PARKING MANAGEMENT SYSTEM

SUBMITTED BY

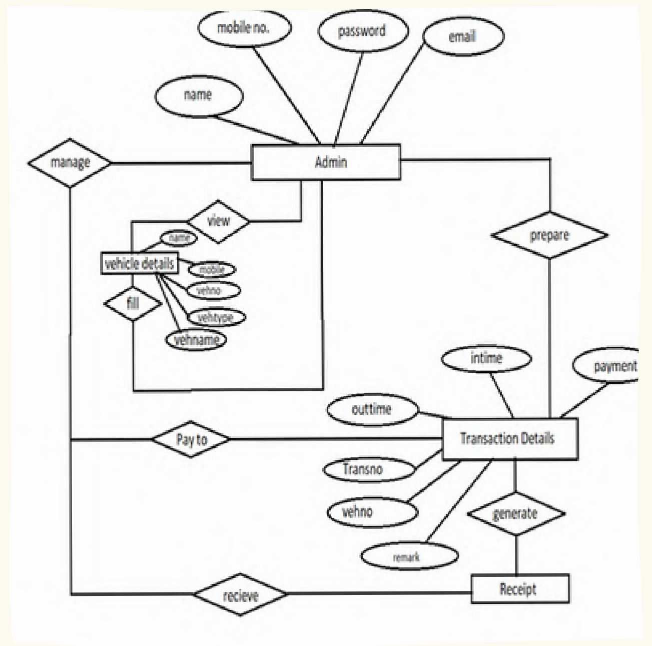
MOHAMMED REHAN SHARIEF MT
(220701515)

SATHISH S
(220701525)

TODAY, WE ARE GOING TO:

1.ABSTARCT

2. ER DIAGRAM



3. PYTHON CODE

ABSTRACT

The PMS aims to reduce the time spent searching for parking, decrease traffic congestion, and minimize environmental impact by streamlining the entire parking process. Additionally, it supports dynamic pricing strategies to maximize revenue for parking operators while offering flexible pricing options for users. The system's modular architecture ensures scalability and adaptability to various types of parking facilities, from small lots to large multi-level garages.

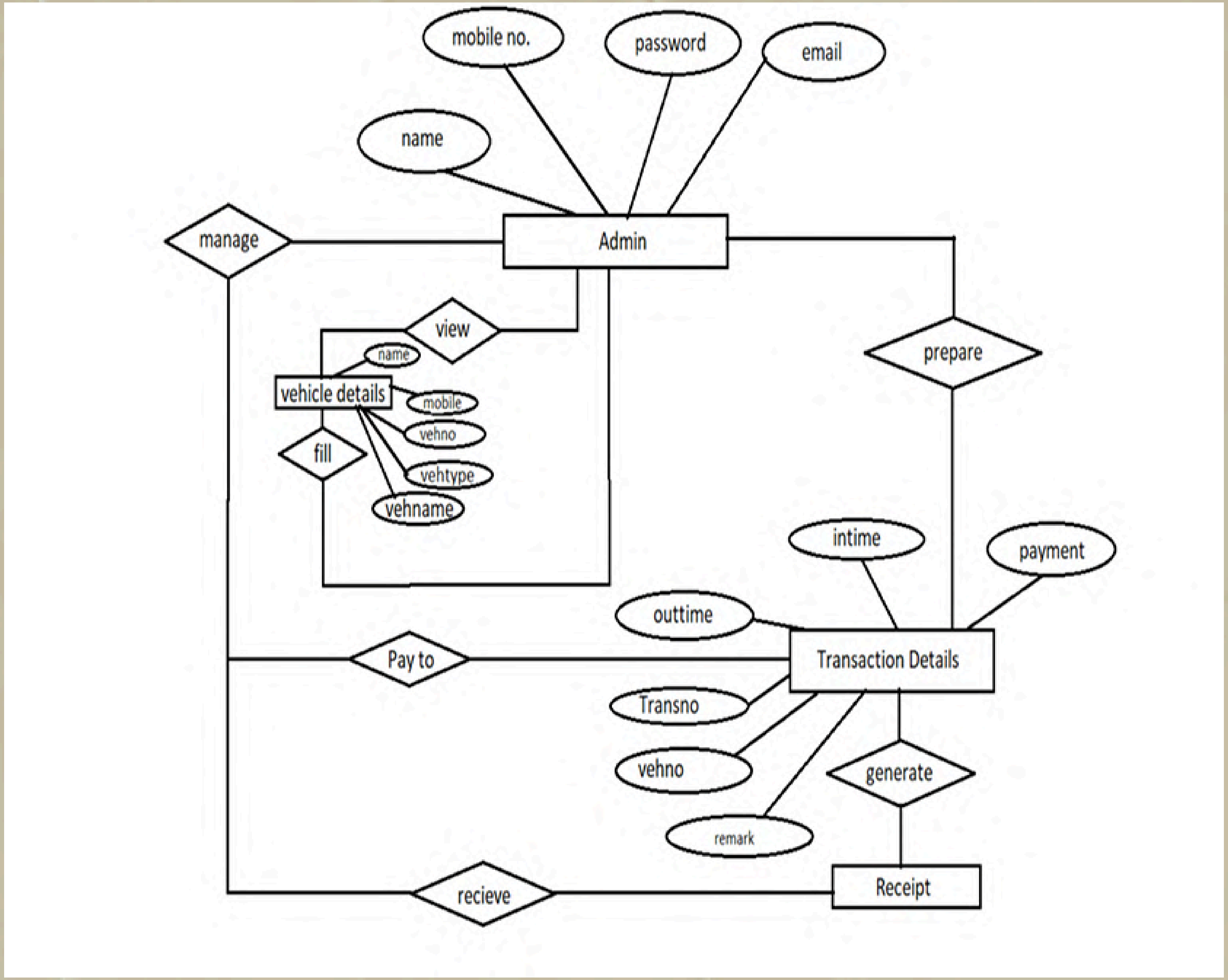
MORE ABOUT ABSTARCT

The PMS aims to reduce the time spent searching for parking, decrease traffic congestion, and minimize environmental impact by streamlining the entire parking process. Additionally, it supports dynamic pricing strategies to maximize revenue for parking operators while offering flexible pricing options for users. The system's modular architecture ensures scalability and adaptability to various types of parking facilities, from small lots to large multi-level garages.

KEY FEATURES

- AUTOMATED EXIT AND ENTRY
- REAL TIME SPACE AVAILABILITY
- DIGITAL PAYMENT PROCESS
- USER-FRIENDLY INTERFACES
- ADVANCED ANALYTICS
- DYNAMIC PRICING STRATAGIES

Entity Relationship Diagram



TECHNOLOGY USED

- PYTHON

  IT IS USED IN THE PROJECT TO TAKE INPUT FROM THE USER.

- MYSQL

  MYSQL PACKAGE IS USED FOR SENDING AND RECEIVING DATA FROM THE DATABASE TO PYTHON AND VICE-VERSA.

SOURCE CODE - PYTHON

```
IMPORT OS
IMPORT PLATFORM
IMPORT MYSQL.CONNECTOR
FROM MYSQL.CONNECTOR IMPORT ERROR


# DATABASE CONNECTION
DEF CREATE_CONNECTION():
TRY:
CONNECTION = MYSQL.CONNECTOR.CONNECT(
HOST="LOCALHOST",
USER="ROOT",
PASSWORD="REHAN@SQL2005",
DATABASE="PARKINGSYSTEM"
)
IF CONNECTION.IS_CONNECTED():
PRINT("CONNECTED TO MYSQL DATABASE")
RETURN CONNECTION
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")
RETURN NONE
```

```
MYDB = CREATE_CONNECTION()
IF NOT MYDB:
EXIT(1)  # EXIT IF CONNECTION IS NOT ESTABLISHED

MYCURSOR = MYDB.CURSOR()


# ENSURE THE TABLES EXIST WITH THE CORRECT SCHEMA
DEF CREATE_TABLES():
TABLE_CREATION_QUERY_PARKMASTER12 = """
CREATE TABLE IF NOT EXISTS PARKMASTER12 (
PID INT PRIMARY KEY,
PNM VARCHAR(30) NOT NULL,
LEVEL VARCHAR(30) NOT NULL,
FREESPACE VARCHAR(30) NOT NULL,
VEHICLENO VARCHAR(30) NOT NULL,
NOD INT NOT NULL,
PAYMENT INT NOT NULL
);
```

```
TABLE_CREATION_QUERY_VEHICLE = """
CREATE TABLE IF NOT EXISTS VEHICLE (
VID INT PRIMARY KEY,
VNM VARCHAR(30) NOT NULL,
DATEOFPUR DATE NOT NULL
);
"""

TRY:
MYCURSOR.EXECUTE(TABLE_CREATION_QUERY_PARKMASTER12)
MYCURSOR.EXECUTE(TABLE_CREATION_QUERY_VEHICLE)
MYDB.COMMIT()
EXCEPT ERROR AS E:
PRINT(F"ERROR CREATING TABLES: {E}")

CREATE_TABLES()


DEF ADD_RECORD():
TRY:
PID = INT(INPUT("ENTER THE PARKING NUMBER: "))
PNM = INPUT("ENTER THE PARKING NAME: ")
LEVEL = INPUT("ENTER LEVEL OF PARKING: ")
FREESPACE = INPUT("IS THERE ANY FREE SPACE OR NOT (YES/NO): ")
VEHICLENO = INPUT("ENTER THE VEHICLE NUMBER: ")
NOD = INT(INPUT("ENTER TOTAL NUMBER OF DAYS FOR PARKING: "))

PAYMENT = MIN(MAX(NOD * 20, 20), 120)  # ADJUST PAYMENT FOR UP TO 6 DAYS

RECORD = (PID, PNM, LEVEL, FREESPACE, VEHICLENO, NOD, PAYMENT)
```

```
SQL = ('INSERT INTO PARKMASTER12 (PID, PNM, LEVEL, FREESPACE, VEHICLENO, NOD, PAYMENT) '
'VALUES (%S, %S, %S, %S, %S, %S, %S)')
MYCURSOR.EXECUTE(SQL, RECORD)
MYDB.COMMIT()
PRINT("RECORD ADDED SUCCESSFULLY.")
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")


DEF REC_VIEW():
PRINT("SELECT THE SEARCH CRITERIA: ")
PRINT("1. PARKING NUMBER")
PRINT("2. PARKING NAME")
PRINT("3. LEVEL NO")
PRINT("4. ALL")
TRY:
CH = INT(INPUT("ENTER THE CHOICE: "))

IF CH == 1:
S = INT(INPUT("ENTER PARKING NO: "))
RL = (S,)
SQL = "SELECT * FROM PARKMASTER12 WHERE PID = %S"
MYCURSOR.EXECUTE(SQL, RL)
ELIF CH == 2:
S = INPUT("ENTER PARKING NAME: ")
RL = (S,)
SQL = "SELECT * FROM PARKMASTER12 WHERE PNM = %S"
MYCURSOR.EXECUTE(SQL, RL)
```

```
ELIF CH == 3:
S = INPUT("ENTER LEVEL OF PARKING: ")
RL = (S,)
SQL = "SELECT * FROM PARKMASTER12 WHERE LEVEL = %S"
MYCURSOR.EXECUTE(SQL, RL)
ELIF CH == 4:
SQL = "SELECT * FROM PARKMASTER12"
MYCURSOR.EXECUTE(SQL)
ELSE:
PRINT("INVALID CHOICE.")
RETURN

RES = MYCURSOR.FETCHALL()

IF CH == 4:
PRINT("DETAILS ABOUT PARKING ARE AS FOLLOWS:")
PRINT("(PARKING ID, PARKING NAME, LEVEL, FREESPACE (Y/N), VEHICLE NO, NO OF DAYS FOR PARKING, PAYMENT)")
FOR X IN RES:
    PRINT(X)
PRINT('TASK COMPLETED')
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")


DEF VEHICLE_DETAIL():
TRY:
```

```
PRINT("VEHICLE RECORD ADDED SUCCESSFULLY.")
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")


DEF VEHICLE_VIEW():
TRY:
VID = INT(INPUT("ENTER THE VEHICLE NUMBER OF THE VEHICLE WHOSE DETAILS ARE TO BE VIEWED: "))
RL = (VID,)
SQL = (
"SELECT PARKMASTER12.PID, PARKMASTER12.PNM, PARKMASTER12.VEHICLENO, VEHICLE.VID, VEHICLE.VNM "
"FROM PARKMASTER12 "
"INNER JOIN VEHICLE ON PARKMASTER12.PID = VEHICLE.VID "
"WHERE VEHICLE.VID = %S"
)
PRINT('THE FOLLOWING ARE THE DETAILS YOU WANTED:')
MYCURSOR.EXECUTE(SQL, RL)
RES = MYCURSOR.FETCHALL()
FOR X IN RES:
PRINT(X)
PRINT('TASK COMPLETED')
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")
```

```
DEF REMOVE():
TRY:
VID = INT(INPUT("ENTER THE VEHICLE NUMBER OF THE VEHICLE TO BE DELETED: "))
RL = (VID,)
SQL = "DELETE FROM VEHICLE WHERE VID = %S"
MYCURSOR.EXECUTE(SQL, RL)
MYDB.COMMIT()
PRINT('REMOVED AS PER THE COMMAND')
EXCEPT ERROR AS E:
PRINT(F"ERROR: {E}")


DEF MENU():
PRINT("ENTER 1: TO ADD PARKING DETAIL")
PRINT("ENTER 2: TO VIEW PARKING DETAIL")
PRINT("ENTER 3: TO ADD VEHICLE DETAIL")
PRINT("ENTER 4: TO REMOVE VEHICLE RECORD")
PRINT("ENTER 5: TO SEE THE DETAILS OF VEHICLE")
TRY:
```

```
INPUT_DT = INT(INPUT("PLEASE SELECT AN ABOVE OPTION: "))
IF INPUT_DT == 1:
ADD_RECORD()
ELIF INPUT_DT == 2:
REC_VIEW()
ELIF INPUT_DT == 3:
VEHICLE_DETAIL()
ELIF INPUT_DT == 4:
REMOVE()
ELIF INPUT_DT == 5:
VEHICLE_VIEW()
ELSE:
PRINT("ENTER CORRECT CHOICE....")
RUNAGAIN()
EXCEPT VALUEERROR:
PRINT("INVALID INPUT. PLEASE ENTER A NUMBER.")


DEF RUNAGAIN():
RUNAGN = INPUT('\NWANT TO RUN AGAIN Y/N: ')
WHILE RUNAGN.LOWER() == 'Y':
IF PLATFORM.SYSTEM() == 'WINDOWS':
OS.SYSTEM('CLS')
```

```
        ELSE:
    OS.SYSTEM('CLEAR')
        MENU()
RUNAGN = INPUT('\NWANT TO RUN AGAIN Y/N: ')


      MENU()
```

THANK YOUUU !!!