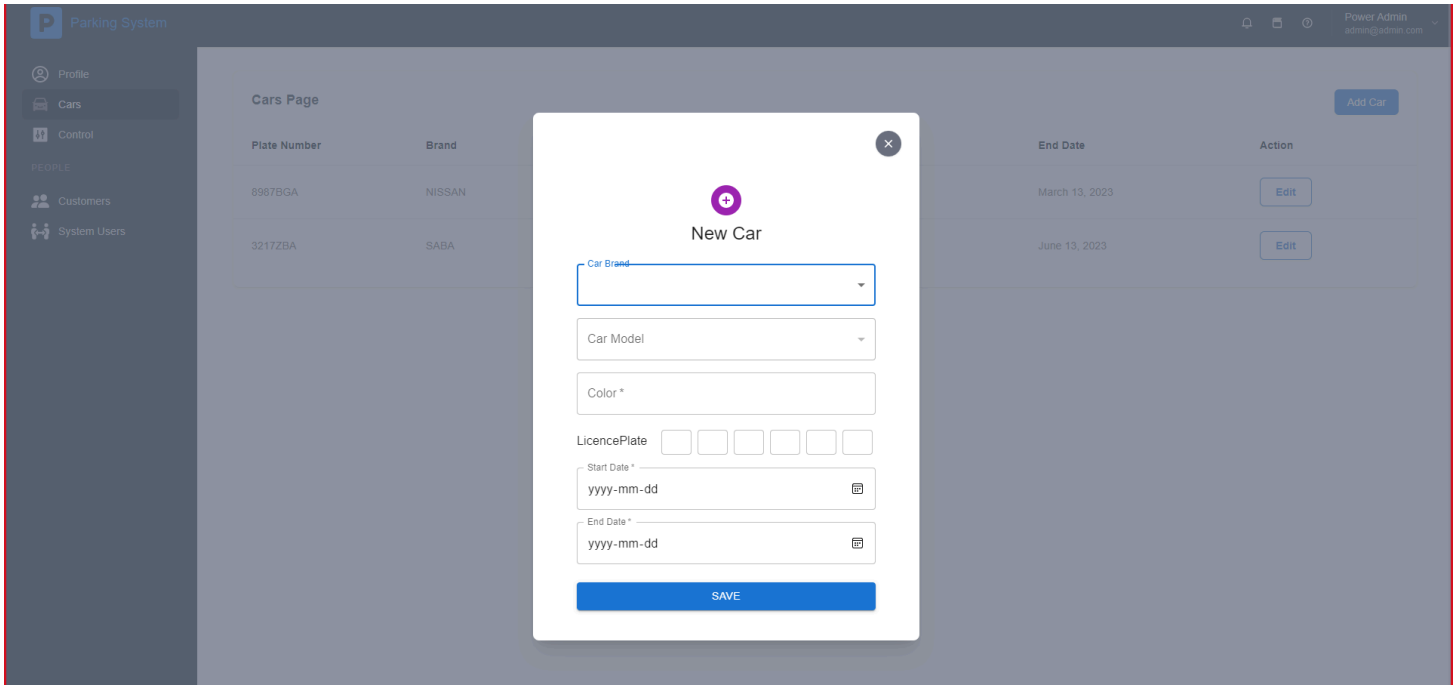

Figure 5.8

We can add a new vehicle using this form. The car brand, model (sub-brand), color, license plate, and start and end subscription dates must be provided.

- **Vehicles list**

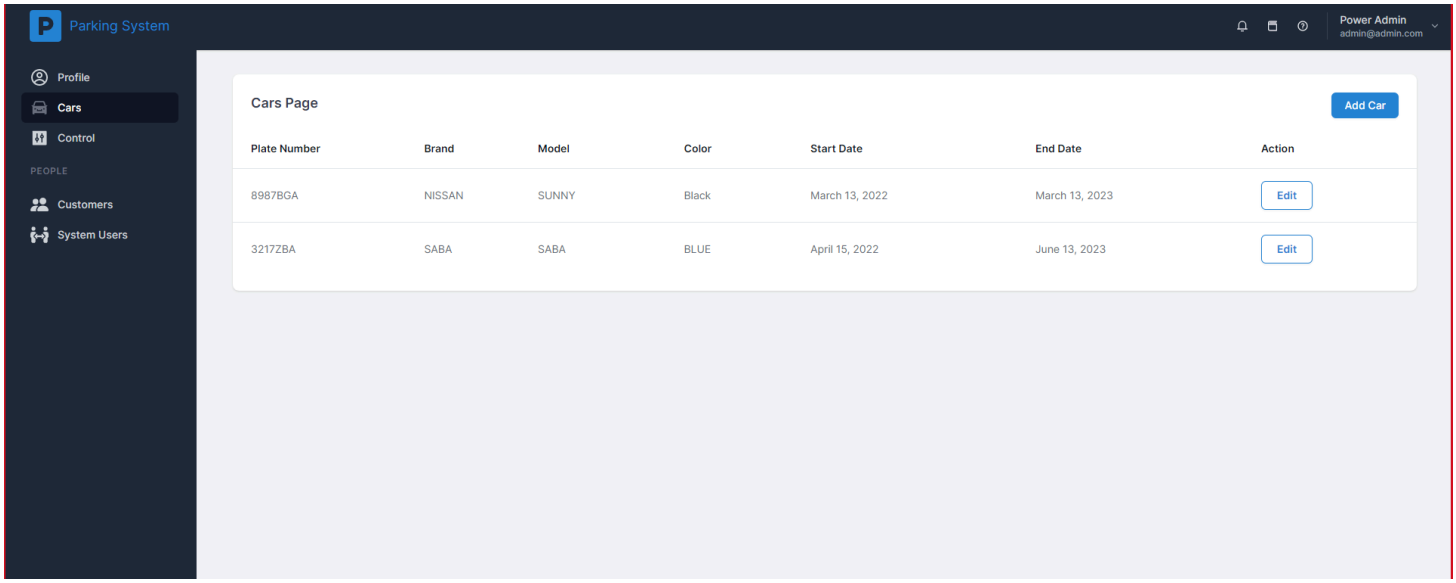

Figure 5.9

This Interface shows the list of vehicles held by the system.

- **Gate Control interface**

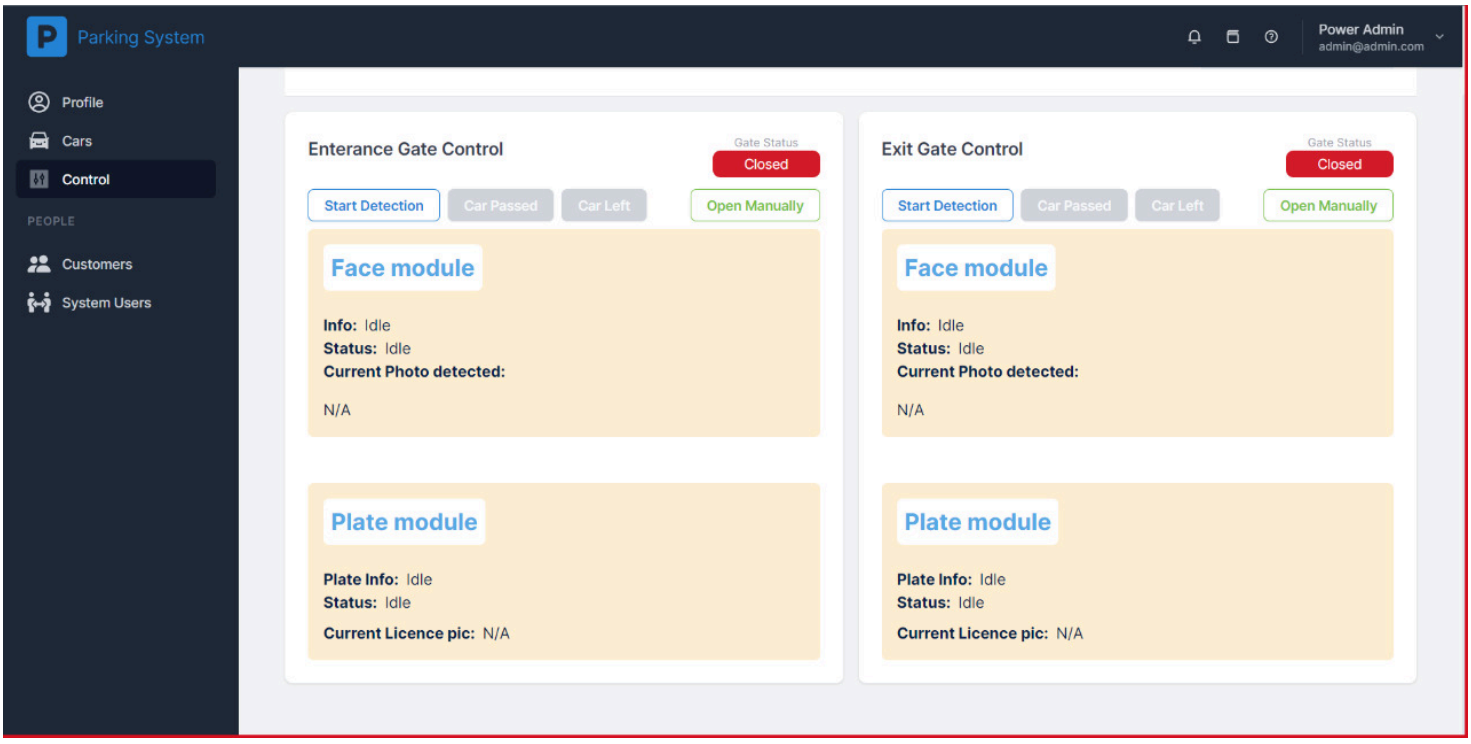

Figure 5.10

This interface has the simulation for the entrance or exit process performed.

- **Select plate video**

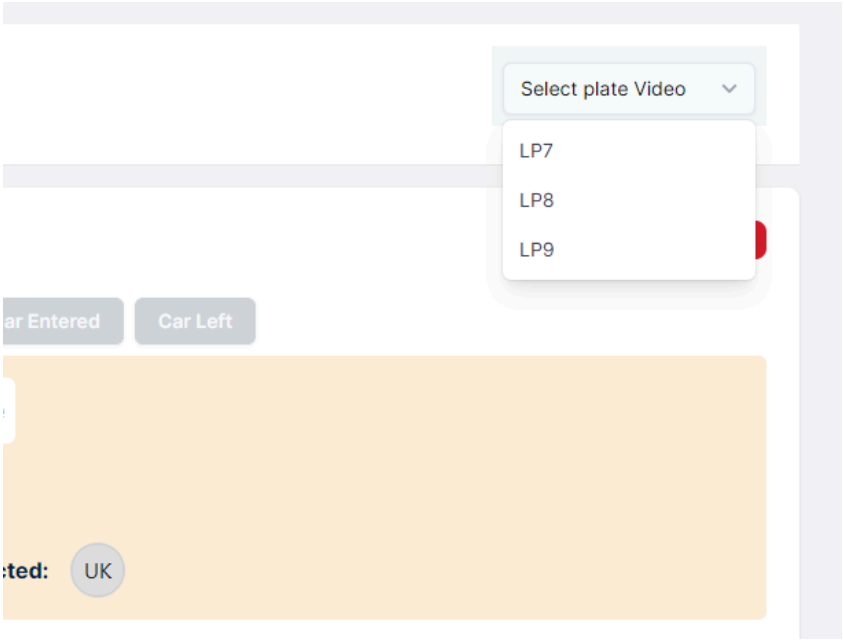

Figure 5.11

This dropping list is to simulate when the car is at the entrance or exit. We select a recorded video of the front or back part of a car showing the license plate region.

- **Logs**

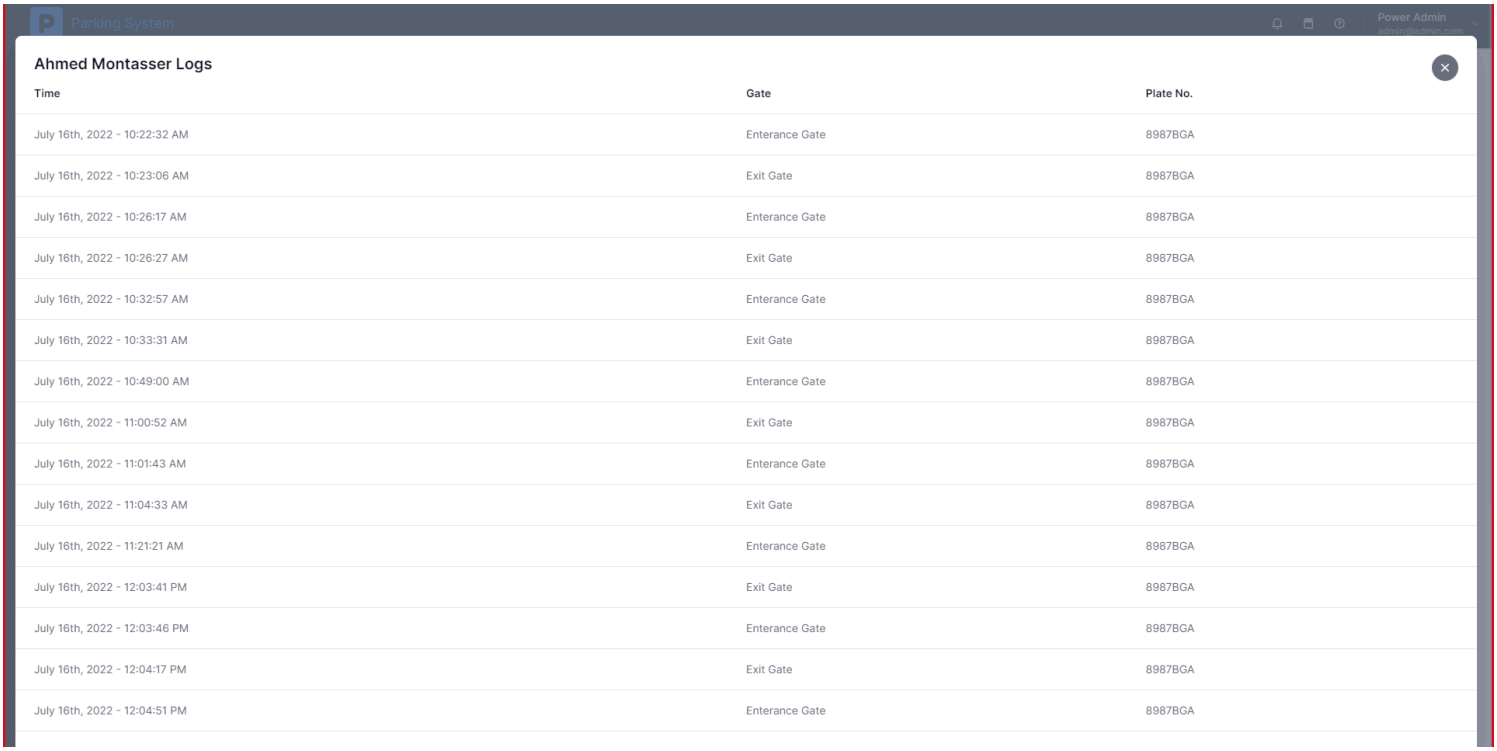

Figure 5.12

On the Logs page, we can see all previous logs for any user. It contains the date and time of his visits, which gate he used, and his plate number.

5.3 Backend

In this part we will have a look at the backend part.
There are 2 main parts in the Backend:

1. Web Server
2. Database Server

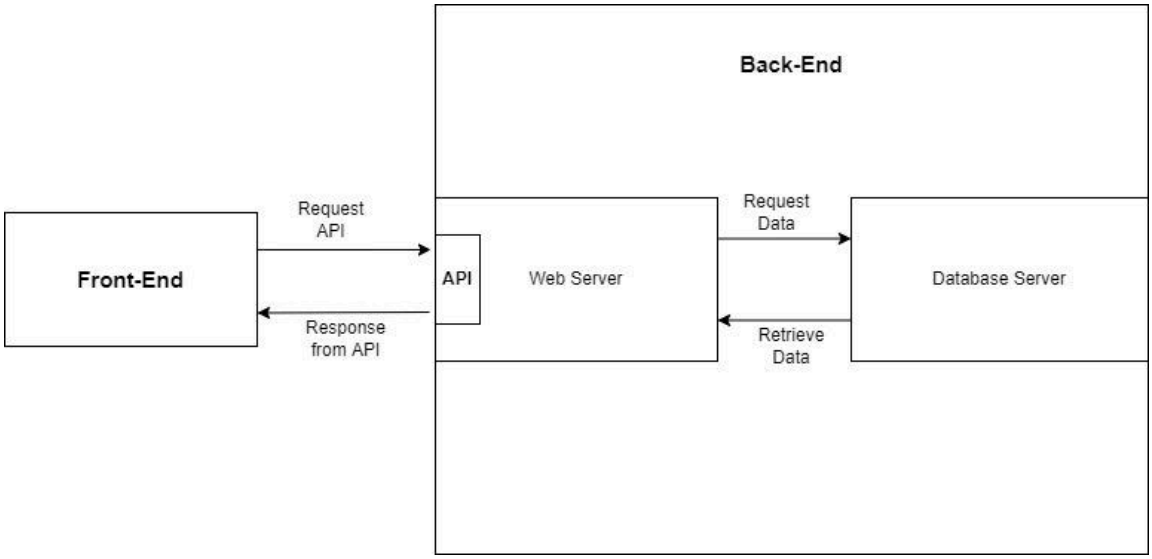

Figure 5.13 Frontend, backend, APIs

5.3.1 Web Server

This part is developed using ASP.NET Core (Web API). Since the front-end deals with the backend through the API.

APIs are classified into controllers according to the entity it is dealing with.
For instance , System Users controller is the part to deal with all that concerns System Users Entity such as:

- https://<HOST-NAME>/api/SystemUsers/login: is used to login System Users Account.
- https://<HOST-NAME>/api/SystemUsers/me: is used to show data about logged accounts.

Some APIs may have access to the database to do its functionality such as Adding a new long-term Participant. This is done through Entity Framework as it provides an easy way and already-built methods to deal with the database and retrieve data from there instead of writing SQL Operations. This had some advantages :

1. Prevent some issues like SQL injection attacks.
2. Easy way for development among peers since by using the classical way it was necessary to share more documents -SQL Scripts- holding SQL operations so that it is used at every local machine.

5.3.2 Database Server

This part is implemented using Microsoft SQL Server Database Management System as it is the most compatible with C# and ASP.Net Core.

- **ERD Diagram**

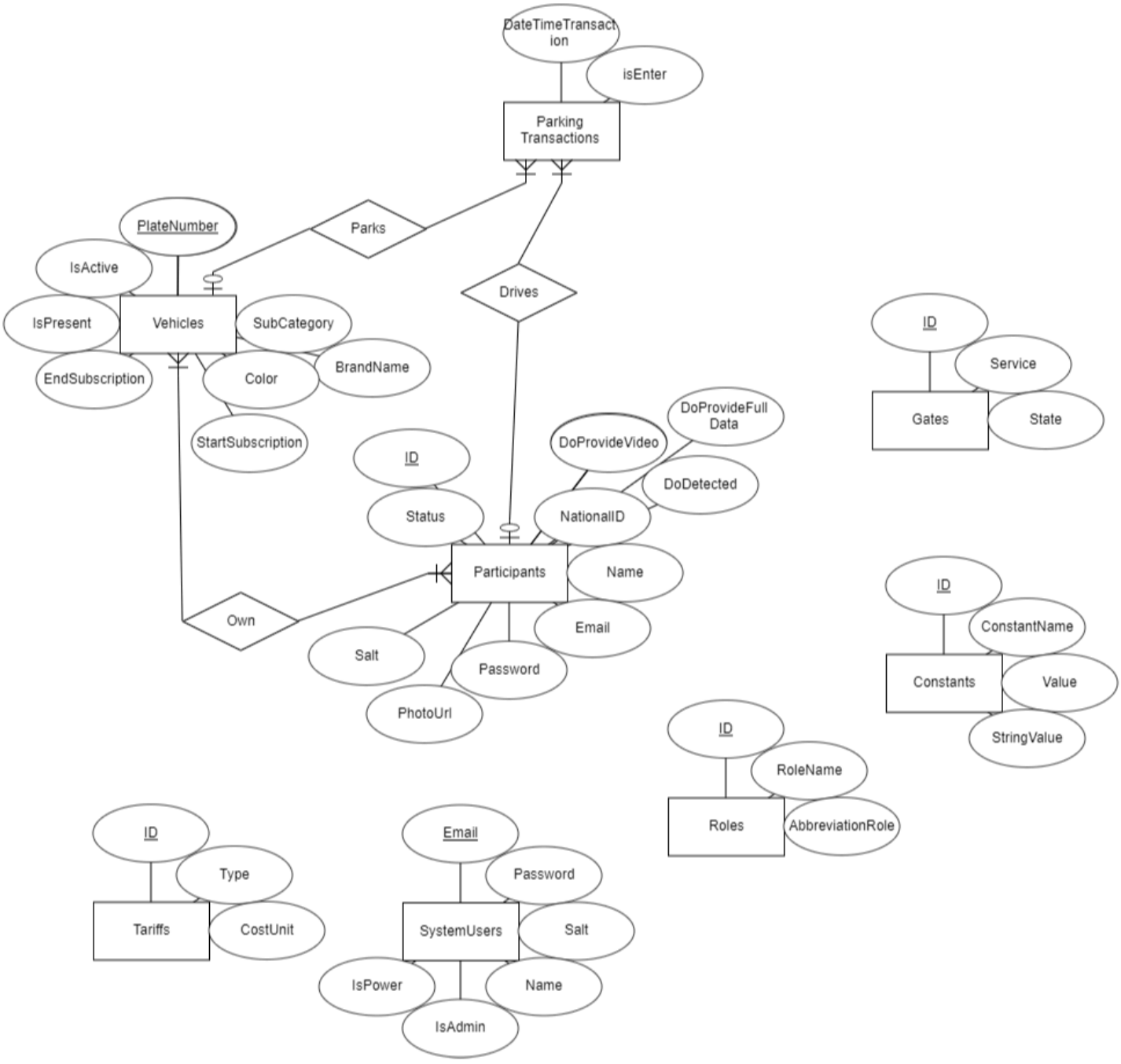

Figure 5.14 Entity-Relationship Diagram

- **Relational Scheme**

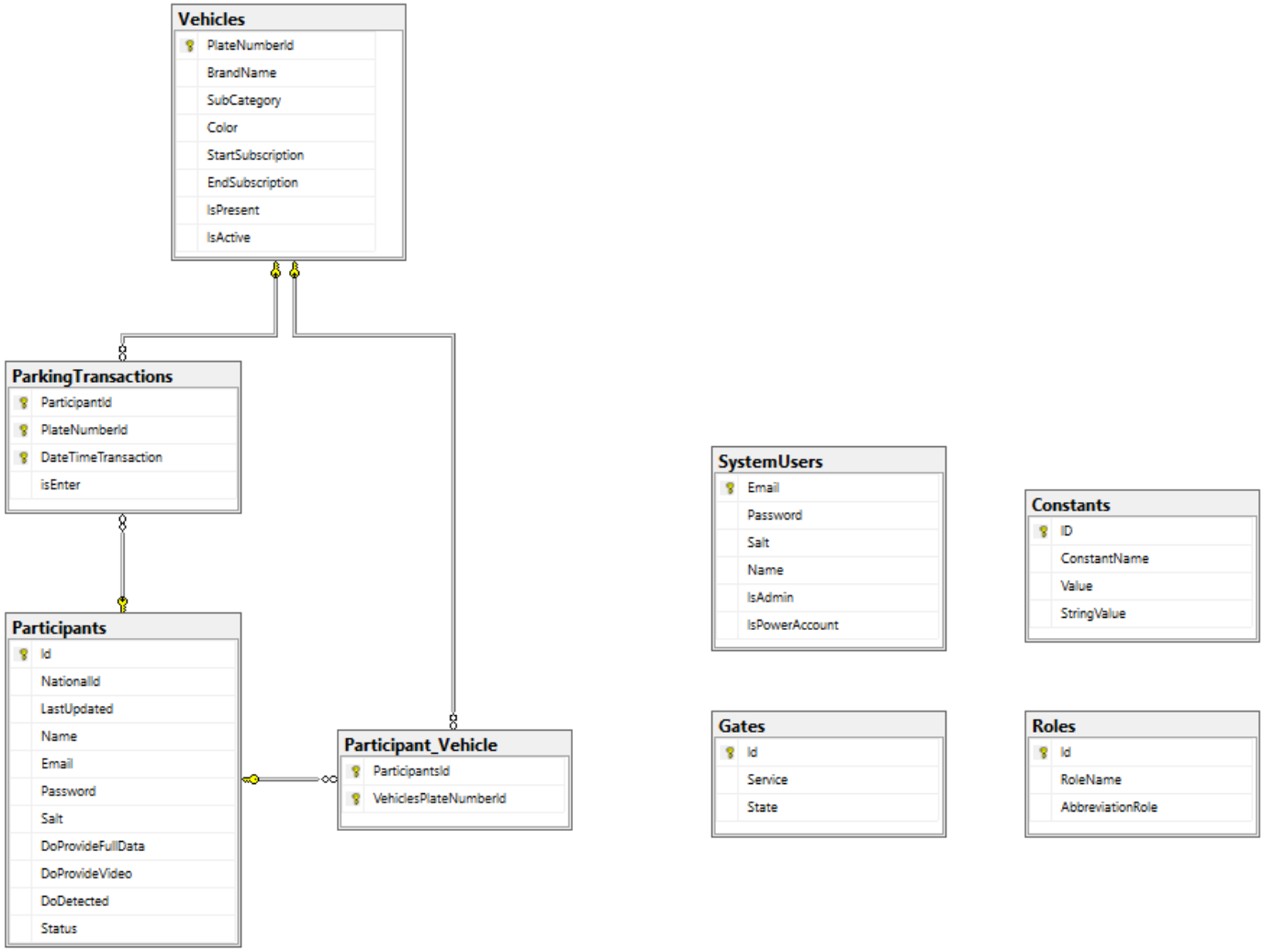

Figure 5.15 Relational Diagram

5.3.3 Security

- Using Password + Salt [Used in Authentication]
    1) At the SignUp Event, A random Salt is generated and saved as Plain in Database.
    2) This salt is padded to the end of the password then the yield is hashed and Hash is stored in the database.
    3) At every Login Event, The Password is provided as plain.
    4) The salt is retrieved from the Database using the ID of the User.
    5) The salt is padded to the Password provided , then the yield is hashed.
    6) The hash already-stored in the Database is retrieved and compared to the hash resulted at the login event.
    7) If they are similar, then Authentication takes place.
       Otherwise, User is not authenticated.

- Using Json Web Token (JWT) Token while Login [Used in Authorization]

A JSON web token(JWT) is a JSON *Object* which is used to securely transfer information over the web(between two parties). It can be used for an authentication system and can also be used for information exchange.The token is mainly composed of *header, payload, and signature*. These three parts are separated by dots(.). JWT defines the structure of information we are sending from one party to the another

5.3.4 SignalR

SignalR is a library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data. You can refer to (SignalR Intro)[9] and (Comparison between Http requests and SignalR)[10] for more info.

In our process when a client arrives at the gate the gate sends a signal to backend server which in turn it sends requests to both the face recognition model and plate recognition model, then they send back their response which can be a failure or success, then the backend server checks the validation of all the given data to decide if the access is allowed or not. All those stages happen without any info given for the user. So we want a way to send info to the frontend about what is happening before ending the request. Here we have to use realtime sockets, which get initialized when the frontend sends a request then we have a channel to send any info from the backend to the frontend without having to wait for a request. We use the channel created to send that the face or plate models have started detection, then we send the response of each model that has been received in the backend to the frontend on the channel, And any other info needed are sent on the channel. Before ending the request we close the connection of the hub.

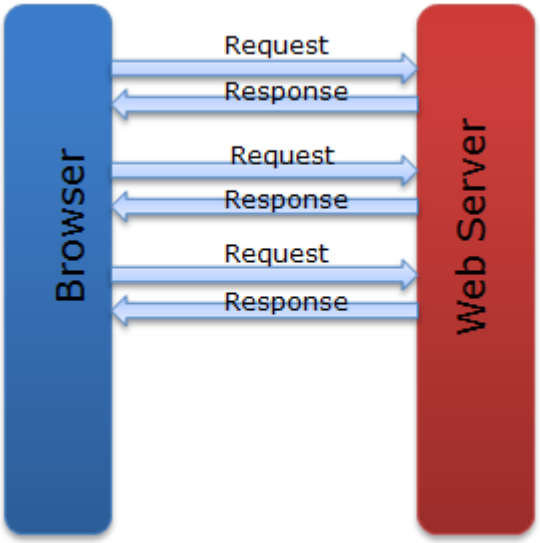

Figure 5.16 Normal Http requests and responses

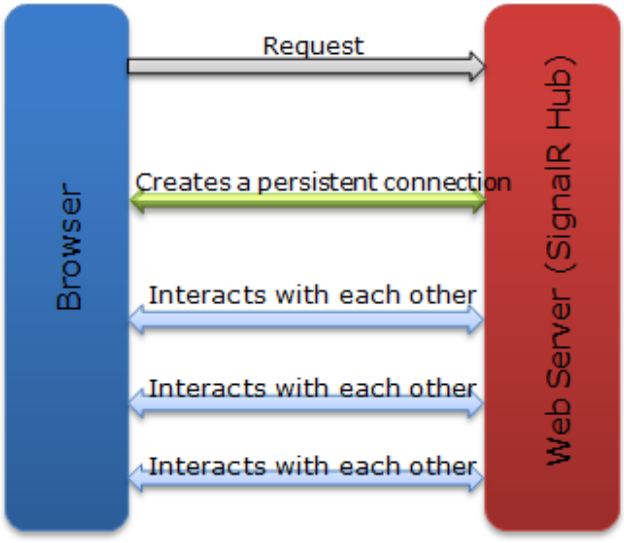

Figure 5.17 SignalR Connection

Chapter 6: Future Work

6.1 The causal parker (one-time visitor) scenario

The main difference between season parker and casual visitor is that the casual visitor is neither stored in the database (Face and Plate Number) nor the Face Model.

Therefore, the scenario is:

- Entrance Scenario:

1) Casual Parker passes at the entrance
2) Face Model tries to recognize the driver and search for the face in classes stored .
3) OCR reads the plate Number.
4) Since the driver data nor the Plate Number are found in Database, A video is captured for the driver's face.
5) Driver is allowed to enter (gate is opened) and at the same moment model trains using the video to store the casual parker in classes.
6) Plate Number and details of Casual Parker are stored temporarily in the Database.

- Exit Scenario:

1) Casual Parker passes at the exit.
2) The Face Model recognizes the driver as well as the OCR reads Plate Number.
3) Check for the pairing held at the entrance before (Meaning that the casual parker is already allowed to drive the vehicle)
4) If it is allowed :
    a) Temporary details of Casual Parker are deleted.
    b) Gate is opened
5) If it is not allowed : Gate is not open.

6.2 Real time architecture

- The current architecture
    - Frontend is responsible for detecting if a vehicle is waiting to enter the parking or not by a button simulating the wired loop, and send this info to backend.
    - Then the backend invokes the face recognition and the plate recognition models to run and return their results.
    - This architecture is a bit slow because of the time consumed during operating the two models.
    - This architecture is for simulation and developing purposes, Not suitable for a real time system.
- The real time system architecture
    - The face and plate models will be operating on a microprocessor like fpga.
    - There will be two flags, one inside the face model and another inside the plate model.
    - The backend server will be hosted on the cloud.
    - When the face model or plate model detects a face or plate they send their info to the backend server.
    - The backend server then checks if both flags are true which indicates that there is actually a car.
    - It then validates the info received (face id - plate id - subscription).
    - Then through a socket the backend invokes the gate to be open.

6.3 OCR pipeline enhancement

- **Collecting more real world data**

We already faced a lack of outsourcing data of Egyptian license plate images, let alone video data which is a time-consuming process and a cooperative task rather than individuals.

The images we trained and tested on so far are captured from different distances and various setups. We predict better results by having a fixed set up to collect data.

The required set-up to collect more data

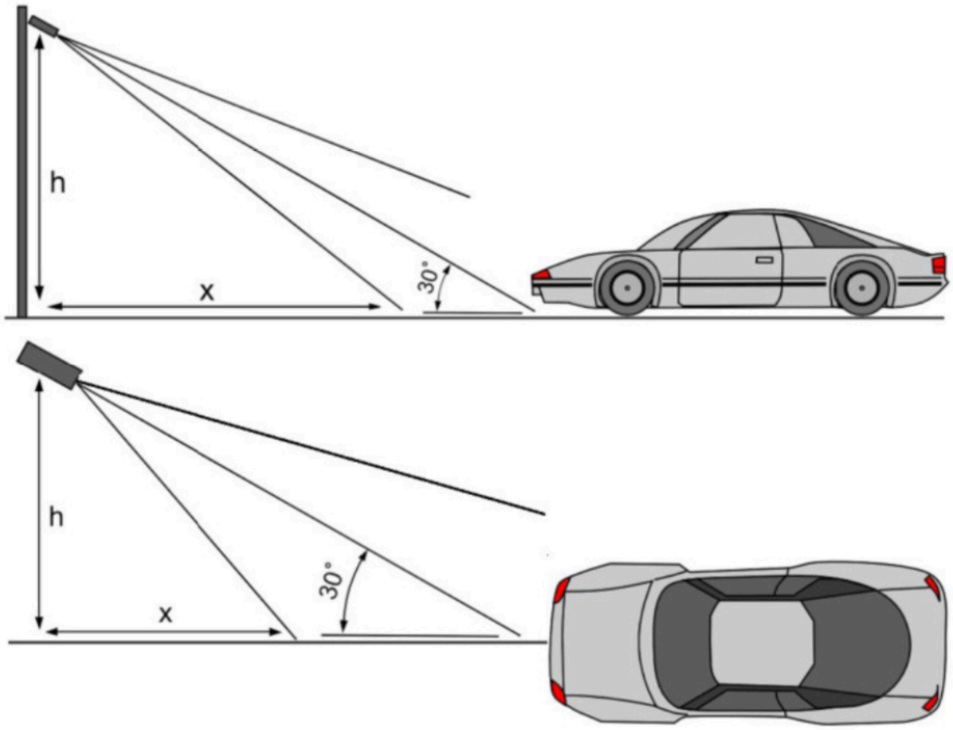

Figure 6.1 The required setup where the height of the LPR camera h is 3 meters, and the distance x is between 1 meter to 3 meters.

Same processes discussed in Chapter 3: License Plate Detector and Chapter 4: Optical Character Recognition can be easily applied and modified with the futurely collected data.

GitHub Repo

https://github.com/mohamed-samy2499/Smart-Authenticating-Parking-System

References

[1] Viola, P. and Jones, M. (2001) Rapid Object Detection Using a Boosted Cascade of Simple Features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, 8-14 December 2001, I-511-I-518.

[2] Dalal, Navneet, and Bill Triggs. Histograms of Oriented Gradients for Human Detection. 2005.

[3] Zhang, Kaipeng, et al. *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*. 11 April 2016.

[4] Schroff, Florian, et al. *FaceNet: A Unified Embedding for Face Recognition and Clustering*. 12 March 2015.

[5] Smith, Ray. Tesseract OCR. 5, 2020.

[6] M.A. Massoud, M. Sabee, M. Gergais, and R. Bakhit. "Automated new license plate recognition in Egypt" Alexandria Engineering Journal, vol. 52, no. 3, 2013. doi:10.1016/j.aej.2013.02.005

[7] Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” 2016, pp. 1-2. arXiv:1506.02640v5.

[8] Redmon, Joseph, and Ali Farhadi. “YOLO9000: Better, Faster, Stronger.” 25 Dec 2016. arXiv:1612.08242v1.

[9] Introduction to SignalR - Microsoft Documentation.

[10] Jose M. Aguilar. “ASP.Net SignalR, Incredible Simple Real-Time Features for Your Web App.”