



# "Pay As You Park" Smart Parking Solution 2021-198







# Student Information

Student ID	Student Name	Presentation Slides
IT18012552	M.D.S.M. Antany	Validate the parking yards standard and suggest the solution for parking yards
IT18154672	Priyankara A.D.D	Introduction and Find the availability of free spaces inside parking yard
IT18013092	Aadil M.R.M	Suggest and direct to most suitable parking yard for user
IT18013924	Ferreira L.V	Internal navigation in parking yards



# Introduction



WHAT IS SMART PARKING ?



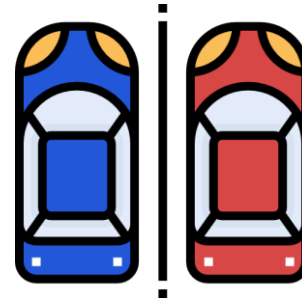
DOES CURRENT SOCIETY NEED  
A SMART PARKING SOLUTION?

# Research Problem



- Existing payment process in parking systems

- Hardness to find a parking yard to park
- Difficulty of navigation inside parking yards



- Difficulty of measuring parking yards  
standard without human interaction





# Objectives

- Introduce "pay as you park" concept to the parking system.
- Support users to find the most suitable parking yard
- Provide the best experience in parking using smart technology

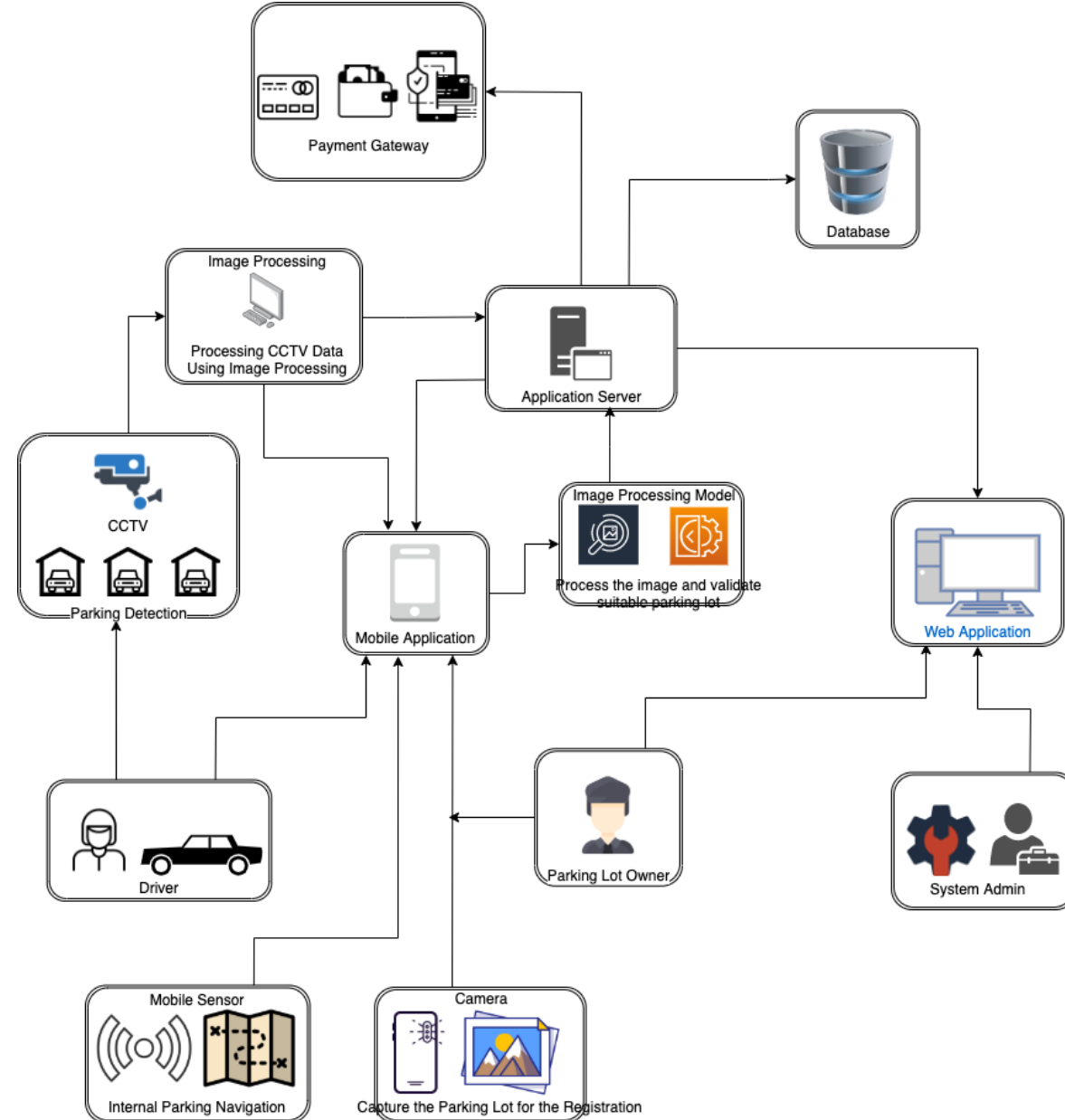


# Research Components

- Find the availability of free spaces inside parking yard
- Suggest and direct to most suitable parking yard for user
- Internal navigation in parking yards
- Measuring parking yards standard and validate suitability before the yard registration



# System Diagram







**IT18154672 | PRIYANKARA A. D. D.**

Software Engineering





# INTRODUCTION

- Research Problem
- Research Gap
- Objectives



# RESEARCH QUESTION

- What is the cost-friendly and accurate alternative to identify car parking spaces in a parking slot?





# OBJECTIVES

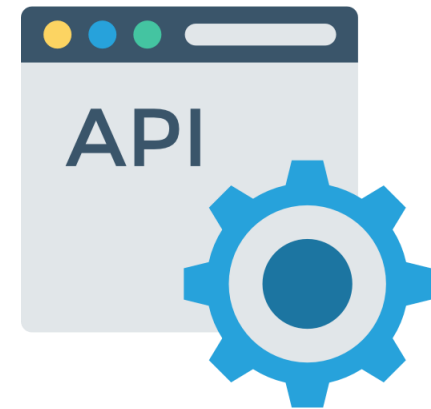
- Identifying vacant parking slots in a parking lot.





# SUB OBJECTIVES

- Save time of the user by pre-identifying vacant spaces.
- Send the processed data for the further process.





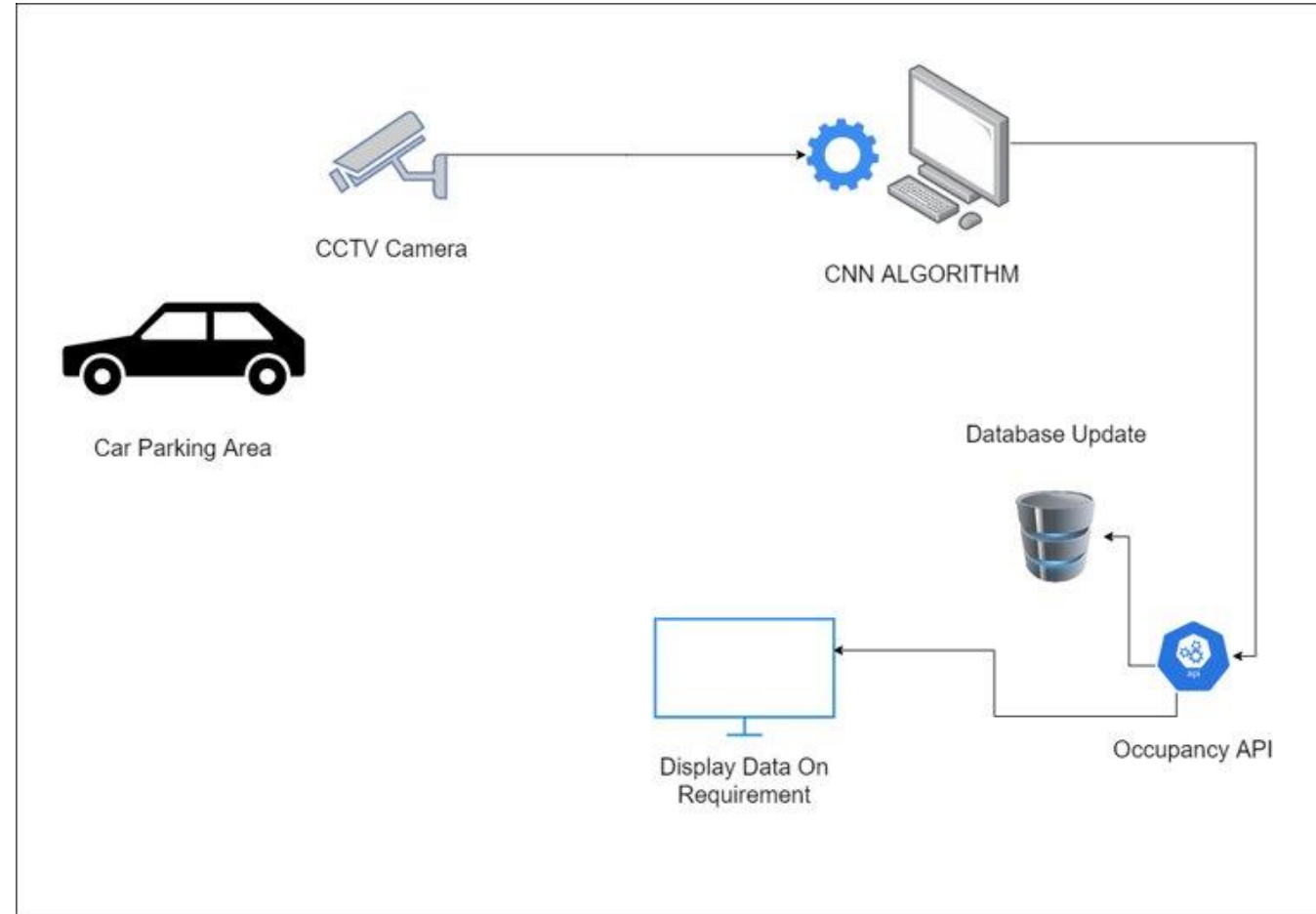


# RESEARCH METHODOLOGY

- System Diagram
- Technologies used



# SYSTEM DIAGRAM







# TECHNOLOGIES USED

- Identifying vacant/available parking spaces
- Number of vehicles detection
  - RestNet 50 CNN - (Convolutional Neural Networks) with
- REST APIs
  - Express JS with MongoDB

# EVIDENCE OF COMPLETION

```
% train the cubic SVM classifier with 5-fold cross validation, should take  
% 6 seconds to train the classifier
```

```
tic  
% using 5 fold cross validation in the model  
[trainedClassifier, validationAccuracy, validationPredictions] =...  
    trainClassifier(training_matrix);  
toc
```

Elapsed time is 4.712597 seconds.

```
disp('Training finished for the SVM classifier')
```

Training finished for the SVM classifier

```
% display the cross-validation accuracy of the trained classifier for the training data  
fprintf('The validation accuracy of the classifier %f %%.\\n',validationAccuracy*100);
```

The validation accuracy of the classifier 99.833333 %.







# EVIDENCE OF COMPLETION







# EVIDENCE OF COMPLETION







# REFERENCES

- [1] Sukumar, M. B., Sireesha, G., Ashok, A., Mounish, G., & Prathap, D. Real Time Image Processing Based Vacant Car Parking Occupancy Information System.
- [2] Nwave, (2021), Advantages and Disadvantages of Smart Parking Sensors | Nwave [Online] Available: <https://www.nwave.io/news/pros-and-cons-of-smart-parking-systems/> [Accessed 20 Feb 2021]
- [3] Paidi, V., Fleyeh, H., Håkansson, J., & Nyberg, R. G. (2018). Smart parking sensors, technologies and applications for open parking lots: a review. *IET Intelligent Transport Systems*, 12(8), 735-741.
- [4] PcMag, (2021), Definition of smart parking | PCMag [Online] Available: <https://www.pcmag.com/encyclopedia/term/smart-parking#:~:text=A%20vehicle%20parking%20system%20that,incoming%20drivers%20to%20available%20locations.&text=With%20the%20Smart%20Park%20system,car%2C%20smart%20home%20and%20smart> [Accessed 20 Feb 2021]
- [5] Gunasekara, G. G. Y. U., Gunasekara, A. D. A. I., & Kathriarachchi, R. P. S. (2015). A Smart Vehicle Parking Management Solution.
- [6] Karunarathne, M. S., & Nanayakkara, L. D. J. F. (2014). A Prototype to Identify Availability of a Car in a Smart Car Park with Aid of Programmable Chip and Infrared Sensors. *Journal of Emerging Trends in Computing and Information Sciences*, 5(2).
- [7] Nandyal, S., Sultana, S., & Anjum, S. (2017). Smart car parking system using arduino uno. *International Journal of Computer Applications*, 975(169), 1.



[8] Bachani, M., Qureshi, U. M., & Shaikh, F. K. (2016). Performance analysis of proximity and light sensors for smart parking. *Procedia Computer Science*, 83, 385-392.

[9] Britannica, (2021), Image processing | computer science | Britannica [Online]  
Available: <https://www.britannica.com/technology/image-processing> [Accessed 21 Feb 2021]

[10] True, N. (2007). Vacant parking space detection in static images. *University of California, San Diego*, 17, 659-662.

[11] Ichihashi, H., Notsu, A., Honda, K., Katada, T., & Fujiyoshi, M. (2009, August). Vacant parking space detector for outdoor parking lot by using surveillance camera and FCM classifier. In *2009 IEEE International Conference on Fuzzy Systems* (pp. 127-134). IEEE.





# IT18013092 | AADIL M.R.M

Software Engineering



# INTRODUCTION

- Research Question
- Specific and Sub Objectives





# RESEARCH PROBLEM

- Problems faced by the drivers when finding a parking yard.
- How does it affect the society ?
- How does it affect the environment ?



# Objectives

- Identify the nearest parking yard around user/user destination.
- Suggest optimal parking yard to park the vehicle based on key factors.
- Provide a cross platform mobile app to perform the task





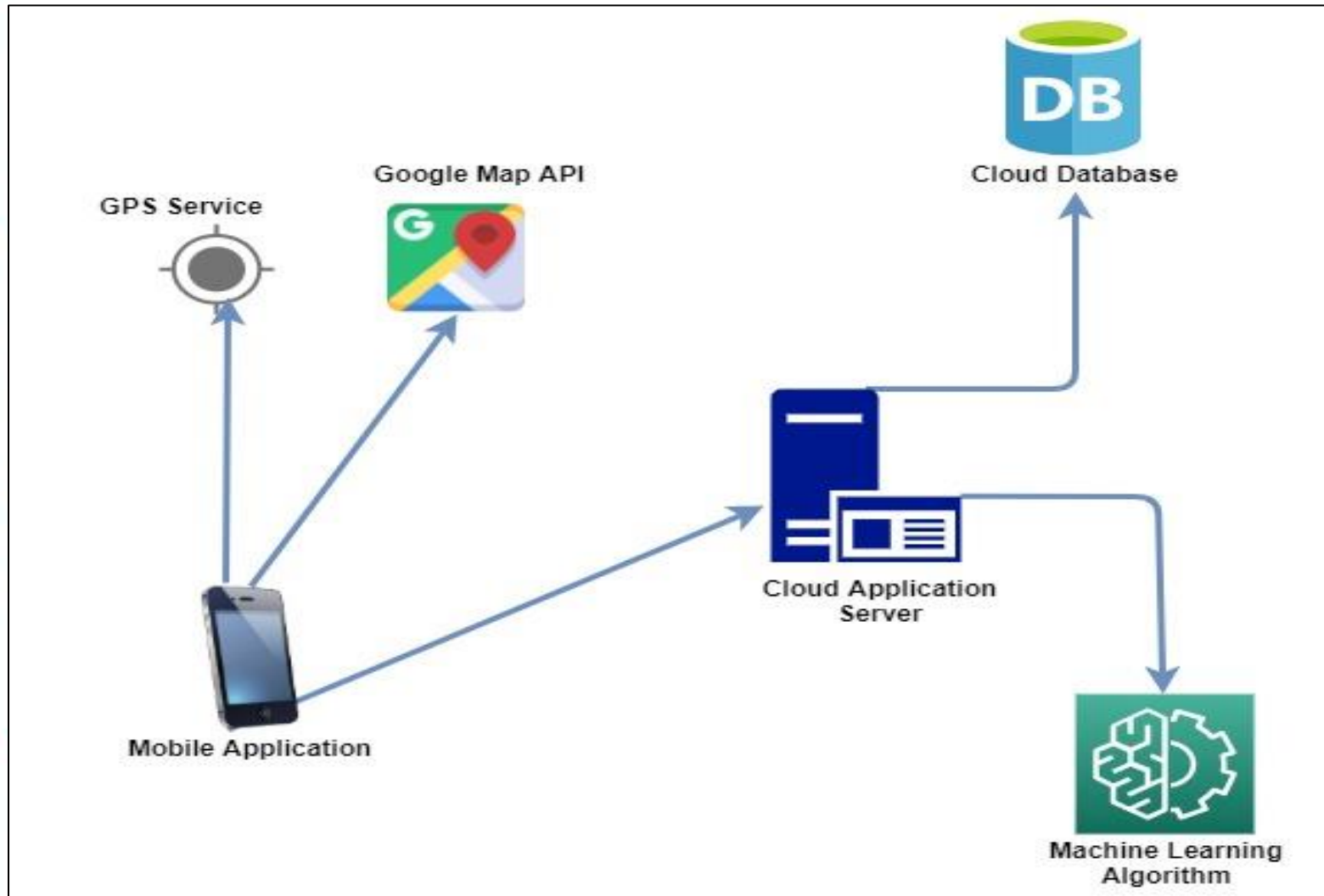
# RESEARCH METHODOLOGY

SYSTEM DAIGARM

TECHNOLOGY AND TECHNIQUES USED



# SYSTEM DIAGRAM







# TECHNOLOGY AND TECHNIQUES USED

- Retrieving the current location of user
  - GPS related technology
  - Google Map API – visualize the location
- Suggest optimal parking yard to park the vehicle based on key factors
  - Machine learning algorithms : SARIMA
  - Haversine Algorithm
  - Google Map API – visualize the locations and directions

# EVIDENCE FOR THE COMPLETION

## Haversine Imp

```
suggest_nearest_yard.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5 July

+ Code + Text
[ ] parkin_yards = parkin_yards.rename(columns={'X':'lat','Y':'lon'})
parkin_yards.head()

  ID  lat  lon  Name  address  no of slots
0  P1  79.882664  6.914951  LOLC Car Park  21, 25 Chandraleka Mawatha, Colombo 00800  120
1  P2  79.876301  6.883533  Best Western Elyon Colombo  Baseline Road, 102A Kirulapone Ave, Colombo 00500  75
2  P3  79.850030  6.933701  Fort  Fort, Colombo  100
3  P4  79.864816  6.928458  Maradana Railway Station  Jayantha Weerasekara Mawatha, Colombo 01000  225
4  P5  79.870644  6.879944  Tenaga Carparks (Pvt)Ltd  Level 4, 124 Maya Ave, Colombo 00500  300

Haversine Formula

from math import radians, cos, sin, asin, sqrt
def dist(lat1, long1, lat2, long2):
    # convert decimal degrees to radians
    lat1, long1, lat2, long2 = map(radians, [lat1, long1, lat2, long2])
    # haversine formula
    dlon = long2 - long1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371 * c
    return km

def find_nearest(lat, long):
    distances = parkin_yards.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    return parkin_yards.loc[distances.idxmin(), 'ID']

find_nearest(79.876301, 6.883533)

P2

Double-click (or enter) to edit
```

## Dataset Pattern Overview

```
jupyter DataSetPatternOverview Last Checkpoint: Last Saturday at 22:20 (autosaved)
File Edit View Insert Cell Kernel Widgets Help

In [1]: # Import libraries and custom functions defined in Workbook_Init.py
from initial import *

df_raw = pd.read_csv('./dataset.csv')
df_raw.head(10)


Out[1]:
  parking_yard_id  capacity  occupancy  last_update
0  BHMBCCMKT01      577         61  10/4/2016 7:59
1  BHMBCCMKT01      577         64  10/4/2016 8:25
2  BHMBCCMKT01      577         80  10/4/2016 8:59
3  BHMBCCMKT01      577        107  10/4/2016 9:32
4  BHMBCCMKT01      577        150  10/4/2016 9:59
5  BHMBCCMKT01      577        177  10/4/2016 10:26
6  BHMBCCMKT01      577        219  10/4/2016 10:59
7  BHMBCCMKT01      577        247  10/4/2016 11:25
8  BHMBCCMKT01      577        259  10/4/2016 11:59
9  BHMBCCMKT01      577        266  10/4/2016 12:29

In [2]: df_clean = df_raw.copy()
df_clean.last_update = df_clean.last_update.astype('datetime64')
df_clean['PercentOccupied'] = df_clean.occupancy / df_clean.capacity
df_clean['date'] = df_clean.last_update.dt.date
df_clean['dayofweek'] = df_clean.last_update.dt.dayofweek
df_clean['date_time_halfhour'] = df_clean.last_update.dt.round('30min')
df_clean['time'] = df_clean.date_time_halfhour.dt.time
```



# EVIDENCE FOR THE COMPLETION

## SARIMA Model



```
jupyter parking_availability Last Checkpoint: a day ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

# print('-'*77)
# print('ARIMA Model Metrics on Test Data')
# print('='*77)
# report_metrics(test.squeeze(), y_pred_AR.squeeze())

In [27]: %%time
# Define and fit SARIMA model
my_seasonal_order = (1, 1, 1, 48)
sarima_model = SARIMAX(train, order=(1, 0, 1), seasonal_order=my_seasonal_order)
results_SAR = sarima_model.fit(dispatch=-1)

C:\Users\moham\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency 30T will be used.
  warnings.warn('No frequency information was')
C:\Users\moham\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency 30T will be used.
  warnings.warn('No frequency information was')

Wall time: 1min 52s

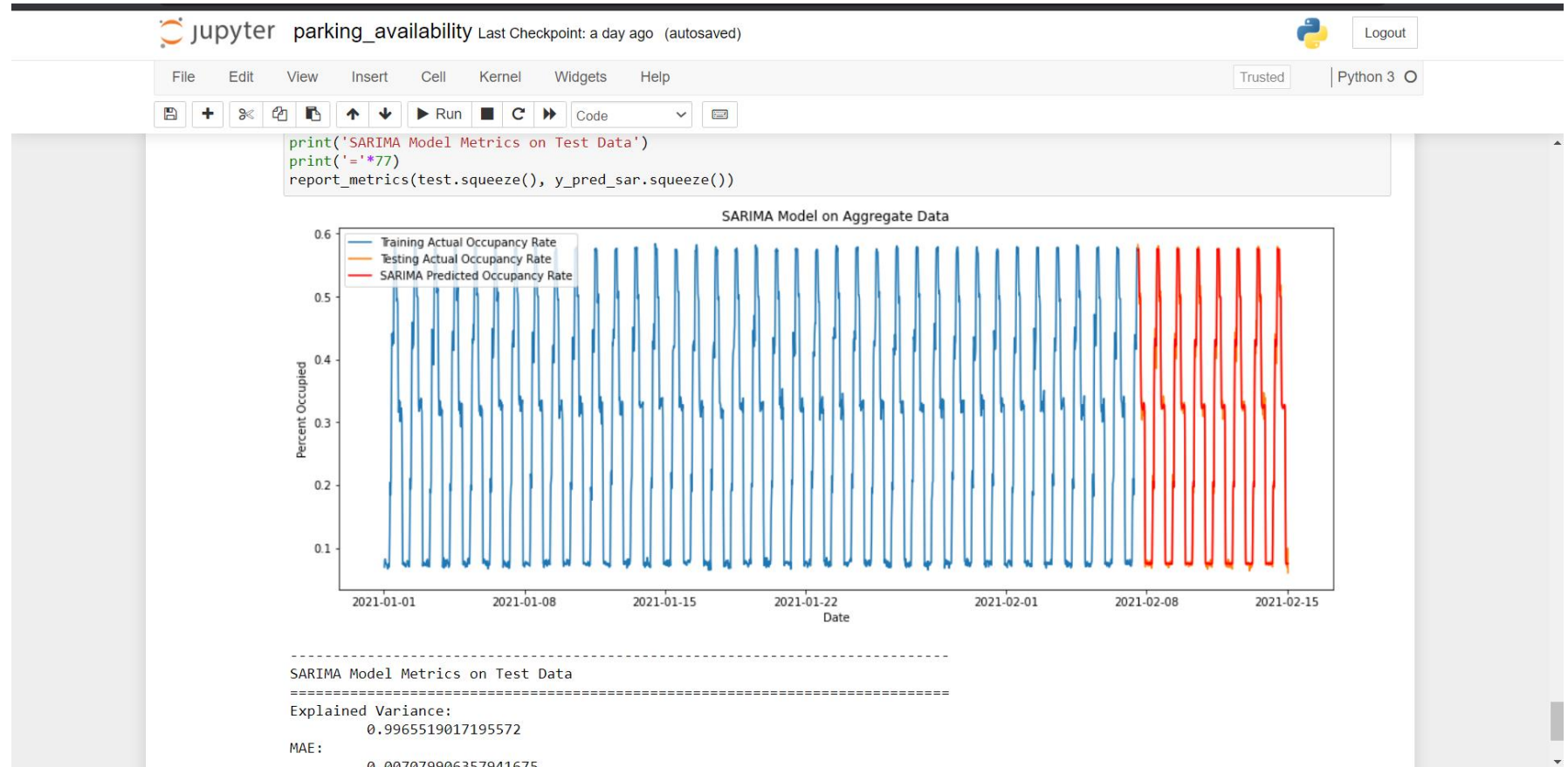
In [28]: plt.figure(figsize=(16,6))
plt.title('SARIMA Model on Aggregate Data')
plt.plot(train, label='Training Actual Occupancy Rate')
plt.xlabel('Date')
plt.ylabel('Percent Occupied')
y_pred_sar = pd.Series(results_SAR.forecast(steps=len(test)).values, index=test.index)
plt.plot(test, label='Testing Actual Occupancy Rate')
plt.plot(y_pred_sar, color='red', label='SARIMA Predicted Occupancy Rate')
plt.legend()

plt.show()
```



# EVIDENCE FOR THE COMPLETION

## SARIMA Forecasting







# IT18013924 | FERREIRA L.V.

Software Engineering



# INTERNAL PARKING NAVIGATION INSIDE A PARKING AREA







# INTRODUCTION

- What is a parking and an Internal Navigation inside a parking area?
- Indoor/outdoor parking areas.
- Why use Beacon technology and its advantages.



# OBJECTIVES

- Identify the user's position
- View user's position in a map
- Show users path to free parking slots



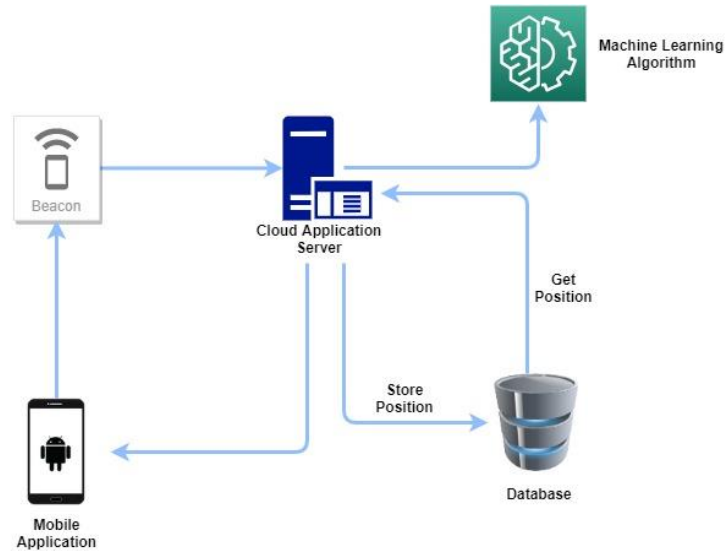


# METHODOLOGY

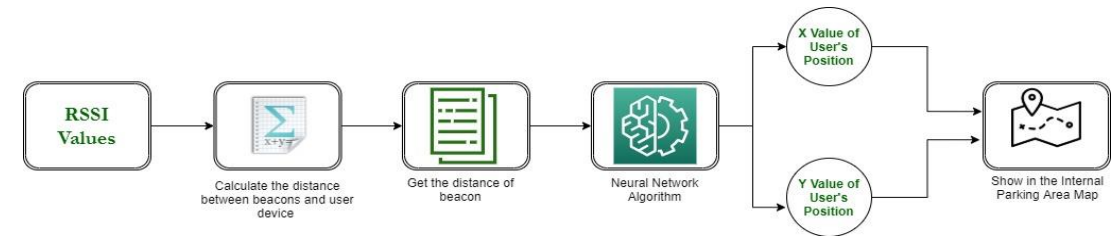
- Models Created (NN Model)
- Accuracy of these Models
- How to show predicted location in a map



# SYSTEM DIAGRAM AND COMPONENT FLOW



System Diagram



Component Flow





# TECHNOLOGY AND TECHNIQUES TO BE USED

- Identify the User's Position
  - Beacons
  - Calculate Distance of the Beacons(by getting RSSI values)
  - Machine learning algorithms : Neural Network(Sequential)
  - Pass image of the map and show user's position
  - Show path to the destination(free slot)



# COMPLETION OF THE COMPONENT

- Trained the Model
- Dummy Map which can show User Position
- Method Implemented to get position of a user when give three beacon distances to user as input Parameters
- Implement a Code to calculate the distance by using Beacon Bluetooth RSSI Values
- Used Kalman Filter to get more Accurate RSSI value
- Implement a way to show the path from user to the destination
- Design All the UIs and Databases (API used with dio to retrieve data from mongo db)





# FUTURE WORK

- Identify the stories of a building by using Beacons.
- Integrate the Application

# EVIDENCE FOR THE COMPLETION

## Model Predicts Position Y

```
202/202 [=====] - 0s 79us/step - loss: 0.9524 - acc: 0.5941 - val_loss: 0.7917 - val_acc: 0.7391
Epoch 8/200
202/202 [=====] - 0s 89us/step - loss: 0.9447 - acc: 0.5941 - val_loss: 0.8248 - val_acc: 0.7391
Epoch 9/200
202/202 [=====] - 0s 89us/step - loss: 0.9397 - acc: 0.5941 - val_loss: 0.7696 - val_acc: 0.7391
Epoch 10/200
202/202 [=====] - 0s 89us/step - loss: 0.9290 - acc: 0.5941 - val_loss: 0.7544 - val_acc: 0.7391
Epoch 11/200
202/202 [=====] - 0s 74us/step - loss: 0.9261 - acc: 0.5941 - val_loss: 0.7423 - val_acc: 0.7391
Epoch 12/200
202/202 [=====] - 0s 74us/step - loss: 0.9210 - acc: 0.5941 - val_loss: 0.8027 - val_acc: 0.7391
Epoch 13/200
202/202 [=====] - 0s 74us/step - loss: 0.9176 - acc: 0.5941 - val_loss: 0.7401 - val_acc: 0.7391
Epoch 14/200
202/202 [=====] - 0s 74us/step - loss: 0.9127 - acc: 0.5941 - val_loss: 0.7564 - val_acc: 0.7391
Epoch 15/200
```

```
In [1]: import pandas as pd
2
3 dataset=pd.read_csv('boston_readings.csv').values

In [2]: dataset.shape

In [3]: #to predict Y
2 target=dataset[:,4]
3 from keras.utils import np_utils
4 categorical_target=np_utils.to_categorical(target)

C:\Users\Administrator\anaconda3\envs\notebooks\lib\site-packages\ipykernel_launcher.py:36: FutureWarning: Conversion of the second argument of `subdtype` from `float` to `np.float64` is deprecated. In the future, it will be treated as `np.float64 == np.dtype(float).type`.
from _collections import defaultdict
from _collections import defaultdict

In [4]: from sklearn.model_selection import train_test_split
2
3 train_data,test_data,train_target,test_target=train_test_split(data,categorical_target,test_size=0.1)

In [5]: from keras.models import Sequential
2 from keras.layers import Dense
3 model=Sequential()
4 model.add(Dense(16,input_dim=1,activation='relu'))
5 model.add(Dense(16,activation='relu'))
6 model.add(Dense(16,activation='softmax'))
7 model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

In [6]: model.fit(train_data,train_target,epochs=200,validation_split=0.1)

Train on 180 samples, validate on 20 samples
Epoch 1/200
202/202 [=====] - 0s 2m/step - loss: 4.6882 - acc: 0.2070 - val_loss: 3.3048 - val_acc: 0.4348
Epoch 2/200
```

```
202/202 [=====] - 0s 79us/step - loss: 0.9524 - acc: 0.5941 - val_loss: 0.7917 - val_acc: 0.7391
Epoch 8/200
202/202 [=====] - 0s 89us/step - loss: 0.9447 - acc: 0.5941 - val_loss: 0.8248 - val_acc: 0.7391
Epoch 9/200
202/202 [=====] - 0s 89us/step - loss: 0.9397 - acc: 0.5941 - val_loss: 0.7696 - val_acc: 0.7391
Epoch 10/200
202/202 [=====] - 0s 89us/step - loss: 0.9290 - acc: 0.5941 - val_loss: 0.7544 - val_acc: 0.7391
Epoch 11/200
202/202 [=====] - 0s 74us/step - loss: 0.9261 - acc: 0.5941 - val_loss: 0.7423 - val_acc: 0.7391
Epoch 12/200
202/202 [=====] - 0s 74us/step - loss: 0.9210 - acc: 0.5941 - val_loss: 0.8027 - val_acc: 0.7391
Epoch 13/200
202/202 [=====] - 0s 74us/step - loss: 0.9176 - acc: 0.5941 - val_loss: 0.7401 - val_acc: 0.7391
Epoch 14/200
202/202 [=====] - 0s 74us/step - loss: 0.9127 - acc: 0.5941 - val_loss: 0.7564 - val_acc: 0.7391
Epoch 15/200
```

```
In [7]: 1 import numpy as np
2 predicted_target=model.predict(test_data)
3 print(np.argmax(predicted_target,axis=1))

[180 180 137 223 223 180 223 180 180 180 223 180 137 137 137 180 180
 223 180 180 223 180 137 137]
```

```
In [8]: 1 print(np.argmax(test_target,axis=1))

[180 180 137 223 223 180 223 180 223 180 223 180 137 137 137 223 180
 223 180 180 223 180 137 137]
```

```
In [9]: 1 from sklearn.metrics import accuracy_score
2
3 accuracy=accuracy_score(np.argmax(test_target,axis=1),np.argmax(predicted_target,axis=1))
4 print(accuracy)

0.88
```

```
In [10]: 1 model.save_weights('y_weights.h5') #to save only weights
2 model.save('y.model') #to save the whole model
```



# EVIDENCE FOR THE COMPLETION

## Model Predicts Position X

```
localhost:8888/notebooks/Desktop/Lahiru%20Ver/creating%20the%20model%20for%20X.ipynb

jupyter creating the model for X Last Checkpoint 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [1]: 1 import pandas as pd
        2 dataset=pd.read_csv('beacon_readings.csv').values

In [2]: 1 datasetdataset[:,0:3]

In [3]: 1 #to predict X
        2 target=dataset[:,1]
        3 from keras.utils import np_utils
        4
        5 categorized_target=np_utils.to_categorical(target)

C:\Users\Administrator\anaconda3\envs\newenv1\lib\site-packages\h5py\_init_.py:36: FutureWarning: Conversion of the second argument of issubdtype from 'float' to 'np.float64' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

In [4]: 1 from sklearn.model_selection import train_test_split
        2
        3 train_data,test_data,train_target,test_target=train_test_split(data,categorized_target,test_size=0.1)

In [5]: 1 from keras.models import Sequential
        2 from keras.layers import Dense
        3
        4 model=Sequential()
        5 model.add(Dense(64,input_dim=3,activation='relu'))
        6 model.add(Dense(32,activation='relu'))
        7 model.add(Dense(16,activation='softmax'))
        8
        9 model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

In [6]: 1 model.fit(train_data,train_target,epochs=200,validation_split=0.1)

Train on 202 samples, validate on 23 samples
Epoch 1/200
202/202 [=====] - 0s 718us/step - loss: 4.0376 - acc: 0.4752 - val_loss: 2.6483 - val_acc: 0.4348
Epoch 2/200
```

```
localhost:8890/notebooks/Desktop/Lahiru%20Ver/creating%20the%20model%20for%20Y.ipynb

jupyter creating the model for Y Last Checkpoint: Yesterday at 7:48 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

epoch 24/200
202/202 [=====] - 0s 74us/step - loss: 0.5823 - acc: 0.6337 - val_loss: 0.6688 - val_acc: 0.7391
Epoch 55/200
202/202 [=====] - 0s 64us/step - loss: 0.5758 - acc: 0.7475 - val_loss: 0.6697 - val_acc: 0.7391
Epoch 56/200
202/202 [=====] - 0s 79us/step - loss: 0.5717 - acc: 0.6782 - val_loss: 0.6696 - val_acc: 0.6957
Epoch 57/200
202/202 [=====] - 0s 74us/step - loss: 0.5651 - acc: 0.6832 - val_loss: 0.6775 - val_acc: 0.6957
Epoch 58/200
202/202 [=====] - 0s 64us/step - loss: 0.5601 - acc: 0.6733 - val_loss: 0.6594 - val_acc: 0.8261
Epoch 59/200
202/202 [=====] - 0s 59us/step - loss: 0.5550 - acc: 0.7535 - val_loss: 0.6570 - val_acc: 0.8261

In [8]: 1 import numpy as np
        2 predicted_target=model.predict(test_data)
        3 print(np.argmax(predicted_target,axis=1))

[137 180 180 137 180 180 223 180 137 137 180 180 137 137 223 223 223 137
 223 180 180 180 180 180 223]

In [9]: 1 print(np.argmax(test_target,axis=1))

[137 180 180 137 180 180 223 180 137 137 180 180 137 137 223 223 223 137
 223 180 180 180 180 180 223]

In [10]: 1 from sklearn.metrics import accuracy_score
         2
         3 accuracy=accuracy_score(np.argmax(test_target,axis=1),np.argmax(predicted_target,axis=1))
         4 print(accuracy)

1.0

In [11]: 1 model.save_weights('y_weights.h5') #to save only weights
         2 model.save('y.model') #to save the whole model

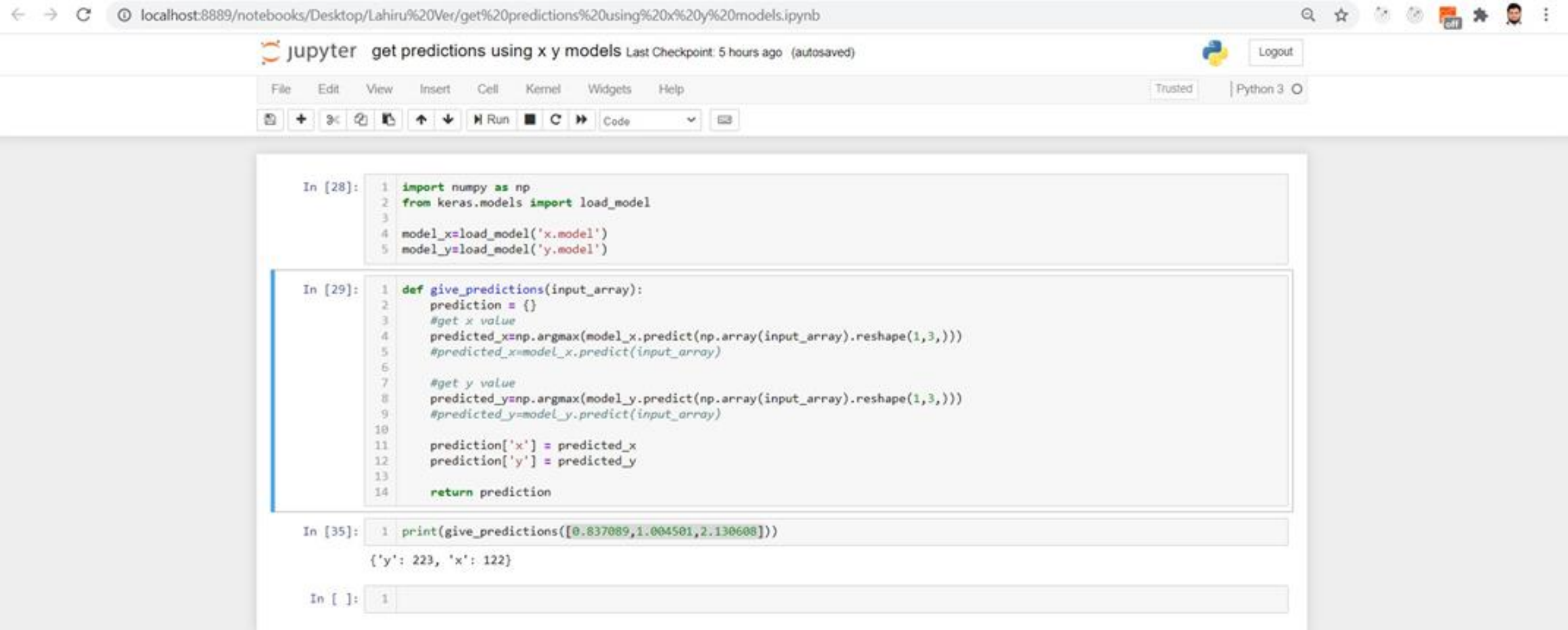
In [12]: 1 model.save('modelY.pkl')

In [ ]: 1
```



# EVIDENCE FOR THE COMPLETION

## Implemented method to get position



The screenshot displays a Jupyter Notebook titled "get predictions using x y models" running on a local host. The notebook contains three code cells. The first cell imports numpy and keras, and loads two models. The second cell defines a function "give\_predictions" that takes an input array and returns a dictionary with predicted 'x' and 'y' values. The third cell prints the output of the function for a specific input array.

```
In [28]: 1 import numpy as np
          2 from keras.models import load_model
          3
          4 model_x=load_model('x.model')
          5 model_y=load_model('y.model')
```

```
In [29]: 1 def give_predictions(input_array):
          2     prediction = {}
          3     #get x value
          4     predicted_x=np.argmax(model_x.predict(np.array(input_array).reshape(1,3)))
          5     #predicted_x=model_x.predict(input_array)
          6
          7     #get y value
          8     predicted_y=np.argmax(model_y.predict(np.array(input_array).reshape(1,3)))
          9     #predicted_y=model_y.predict(input_array)
          10
          11     prediction['x'] = predicted_x
          12     prediction['y'] = predicted_y
          13
          14     return prediction
```

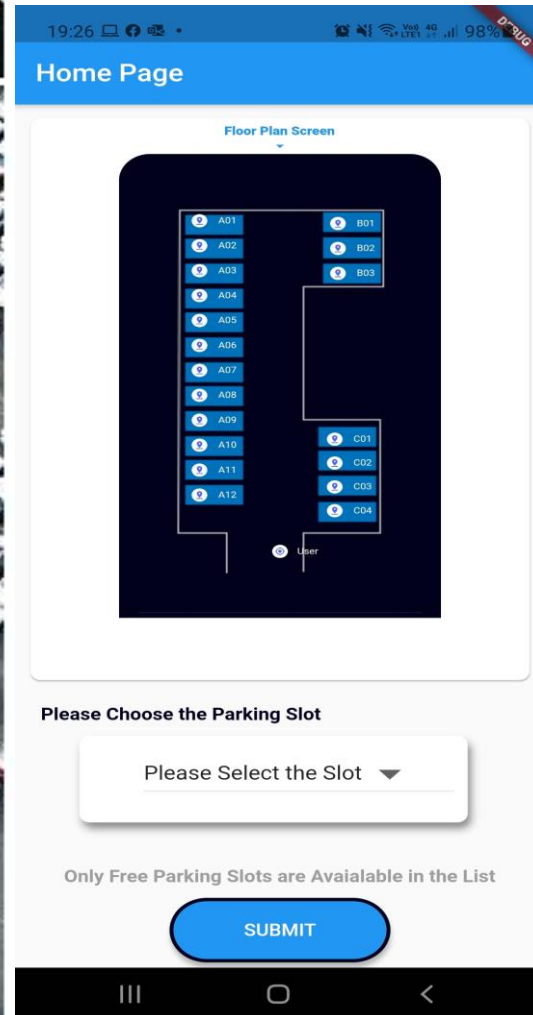
```
In [35]: 1 print(give_predictions([0.837089,1.004501,2.130608]))
          {'y': 223, 'x': 122}
```

```
In [ ]: 1
```

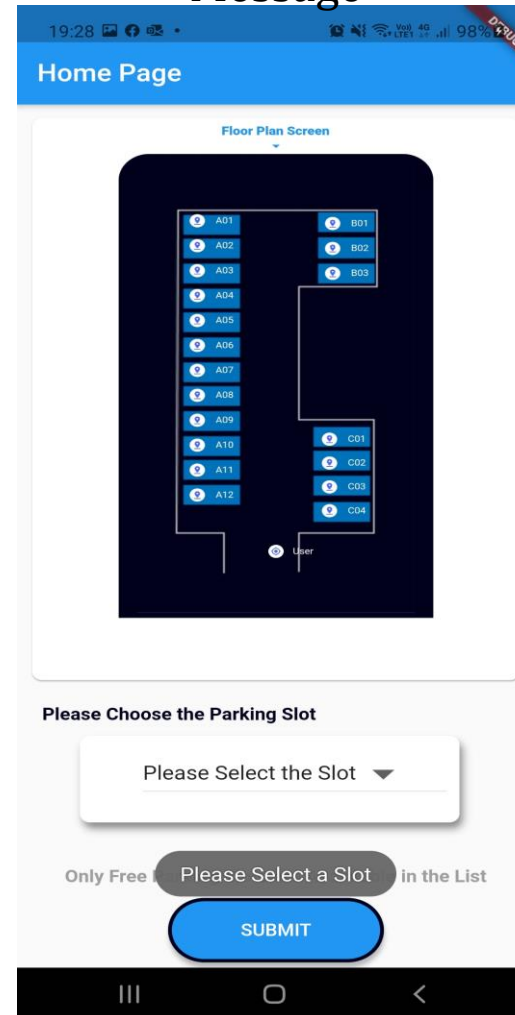


# EVIDENCE FOR THE COMPLETION

Main User Interface



Main User Interface – Toast Message



## User Guide

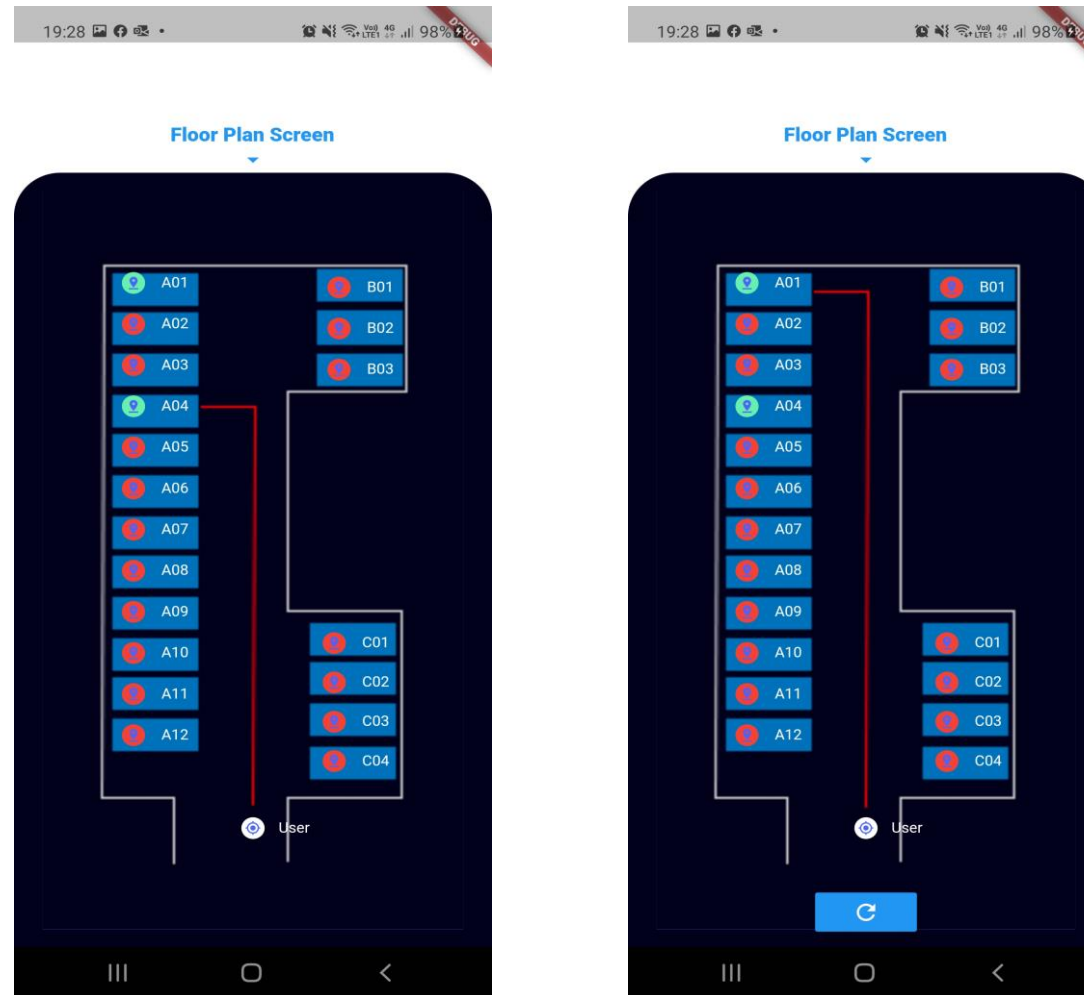
Image of the parking Area

Dropdown to Select the Parking Slot

Toast message will be shown if the user does not select the parking slot

# EVIDENCE FOR THE COMPLETION

## Navigation UI



## User Guide

User's position is shown on the map with the slots

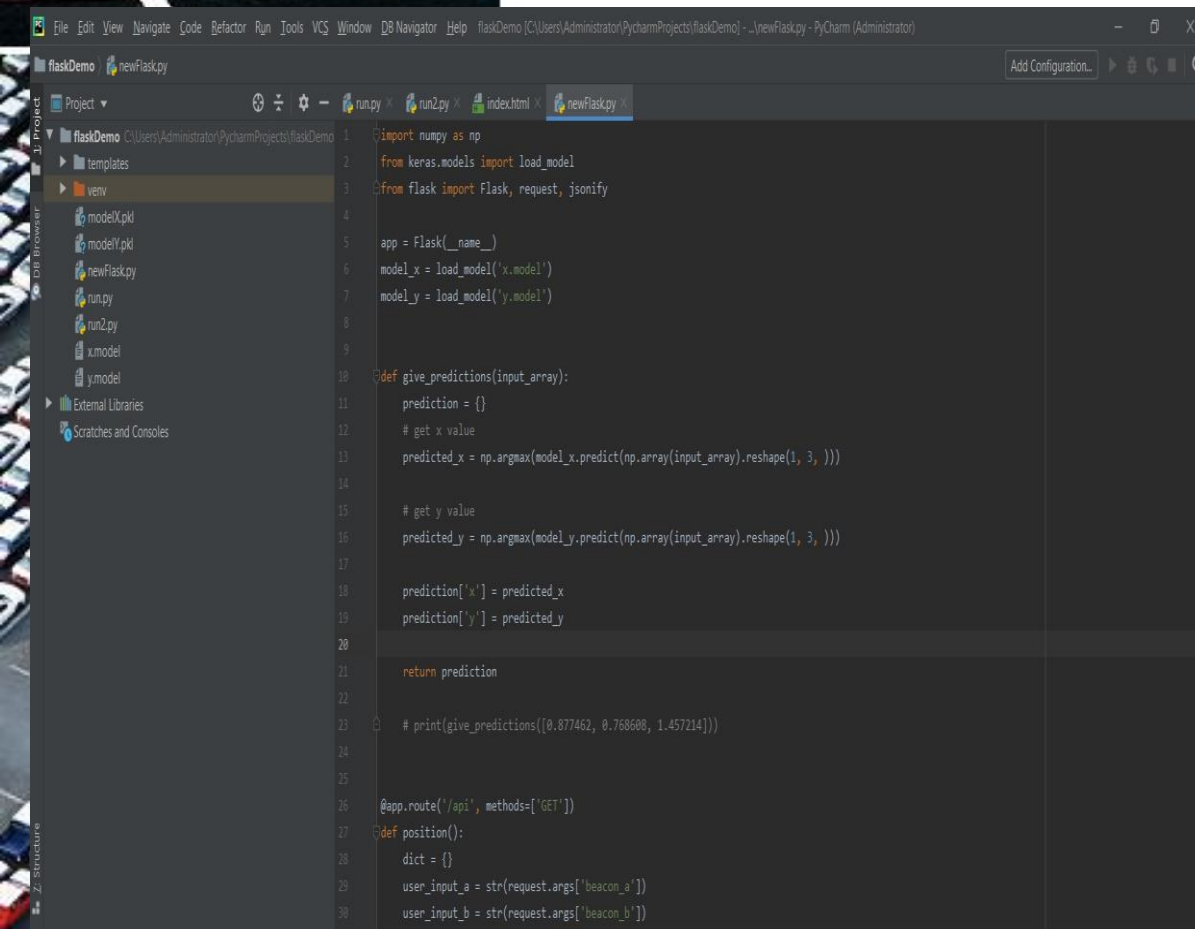
Red Color Slots – Unavailable  
Parking Slots

Green Color Slots – Available  
Parking Slots

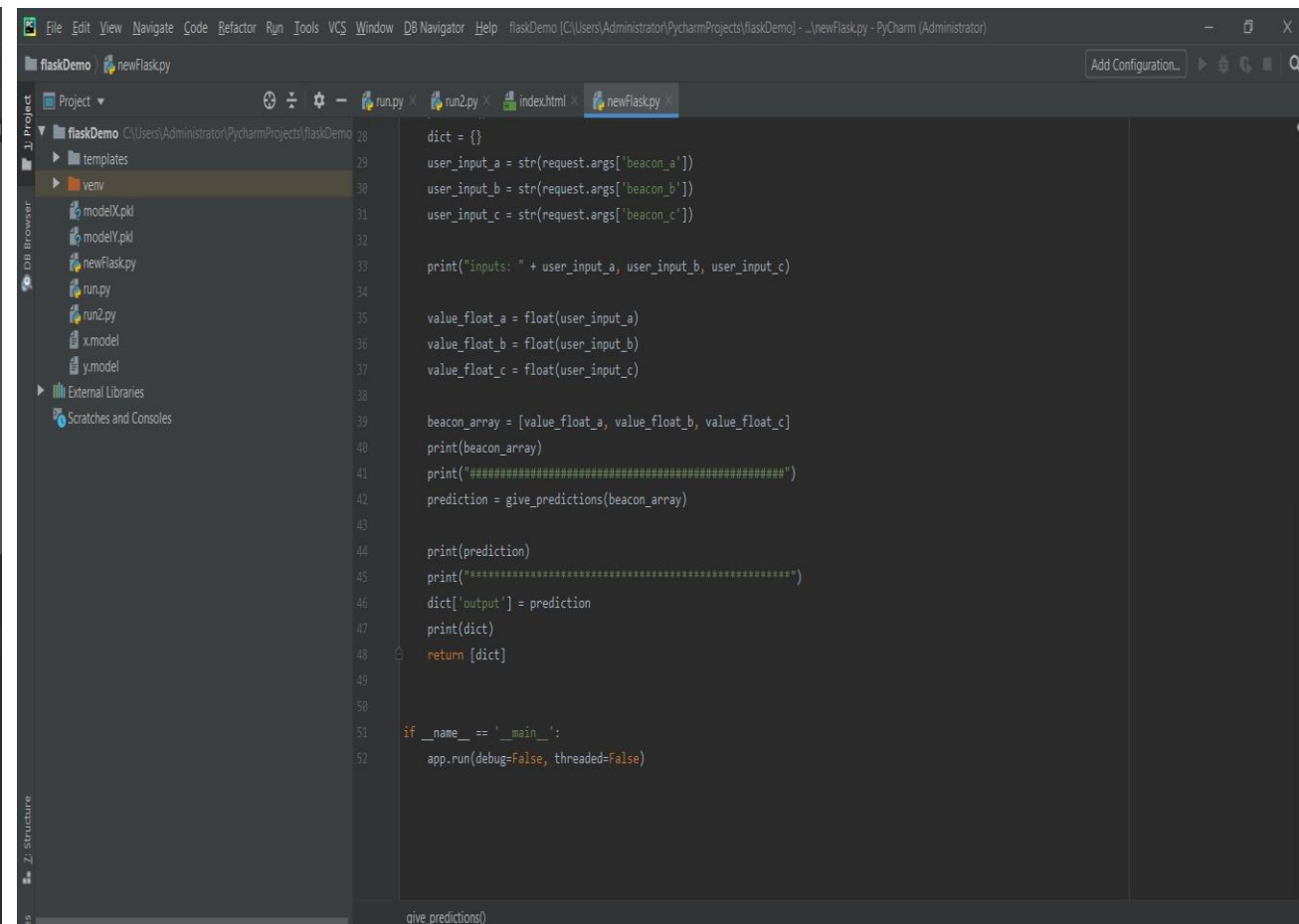


# EVIDENCE FOR THE COMPLETION

## Implemented Model API using Flask



```
1 import numpy as np
2 from keras.models import load_model
3 from flask import Flask, request, jsonify
4
5 app = Flask(__name__)
6 model_x = load_model('x.model')
7 model_y = load_model('y.model')
8
9
10 def give_predictions(input_array):
11     prediction = {}
12     # get x value
13     predicted_x = np.argmax(model_x.predict(np.array(input_array).reshape(1, 3, )))
14
15     # get y value
16     predicted_y = np.argmax(model_y.predict(np.array(input_array).reshape(1, 3, )))
17
18     prediction['x'] = predicted_x
19     prediction['y'] = predicted_y
20
21     return prediction
22
23 # print(give_predictions([0.877462, 0.768608, 1.457214]))
24
25
26 @app.route('/api', methods=['GET'])
27 def position():
28     dict = {}
29     user_input_a = str(request.args['beacon_a'])
30     user_input_b = str(request.args['beacon_b'])
```

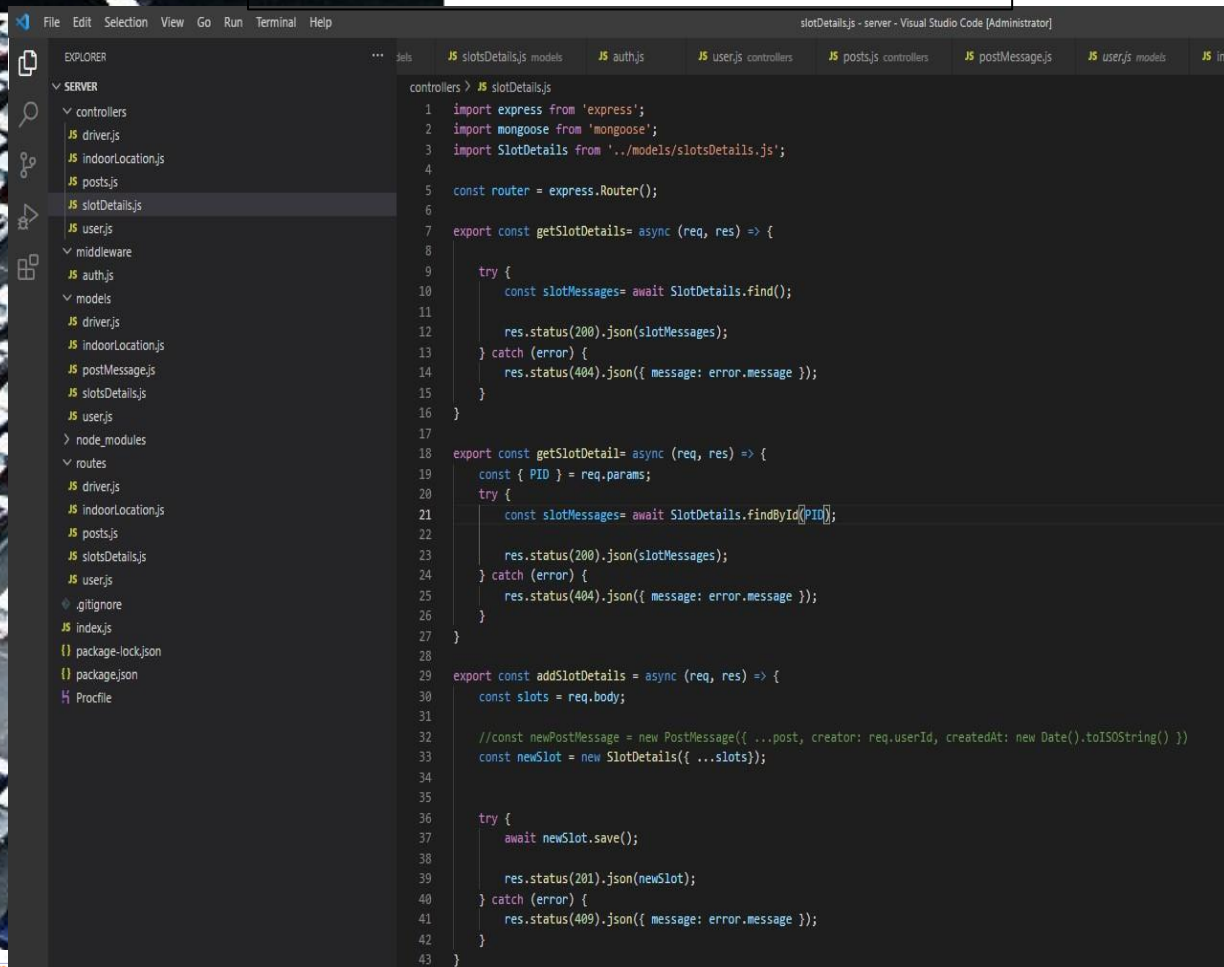


```
28 dict = {}
29 user_input_a = str(request.args['beacon_a'])
30 user_input_b = str(request.args['beacon_b'])
31 user_input_c = str(request.args['beacon_c'])
32
33 print("inputs: " + user_input_a, user_input_b, user_input_c)
34
35 value_float_a = float(user_input_a)
36 value_float_b = float(user_input_b)
37 value_float_c = float(user_input_c)
38
39 beacon_array = [value_float_a, value_float_b, value_float_c]
40 print(beacon_array)
41 print("#####")
42 prediction = give_predictions(beacon_array)
43
44 print(prediction)
45 print("#####")
46 dict['output'] = prediction
47 print(dict)
48 return [dict]
49
50
51 if __name__ == '__main__':
52     app.run(debug=False, threaded=False)
```

# EVIDENCE FOR THE COMPLETION

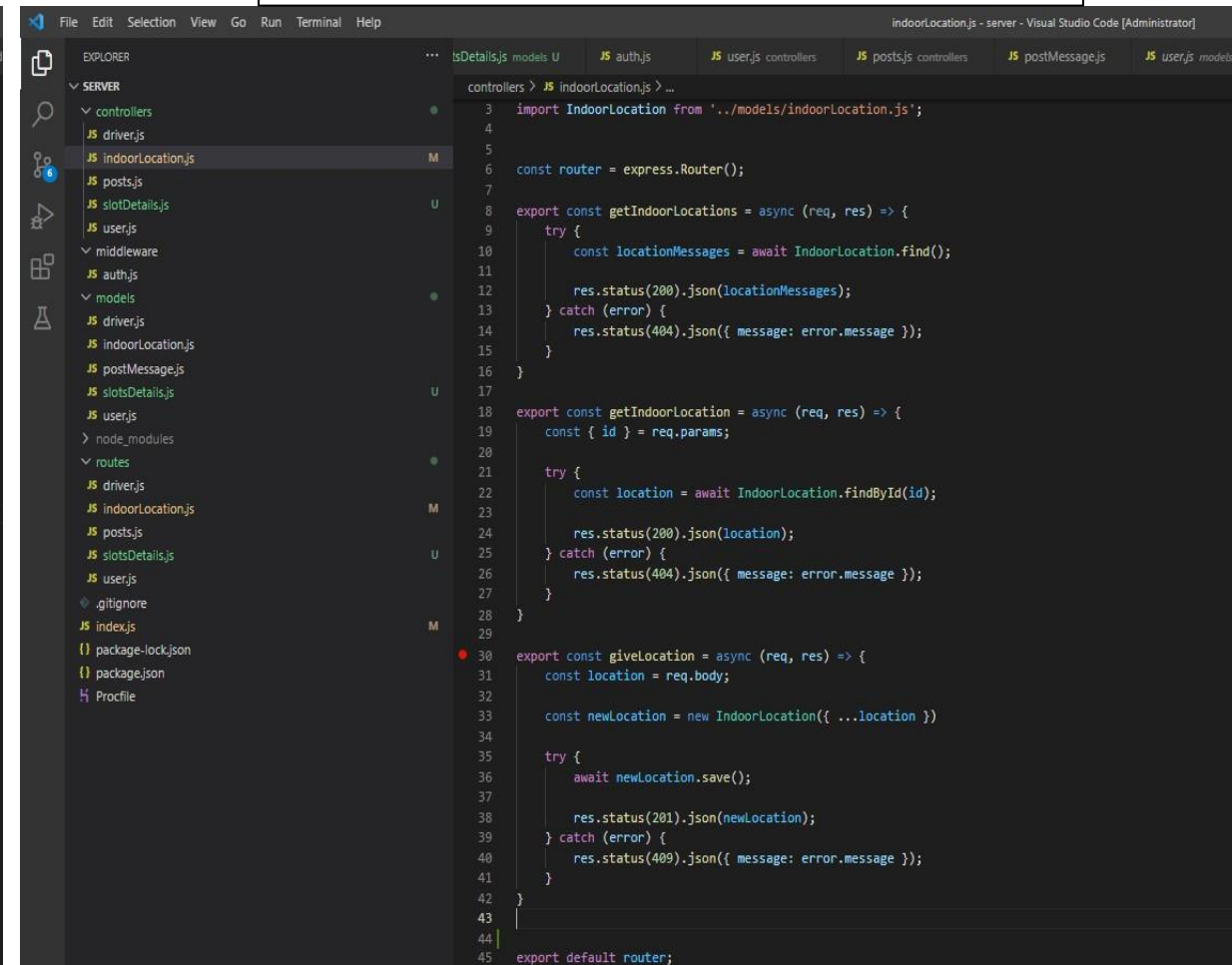
## Implemented MongoDB access API

### Slots Retrieve API Implementation



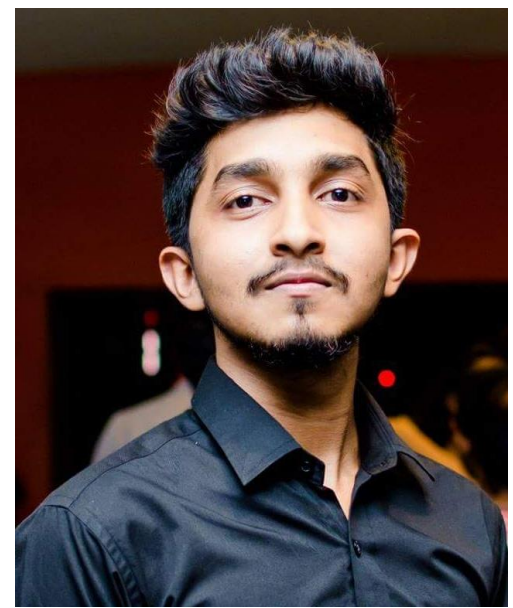
```
1 import express from 'express';
2 import mongoose from 'mongoose';
3 import SlotDetails from '../models/slotDetails.js';
4
5 const router = express.Router();
6
7 export const getSlotDetails = async (req, res) => {
8
9   try {
10     const slotMessages = await SlotDetails.find();
11     res.status(200).json(slotMessages);
12   } catch (error) {
13     res.status(404).json({ message: error.message });
14   }
15 }
16
17 export const getSlotDetail = async (req, res) => {
18   const { PID } = req.params;
19   try {
20     const slotMessages = await SlotDetails.findById(PID);
21     res.status(200).json(slotMessages);
22   } catch (error) {
23     res.status(404).json({ message: error.message });
24   }
25 }
26
27 export const addSlotDetails = async (req, res) => {
28   const slots = req.body;
29
30   //const newPostMessage = new PostMessage({ ...post, creator: req.userId, createdAt: new Date().toISOString() })
31   const newSlot = new SlotDetails({ ...slots });
32
33   try {
34     await newSlot.save();
35     res.status(201).json(newSlot);
36   } catch (error) {
37     res.status(409).json({ message: error.message });
38   }
39 }
40
41
42
43 }
```

### Indoor Location API Implementation



```
1 import IndoorLocation from '../models/indoorLocation.js';
2
3 const router = express.Router();
4
5 export const getIndoorLocations = async (req, res) => {
6   try {
7     const locationMessages = await IndoorLocation.find();
8     res.status(200).json(locationMessages);
9   } catch (error) {
10     res.status(404).json({ message: error.message });
11   }
12 }
13
14 export const getIndoorLocation = async (req, res) => {
15   const { id } = req.params;
16   try {
17     const location = await IndoorLocation.findById(id);
18     res.status(200).json(location);
19   } catch (error) {
20     res.status(404).json({ message: error.message });
21   }
22 }
23
24 export const giveLocation = async (req, res) => {
25   const location = req.body;
26
27   const newLocation = new IndoorLocation({ ...location });
28
29   try {
30     await newLocation.save();
31     res.status(201).json(newLocation);
32   } catch (error) {
33     res.status(409).json({ message: error.message });
34   }
35 }
36
37
38
39
40
41
42
43
44
45 export default router;
```





# IT18012552 | M.D.S.M. ANTANY

Software Engineering



## MEASURING PARKING YARDS STANDARD AND VALIDATE SUITAB BEFORE THE YARD REGISTRATION







# RESEARCH QUESTION

How to identify a standard parking yard without time wasting and a without human interaction before registration?

- Reviewing thousands of parking registration forms by a human is time consuming
- Registration process is too complicated
- Inability to audit the parking yard condition without man power (monthly or annually)



# MAIN OBJECTIVE

Identify the parking yard surface type and Quality of the surface before registering to the system as a valid parking yard





## SUB OBJECTIVES

- Identify the parking yard surface type
- Identify the quality of the parking yard surface under the surface type



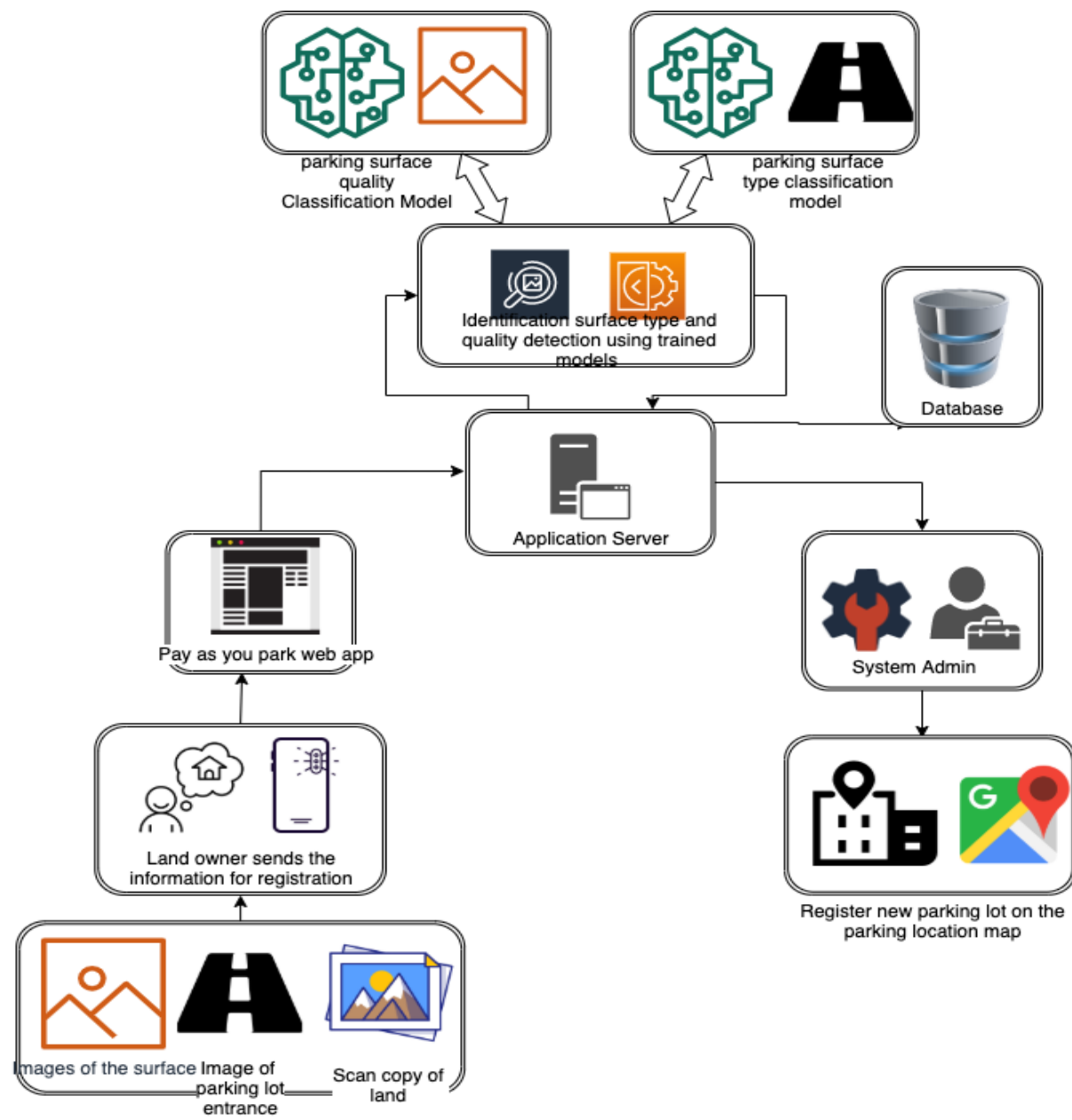
# Research Methodology

- SYSTEM DAIGARM
- TECHNOLOGY AND TECHNIQUES USED



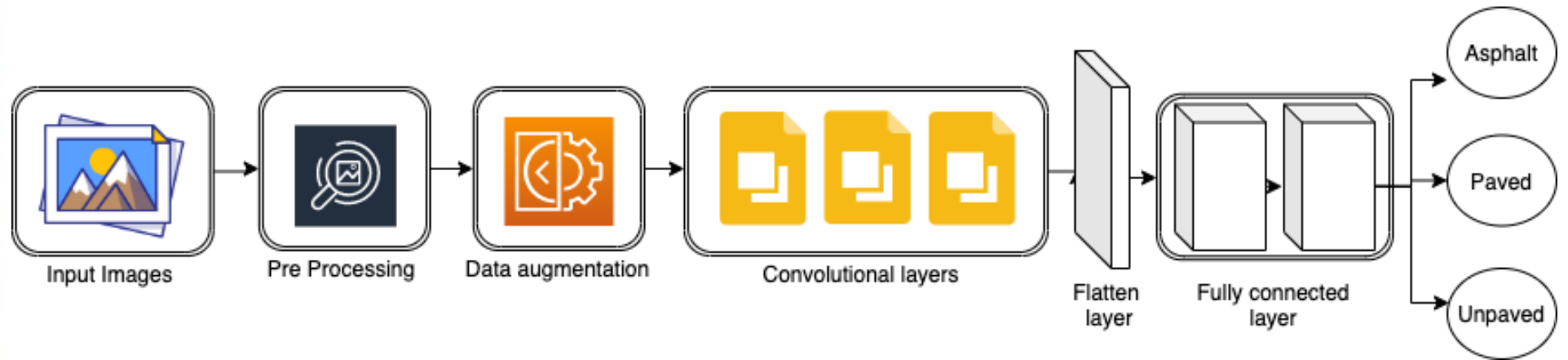


# SYSTEM OVERVIEW DIAGRAM





## SYSTEM FLOWCHART DIAGRAM





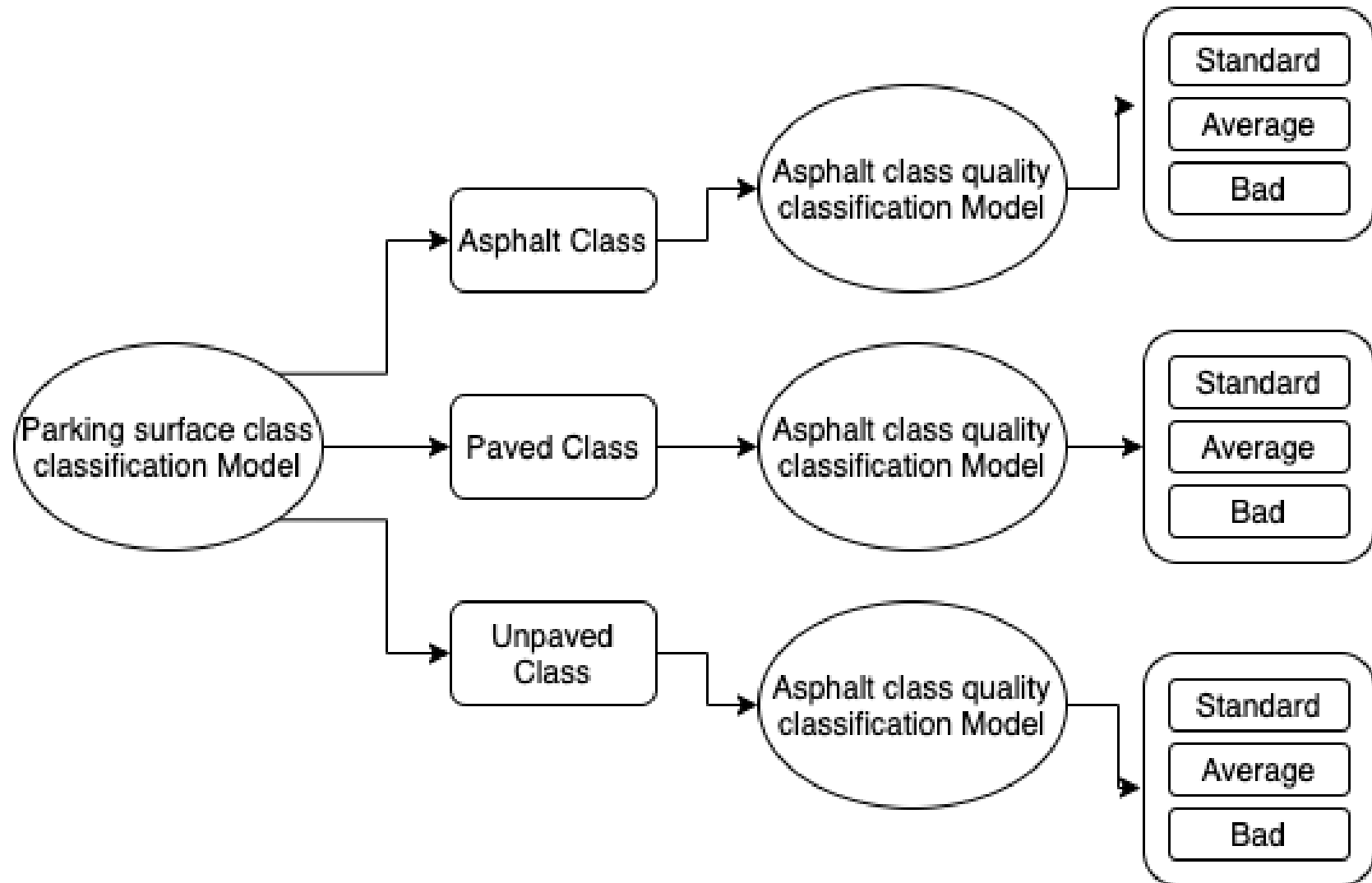


# TECHNOLOGY AND TECHNIQUES USED

- Used Road Traversing Knowledge (RTK) Dataset
- Region of Interest (ROI) is defined as a pre-processing step for each input frame
- The data augmentation consists of increasing and decreasing the brightness in each frame
- Input images are passed to the CNN structure containing three convolution layers and two fully connected layers.
- The flatten layer is used to transform the convolution multi-dimensional tensor into a one-dimensional tensor.
- Model training divided to two parts
  - Parking surface type model
  - Parking surface quality model

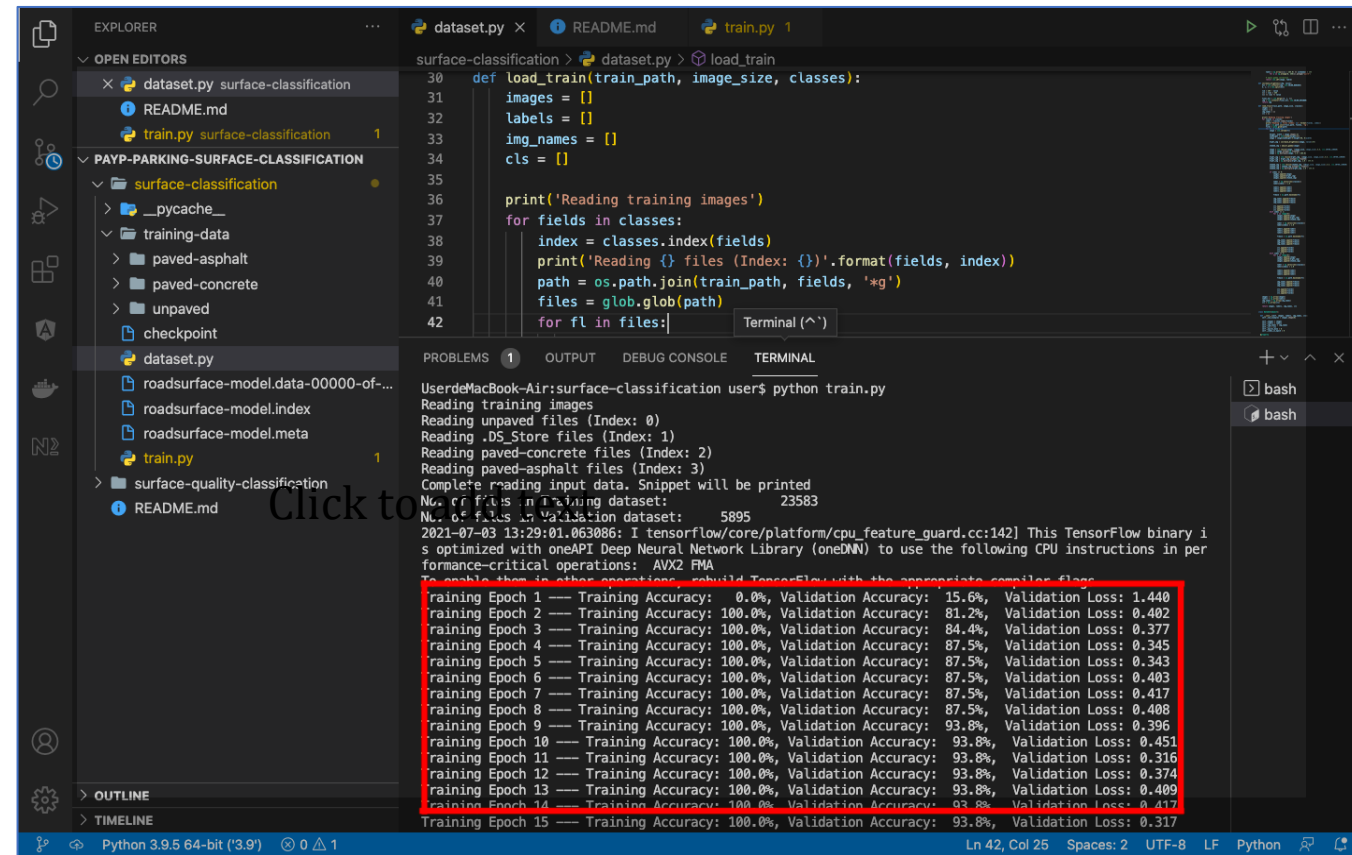


# MODEL ARCHITECTURE





# EVIDENCE FOR COMPLETION



```
def load_train(train_path, image_size, classes):
    images = []
    labels = []
    img_names = []
    cls = []

    print('Reading training images')
    for fields in classes:
        index = classes.index(fields)
        print('Reading {} files (Index: {})'.format(fields, index))
        path = os.path.join(train_path, fields, '*')
        files = glob.glob(path)
        for fl in files:
```

```
Training Epoch 1 --- Training Accuracy: 0.0%, Validation Accuracy: 15.6%, Validation Loss: 1.440
Training Epoch 2 --- Training Accuracy: 100.0%, Validation Accuracy: 81.2%, Validation Loss: 0.402
Training Epoch 3 --- Training Accuracy: 100.0%, Validation Accuracy: 84.4%, Validation Loss: 0.377
Training Epoch 4 --- Training Accuracy: 100.0%, Validation Accuracy: 87.5%, Validation Loss: 0.345
Training Epoch 5 --- Training Accuracy: 100.0%, Validation Accuracy: 87.5%, Validation Loss: 0.343
Training Epoch 6 --- Training Accuracy: 100.0%, Validation Accuracy: 87.5%, Validation Loss: 0.403
Training Epoch 7 --- Training Accuracy: 100.0%, Validation Accuracy: 87.5%, Validation Loss: 0.417
Training Epoch 8 --- Training Accuracy: 100.0%, Validation Accuracy: 87.5%, Validation Loss: 0.408
Training Epoch 9 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.396
Training Epoch 10 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.451
Training Epoch 11 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.316
Training Epoch 12 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.374
Training Epoch 13 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.409
Training Epoch 14 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.417
Training Epoch 15 --- Training Accuracy: 100.0%, Validation Accuracy: 93.8%, Validation Loss: 0.317
```

Parking surface quality classification Model training





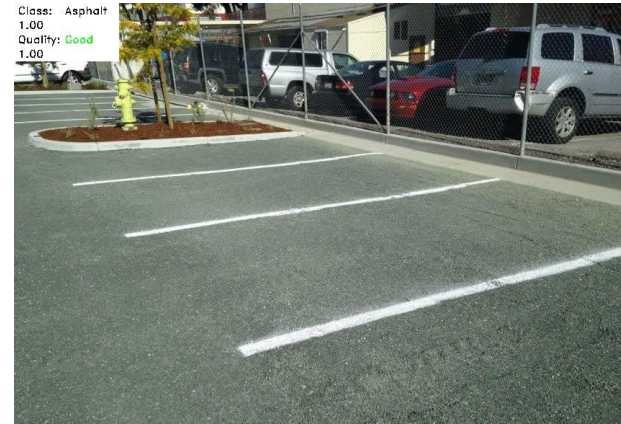
# EVIDENCE FOR COMPLETION



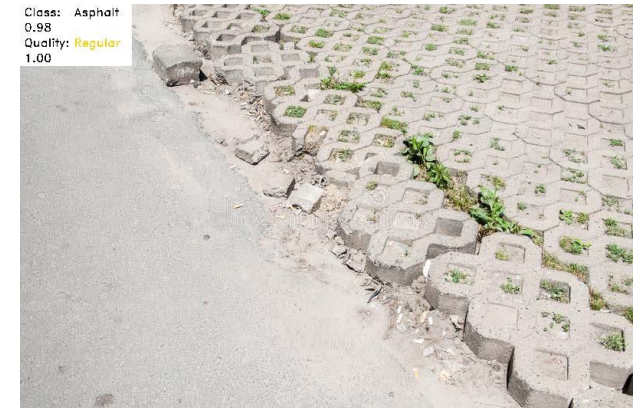
Class: Asphalt  
1.00  
Quality: **Good**  
1.00



Class: Asphalt  
1.00  
Quality: **Good**  
1.00



Class: Asphalt  
0.98  
Quality: **Regular**  
1.00



Class: Asphalt  
0.99  
Quality: **Bad**  
1.00




Class: Asphalt  
0.52  
Quality: **Bad**  
1.00



Surface and quality measurement image output



# EVIDENCE FOR COMPLETION



Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal > paypsurfacemodel

GitHub sachi.antany/payp-surface-detection-model

Open app More

Overview Resources Deploy Metrics Activity Access Settings

Installed add-ons \$0.00/month [Configure Add-ons](#)

There are no add-ons for this app  
You can add add-ons to this app and they will show here. [Learn more](#)

Dyno formation \$0.00/month [Configure Dynos](#)

This app is using free dynos

web gunicorn main:app ON

Collaborator activity [Manage Access](#)

sachi.antany@gmail.com 2 deploys

Latest activity [All Activity](#)

- sachi.antany@gmail.com: Deployed 0c37de12  
Yesterday at 10:07 PM · v4 · [Compare diff](#)
- sachi.antany@gmail.com: Build succeeded  
Yesterday at 10:04 PM · [View build log](#)
- sachi.antany@gmail.com: Deployed 021555c4  
Yesterday at 6:53 PM · v3
- sachi.antany@gmail.com: Build succeeded  
Yesterday at 6:51 PM · [View build log](#)
- sachi.antany@gmail.com: Build failed  
Yesterday at 6:45 PM · [View build log](#)
- sachi.antanv@gmail.com: Build failed

Model deployed on Heroku

# EVIDENCE FOR COMPLETION

A white rectangular form titled 'Sign up' with a red padlock icon above the title. It contains four input fields: 'First Name \*' (with a blue border and a cursor), 'Last Name \*', 'Email Address \*', and 'Password \*' (with an eye icon for toggling visibility). Below these is a 'Repeat Password \*' field. At the bottom are two blue buttons: 'SIGN UP' and 'GOOGLE SIGN IN' (with a white 'G' logo). A link 'ALREADY HAVE AN ACCOUNT? SIGN IN' is at the very bottom.


Parking owner login to pay as you park web application





# EVIDENCE FOR COMPLETION







Pay as you park - Yard Owner

Availability

About

Support





Sachi Antany

LOGOUT

Dashboard

Home

Analytics

Turns

Quick Menu

Yards

Drivers

Transactions

Reports

Notifications

Mail

Feedback


Messages

Staff

Manage

Analytics

Reports



Parliament Exercise Track 2 Parking Lot

a few seconds ago

Parliament Exercise Track 2 Parking Lot


Parking yard quality : Bad

Parliament Exercise Track 2 Parking Lot

Track 1 Walkway Parking & Rest, Maharagama

ADD NEW

DELETE



test 5

10 hours ago

test 5


Parking yard quality : Standard

test 5

address 5

ADD NEW

DELETE



test 6

10 hours ago


test 6

Parking yard quality : Bad

test 6

ADD NEW

DELETE



Administrative Car Park

16 minutes ago

Administrative Car Park

Parking yard quality : Standard

Administrative Car Park

ADD NEW

DELETE

Register a new parking yard

Place name

Address

Max height (ft)

100

Max Width (ft)

100


Max Length (ft)

100

Slot count


Choose file

asp00.jpeg



Veyangoda Post Office

CNN Parking Surface Classification model integrated new parking places registration view and the Owner's dashboard

 **SLIIT**  
FACULTY OF COMPUTING

IT18012552 | M.D.S.M. ANTANY | 2021-198

05-Nov-  
21

61



# REFERENCES

- [1]. M. M. Forrest, Z. Chen, S. Hassan, I. O. Raymond and K. Alinani, "Cost Effective Surface Disruption Detection System for Paved and Unpaved Roads," in IEEE Access, vol. 6, pp. 48634-48644, 2018, doi: 10.1109/ACCESS.2018.2867207.
- [2]. Nienaber, S & Booysen, M.J. (Thinus) & Kroon, Rs. (2015). Detecting Potholes Using Simple Image Processing Techniques and Real-World Footage. 10.13140/RG.2.1.3121.8408.
- [3]. Taluja, Chandan & Thakur, Ritula. (2018). An Intelligent Model for Indian Soil Classification using various Machine Learning Techniques. 2250-3005.
- [4]. Mahmoodi-Eshkaftaki, M., Haghighi, A., & Houshyar, E. (2019). Land Suitability Evaluation using Image Processing based on Determination of Soil Texture-Structure and Soil Features. Soil Use and Management. doi:10.1111/sum.12572
- [5]. Bennett, Jordan. (2019). Smart (Ai) Pothole Detector (Powered by "Tensorflow/TensorRT" on "Google Colab" and or "Jetson Nano" via a Convolutional Artificial Neural Network).





Thank You !



# Q & A

