Theory and Design

# Authentication Protocol

APMS allows a touch-free authentication system with QR-code authentication using an ESP32-CAM device to scan the QR-code. The authentication protocol can be summarized in the following steps:

## Entrance

1. User arrives at the entrance gate of the parking lot.
2. User holds the authentication QR-code towards the camera’s view.
3. ESP32-CAM shall recognize the largest QR-code within its view, and decrypt its message.
4. The cloud backend server shall authenticate the user using the decrypted QR-code message.
5. Once the user is authenticated, the cloud backend shall send the command to the ESP32-CAM to open the gate.

## Exit

1. User arrives at the exit gate of the parking lot.
2. User holds the authentication QR-code towards the camera’s view.
3. ESP32-CAM shall recognize the largest QR-code within it’s view and decrypt its message.
4. The cloud backend server shall authenticate the user using the decrypted QR-code message.
5. Once the user is authenticated, the cloud backend shall send the command to the ESP32-CAM to open the gate.
6. If the user has not paid at the time of exit they will receive an email with a warning.

QR-Code Script

import cv2

import numpy as np

import pyzbar.pyzbar as pyzbar

import urllib.request

#cap = cv2.VideoCapture(0)

font = cv2.FONT\_HERSHEY\_PLAIN

url='http://192.168.43.253:80/' # Local IP for demo purposes

cv2.namedWindow("live transmission", cv2.WINDOW\_AUTOSIZE)

# Checking for frame updates to avoid redundancy

prev=""

pres=""

# Main loop

while True:

img\_resp=urllib.request.urlopen(url+'cam-hi.jpg') # Accessing web server

imgnp=np.array(bytearray(img\_resp.read()),dtype=np.uint8) # Parsing image data

frame=cv2.imdecode(imgnp,-1) # Decoding image frame and displaying it

#\_, frame = cap.read()

decodedObjects = pyzbar.decode(frame) # QR code api decoding the QR code

for obj in decodedObjects:

pres=obj.data

if prev == pres:

pass

else:

print("Type:",obj.type)

print("Data: ",obj.data)

prev=pres

cv2.putText(frame, str(obj.data), (50, 50), font, 2, # Printing out the decoded QR code

(255, 0, 0), 3)

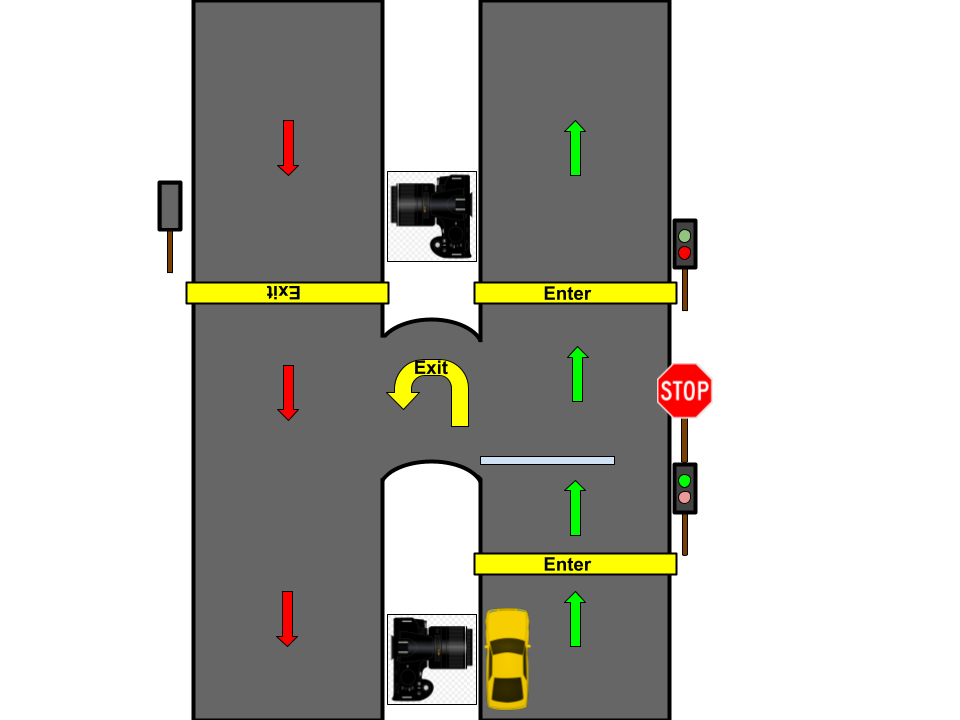
cv2.imshow("live transmission", frame)

key = cv2.waitKey(1)

if key == 27:

break

cv2.destroyAllWindows()

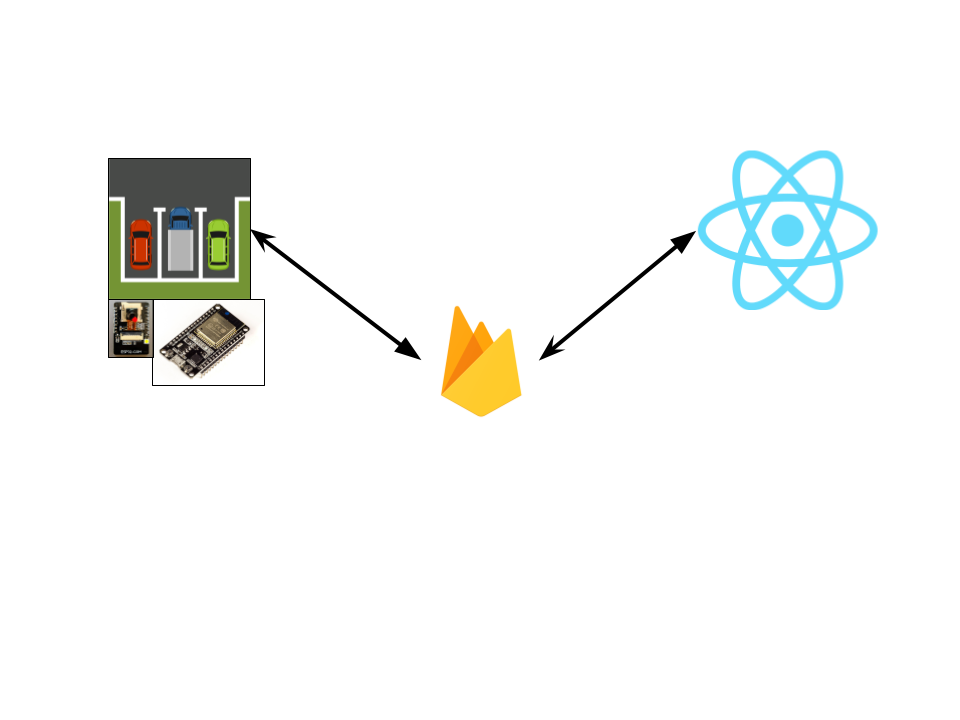


**Figure X:** Parking lot entrance and exit design.

# System Architecture

The Automated Parking Management System (APMS) can be divided into the following subsystems: Parking lot, Web-based GUI, Cloud backend server. The subsystems must work together for the functionality of the parking system. Each subsystem is dependent on one another. The dependency relationship between them is described below.

* Parking lot functionalities, such as gate movements, parking spot verification, alarm system, depend on the cloud backend for functionality.
* The Web-based GUI requires the cloud backend to make api calls and fulfill end-user requests. To retrieve data required for web pages.
* Cloud backend requires the esp32cam and esp23 to be connected to the internet for communication.



**Figure X:** Cloud (Firebase) and Web-based GUI (React) and Parking Lot (ESP32, ESP32CAM).

# Parking Lot Design

The parking lot can be divided into entrance/exit and the parking spots.

## Entrance/Exit

The entrance and exit will contain the hardware necessary for the functionality of authentication which is described in the following table:

**Table X:** List of parts for entrance/exit of parking lot.

| Part Name | Quantity | Part # | Cost (CAD) | Vendor |
| --- | --- | --- | --- | --- |
| [1x 5V Mini-Traffic Light 5mm LED Display Module for Creative DIY.](https://www.walmart.ca/en/ip/1x-5V-Mini-Traffic-Light-5mm-LED-Display-Module-for-Creative-DIY/PRD4LBZAEOEOOX3) | 2 | N/A | 7.82 | Walmart |
| [ESP32-CAM](https://www.amazon.ca/ESP32-CAM-ESP32-Development-Camera-Module/dp/B07T9561M7/ref=sr_1_1_sspa?crid=23G13ZL34UG7F&keywords=esp32+cam&qid=1707485216&sprefix=esp32cam%2Caps%2C71&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1) | 2 | Keenso65cn73iawt | 16.59 | Amazon |
| [SG90 9g Micro Servo Motor](https://www.amazon.ca/MMOBIEL-Airplane-Helicopter-Compatible-Raspberry/dp/B097RD8RB7/ref=asc_df_B097RD8RB7/?tag=googleshopc0c-20&linkCode=df0&hvadid=578819718870&hvpos=&hvnetw=g&hvrand=12143972672927407429&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9000899&hvtargid=pla-1543833503263&mcid=a4548075e8a036199272163f9135d9cf&th=1) | 2 |  | ~10 | N/A |

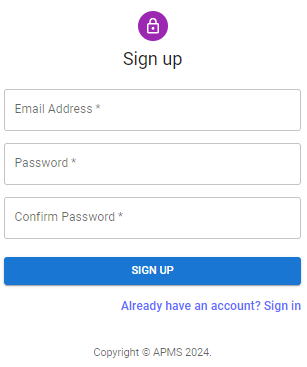
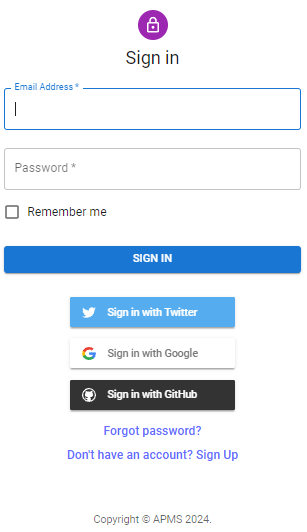
The design of the entrance/exit of the parking lot can be seen in **Figure X**.

# Web-Based GUI Application

The frontend GUI is going to be a web-based application that is hosted on the cloud using Firebase Hosting. The React.js framework is used to automate frontend development and Firebase Cloud functions are used to make api calls for dynamic web pages. The web-app is designed for, but not limited to mobile devices.

## Authentication

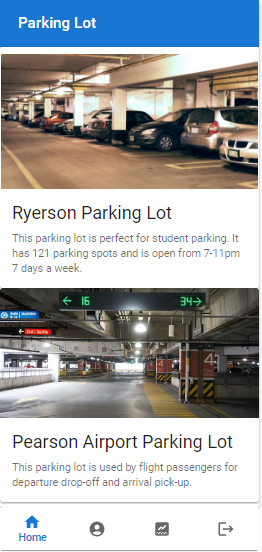
It shall have a basic authentication system, sign-up, login, logout, reset-password. Once the user is logged in, they will be able to visit 3 main pages: Home, Profile, Data, which they can access using the navigation bar. They can logout using the 4th option in the navigation bar. The authentication pages can be seen below.





## Home

This page will display the list of available parking lots registered with APMS as clickable cards. The cards shall display an image of the parking lot and a short description as seen below. Clicking on a parking lot card shall navigate to the parking lot page, which shows detailed information about the parking lot including a map and live occupation statistics, as well as buttons to book a parking spot or to generate a drop-in QRcode. Booking allows the user to select the type of parking: Premium, Disabled, Regular, and then displays a GUI to select available times.



## Profile

The profile pages is a very simple page with multiple information fields for the user to edit, such as drivers license number, license plate number, address, card information, etc. The user can also reset their password from this page, and they will be logged out as soon as their password is reset.

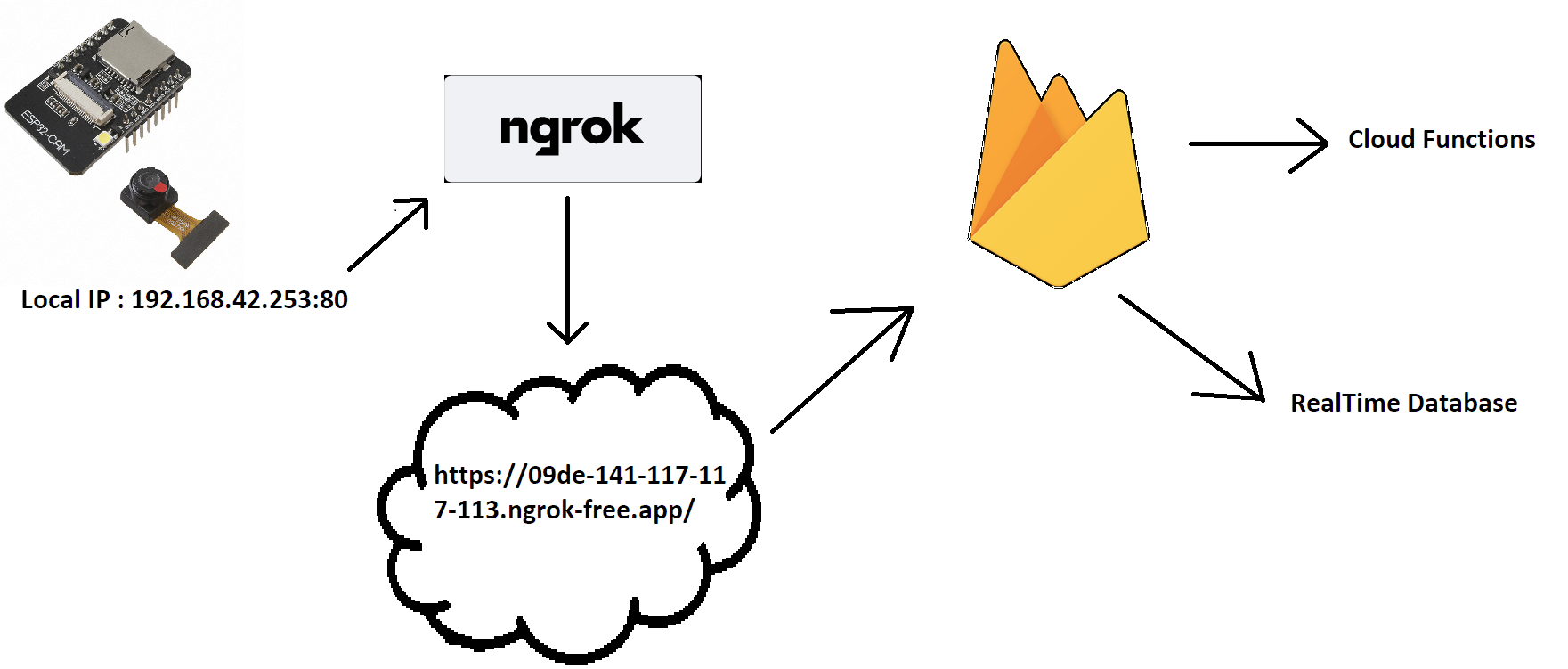
## Data

The collected data from the parking lot data acquisition system is formatted and presented in readable and understandable graphs and tables on the data page. Data related to the occupancy rate of each parking spot on each day of the week will be collected.

# Data Collection and Analysis

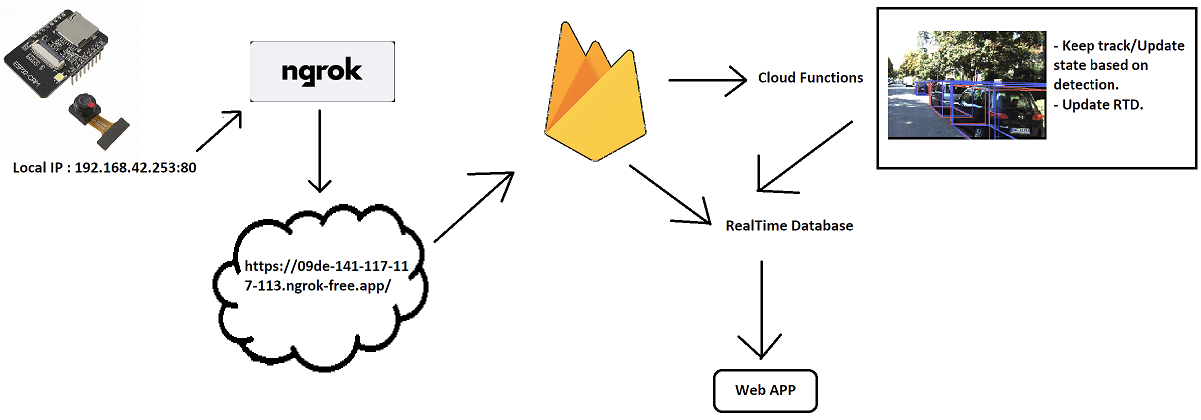
The data stream pipeline begins at ESP-CAM which acts as a web server. The web server code that is running on the ESPCAM, allocates a local ip which exposes the video feed. NGROK, a software that helps set up a reverse proxy/api endpoint fixes a static endpoint to the Web server local ip.

The cloud functions that are set up as part of the firebase database integration, can hence access the endpoint to retrieve image output from the ESPCAM as part of the data collection.



Note: The numbers presented are not real.

The cloud functions run image detection algorithms to then detect vehicles and parking spaces as objects and update the parking spot states accordingly. Using imported libraries including “cv2”, “yolo” which already include inbuilt Object Detection provide robust models that can be further fine tuned for specificity.



The Diagram on the left displays the cloud function interaction and the realtime database. The data analysis that runs based on the image detection algorithm concurrently updates the parking spot states.

Additionally, cost analysis algorithms will help in determining optimal revenue generation. Algorithms observe parking spot usage and using gradient descent functions or cost functions, determine the optimal pricing over time. More usage information over time leads to better results.

# Payment System

The following parking system has been generalized for most parking spots. The pricing may vary based on the location and demand of the parking lot but the system itself is the same as it takes advantage of key timings during the day as well as people who may want to drop in for a couple of hours. This parking payment system consists of 4 options. Morning rate, evening rate, hourly rate and an upgrade option to premium parking spots.

Morning Rate: 5am - 5pm, 25$

Evening Rate: 5pm - 5am, 10$

Hourly Parking: 5$ an hour

Premium parking Spots(more accessible locations): Additional 5$

This payment systems offers several advantages:

1. **Flexibility**: Users can choose between morning, evening rates, or hourly parking based on their needs and duration of stay. This flexibility caters to different schedules and preferences.

2: **Cost- Effectiveness**: The evening rate is significantly lower than the morning rate, which incentivizes people to park during off-peak hours. Hourly parking also allows users to pay only for the time they use, promoting cost-effectiveness.

3. **Premium Parking Option**: Offering premium parking spots for an additional fee provides convenience and accessibility for users who are willing to pay extra for a more accessible location. The use of cameras and sensors ensures fair usage and prevents misuse. These premium spots will be closer to exits and monitored by security cameras.

4. **Booking System**: Allowing users to book parking spots in advance enhances convenience and reduces uncertainty, especially during peak hours or for premium spots.

## Payment Method

Users must book parking in advance through the web-app. After reserving a parking space they must pre-pay online using their banking details.