DBMS Documentation

Team:

KESHAVA S H(PES1UG21CS276)

Likith M C(PES1UG21CS304)

ABSTRACT:

User Specifications:

User Roles and Access Control:
Define user roles such as administrators, law enforcement officers, and citizens.
Administrators have full access to all functionalities.
Law enforcement officers can add violation details and view related information.
Citizens can view their own violation and payment details.

RTO_Vehicle details:
Law enforcement officers can add new vehicle details when issuing a violation.
Administrators can update and manage vehicle records.
Citizens can view their own vehicle details.

Violation details:

Law enforcement officers can record violation details, including date, time, type, location, and description.

Administrators can access and edit violation records. Citizens can view their own violation history.

Individual details:

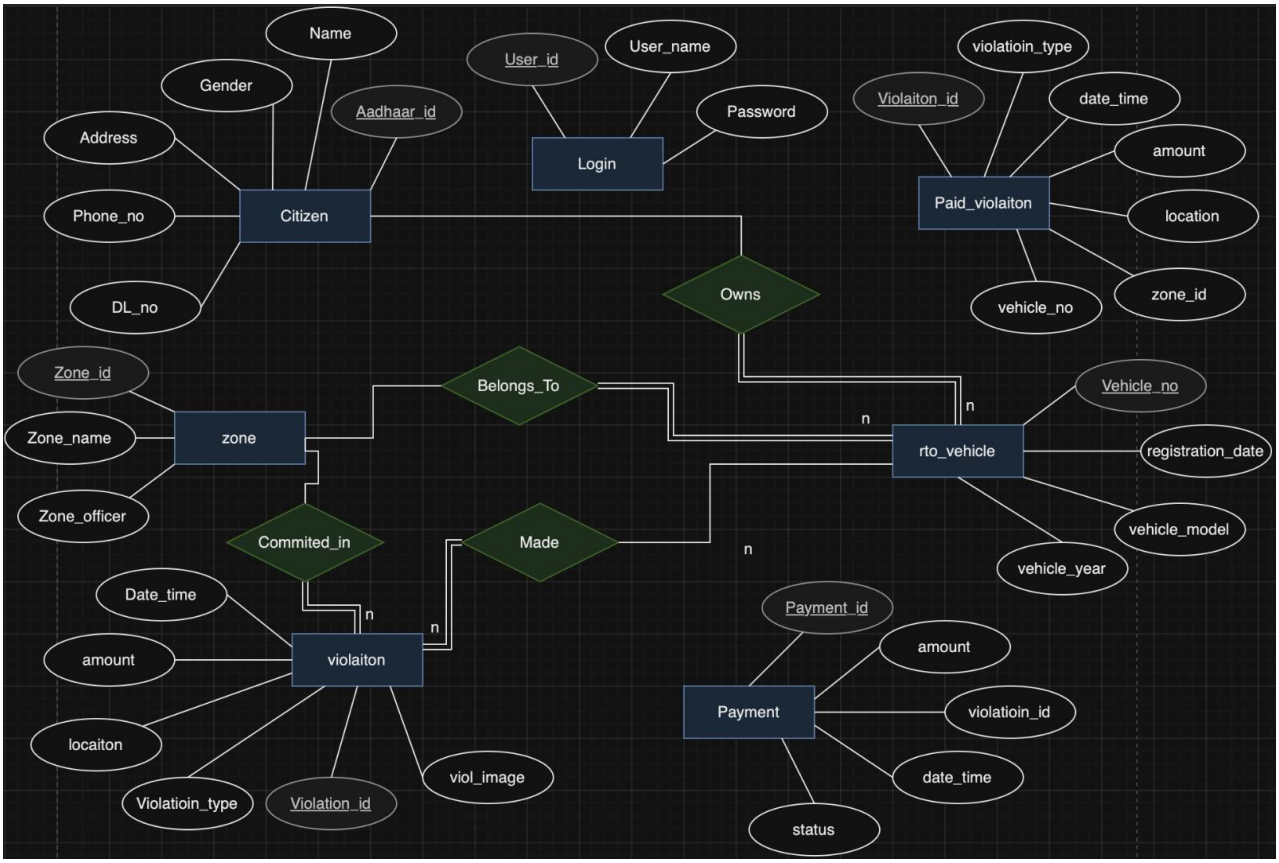
Law enforcement officers can access and edit individual records for both vehicle owners and violators.

Administrators can manage individual records.
Citizens can view and update their personal information.

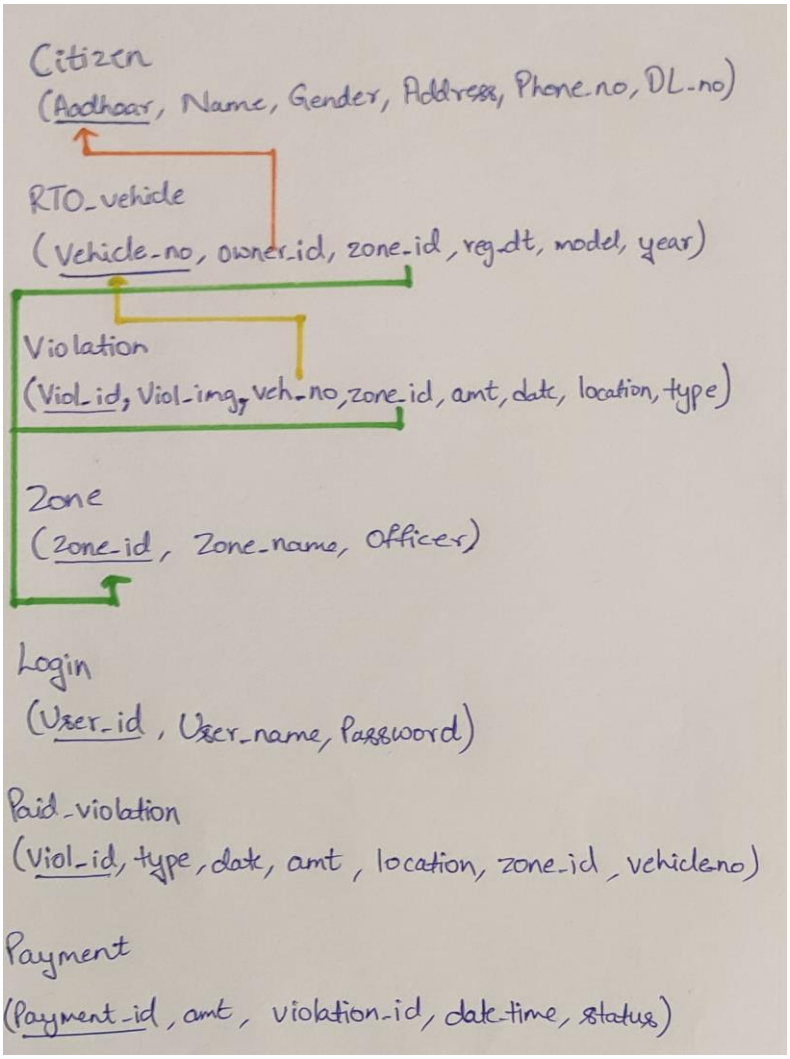
Payment details:
Citizens can make fine payments online through various payment methods. Administrators can view payment records and mark payments as received. Law enforcement officers can check payment status when handling violations.

Zonal details:
Administrators can manage zonal details, including adding and editing zone information. Zonal managers can oversee specific zones and access related data. Law enforcement officers can associate violations with the corresponding zones.

ER DIAGRAM:



RELATIOINAL SCHEMA:



DDL SQL COMMANDS:

```
-- Drop the database if it exists
DROP DATABASE IF EXISTS trafficfine;

-- Create the database
CREATE DATABASE trafficfine;
```

```
USE trafficfine;

CREATE TABLE login (
   u_id INT PRIMARY KEY,
   un VARCHAR(20),
   pass VARCHAR(20)
);

create table RTO_Vehicle(reg_dt DATE,
                vehicle_no varchar(20) PRIMARY KEY NOT NULL,
                vehicle_year INT CHECK (vehicle_year > 1900),
                vehicle_model varchar(30),
                owner_id int,
                zone_id int);

create table violation (violation_id int PRIMARY KEY,
                viol_type varchar(20),
                dt_time TIMESTAMP,
                amount int NOT NULL,
                loc varchar(20),
                zone_id int,
                vehicle_no varchar(20) NOT NULL,
                viol_img longblob);

create table paid_violation(violation_id int PRIMARY KEY,
                   viol_type varchar(20),
                   dt_time TIMESTAMP,
```

```
                amount int NOT NULL,
                loc varchar(20),
                zone_id int,
                vehicle_no varchar(20) NOT NULL);

create table payment (payment_id int PRIMARY KEY,
                violation_id int,
                amount int,
                dt_time TIMESTAMP,
                stat ENUM('success','failure'));

create table citizen(aadhaar_id int PRIMARY KEY,
                c_name varchar(30),
                dl_no varchar(10),
                addr varchar(100),
                gender enum('M','F','O'),
                phno varchar(20));

create table zone (zone_id int PRIMARY KEY,
                zone_name varchar(20),
                zonal_officer varchar(20));

DROP USER IF EXISTS 'project'@'localhost';
CREATE USER 'project'@'localhost' identified by'project123';

GRANT SELECT,INSERT ON trafficfine.login TO 'project'@'localhost';
GRANT INSERT, SELECT, DELETE ON trafficfine.violation TO 'project'@'localhost';
```

GRANT INSERT,SELECT ON trafficfine.payment TO 'project'@'localhost';

GRANT SELECT ON trafficfine.citizen TO 'project'@'localhost';

GRANT SELECT ON trafficfine.rto_vehicle TO 'project'@'localhost';

DROP USER IF EXISTS 'admin'@'localhost';

CREATE USER 'admin'@'localhost' identified by 'likith';

GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;

CRUD OPERATIONS:

Entries to the login table:

INSERT INTO login(u_id, un, pass) VALUES(1, 'likith_machenahalli', 'liki');

INSERT INTO login(u_id, un, pass) VALUES(2, 'keshava', 'keka');

INSERT INTO login(u_id, un, pass) VALUES(3, 'lakshmeesh_bhat', 'laksh');

INSERT INTO login(u_id, un, pass) VALUES(4, 'nitish', 'neethu');

```
mysql> Select * from login;
+------+--------------------+--------+
| u_id | un                 | pass   |
+------+--------------------+--------+
|    1 | likith_machenahalli | liki   |
|    2 | keshava            | keka   |
|    3 | lakshmeesh_bhat    | laksh  |
|    4 | nitish             | neethu |
+------+--------------------+--------+
4 rows in set (0.00 sec)
```

There are 500 values inserted to each the table ‘citizen’ and ‘rto_vehicle’ using the following python code:

```
import random
import string
import pymysql

db = pymysql.connect(
    host="localhost",
    user="admin",
    password="likith",
    database="trafficfine"
)
cursor = db.cursor()

indian_models = ['Hero Honda', 'Bajaj Pulsar', 'Royal Enfield', 'Honda Activa', 'Maruti Suzuki Alto', 'Hyundai Creta', 'Tata Indigo', 'Mahindra Scorpio','RX']

indian_first_names = [
    'Aarav', 'Aishwarya', 'Rahul', 'Priya', 'Sachin', 'Swati', 'Vikram', 'Kiran', 'Anjali', 'Rohit',
    'Neelam', 'Manish', 'Shweta', 'Amit', 'Meera', 'Ganesh', 'Neha', 'Amitabh', 'Madhuri', 'Hrithik'
]

indian_last_names = [
    'Patel', 'Kumar', 'Singh', 'Sharma', 'Gupta', 'Verma', 'Mishra', 'Reddy', 'Jain', 'Mukherjee',
    'Kapoor', 'Yadav', 'Pandey', 'Choudhary', 'Khanna', 'Agarwal', 'Malhotra', 'Mehta', 'Saxena', 'Sinha'
]

street_names = ['Main Street', 'Gandhi Road', 'Shivaji Nagar', 'Rajendra Prasad Avenue', 'Jawaharlal Nehru Lane']
cities =[
    'Bengaluru', 'Mysore', 'Hubli', 'Mangalore', 'Belgaum', 'Davangere', 'Bellary', 'Shimoga', 'Tumkur',
    'Gulbarga', 'Udupi', 'Hassan', 'Bidar', 'Bijapur', 'Raichur', 'Hospet', 'Chitradurga', 'Mandya',
    'Chikkamagaluru', 'Bagalkot', 'Kolar', 'Chamarajanagar', 'Gadag', 'Haveri', 'Karwar', 'Sirsi',
    'Koppal', 'Madikeri', 'Ramanagara'
]

# You can continue to add more cities to the list if needed.

states = ['Karnataka']

genders = ['M', 'F', 'O']

def generate_indian_name():
    first_name = random.choice(indian_first_names)
    last_name = random.choice(indian_last_names)
    full_name = f"{first_name} {last_name}"
    return full_name


def generate_indian_address():
    street = random.choice(street_names)
    city = random.choice(cities)
    state = random.choice(states)
    postal_code = str(random.randint(560000,600000))
    address = f"{street}, {city}, {state} - {postal_code}"
    return address

for _ in range(500):
    reg_year = random.randint(1987, 2023)
    reg_month = random.randint(1, 12)
    reg_day = random.randint(1, 28)
    reg_dt = f'{reg_year}-{reg_month:02d}-{reg_day:02d}'
    vehicle_year = random.randint(1987, 2023)
    vehicle_model = random.choice(indian_models)
    owner_id = random.randint(10**6, 10**7 - 1)
    zone_id = random.randint(1, 6)
    vehicle_no = 'KA' + ''.join(random.choices(string.digits, k=2)) + ''.join(random.choices(string.ascii_uppercase, k=2)) + ''.join(random.choices(string.digits, k=4))

    query = f"INSERT INTO rto_vehicle (reg_dt, vehicle_year, vehicle_model, owner_id, vehicle_no,zone_id) VALUES ('{reg_dt}', {vehicle_year}, '{vehicle_model}', {owner_id},'{vehicle_no}',{zone_id})"
    cursor.execute(query)
    aadhaar_id = owner_id
    c_name = generate_indian_name()
    dl_no = ''.join(random.choices(string.ascii_uppercase, k=3)) + ''.join(random.choices(string.digits, k=7))
    addr = generate_indian_address()
    gender = random.choice(genders)
    phno = str(random.randint(6000000000, 9999999999))

    query = f"INSERT INTO citizen (aadhaar_id, c_name, dl_no, addr, gender, phno) " \
            f"VALUES ({aadhaar_id}, '{c_name}', '{dl_no}', '{addr}', '{gender}', {phno})"
    cursor.execute(query)

db.commit()
db.close()
```

Entries to the Zone Table:

insert into zone(zone_id,zone_name,zonal_officer) values(1,'Bengaluru','Mohmad Saleem');

insert into zone(zone_id,zone_name,zonal_officer) values(2,'Belgavi','Rohan Jagadeesh');

insert into zone(zone_id,zone_name,zonal_officer) values(3,'Hubballi-Dharwad','Ramesh Gokak');

insert into zone(zone_id,zone_name,zonal_officer) values(4,'Shivmogga','Srikanth Kattimani');

insert into zone(zone_id,zone_name,zonal_officer) values(5,'Mangaluru','B.P Dinesh Kumar');

insert into zone(zone_id,zone_name,zonal_officer) values(6,'Mysuru','S Janhavi');

```
mysql> Select * from zone;
+---------+------------------+--------------------+
| zone_id | zone_name        | zonal_officer      |
+---------+------------------+--------------------+
|       1 | Bengaluru        | Mohmad Saleem      |
|       2 | Belgavi          | Rohan Jagadeesh    |
|       3 | Hubballi-Dharwad | Ramesh Gokak       |
|       4 | Shivmogga        | Srikanth Kattimani |
|       5 | Mangaluru        | B.P Dinesh Kumar   |
|       6 | Mysuru           | S Janhavi          |
+---------+------------------+--------------------+
6 rows in set (0.01 sec)
```

Addition of Foreign keys:

ALTER TABLE violation

ADD CONSTRAINT fk1 FOREIGN KEY (zone_id) REFERENCES zone(zone_id)on delete set NULL on update set NULL;

ALTER TABLE violation

ADD CONSTRAINT fk2 FOREIGN KEY (vehicle_no) REFERENCES rto_vehicle(vehicle_no)on delete cascade on update cascade;

ALTER TABLE rto_vehicle

ADD CONSTRAINT fk5 FOREIGN KEY (owner_id) REFERENCES citizen(aadhaar_id);

ALTER TABLE rto_vehicle

ADD CONSTRAINT fk6 FOREIGN KEY (zone_id) REFERENCES zone(zone_id);

LIST OF FUNCTIONALITIES:

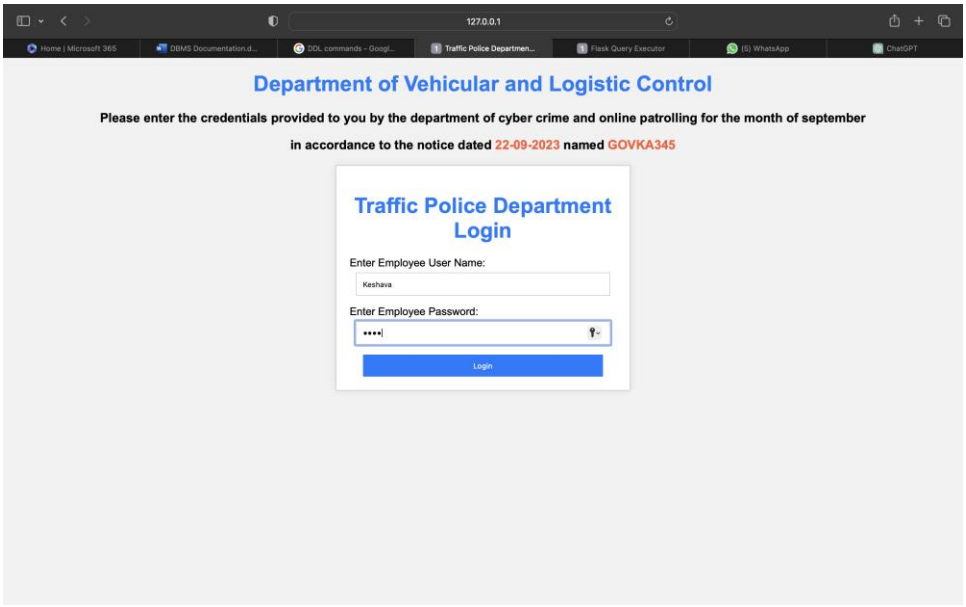
LOGIN PAGE:

Users in the login table(Police Staff) login to get access to be able to add new violations on to the database.

Code:

```python
@app.route('/login_info')
def login_info():
    return render_template('login.html')

@app.route('/login',methods=['POST'])
def login():
    db = pymysql.connect(host="localhost", user="project", password="project123", database="trafficfine")
    cursor = db.cursor()
    if request.method=='POST':
        user_n=str(request.form['u_name'])
        passwd=str(request.form['passwd'])

        query='SELECT * from login where un=%s and pass=%s'
        cursor.execute(query,(user_n,passwd))
        user=cursor.fetchall()
        if(len(user) !=0):
            login_id[0]=1
            time.sleep(1.5)
            return redirect('/')
        else :
            return render_template('wrong_login.html')
    db.commit()
    cursor.close()
    db.close()
```

Login Page:

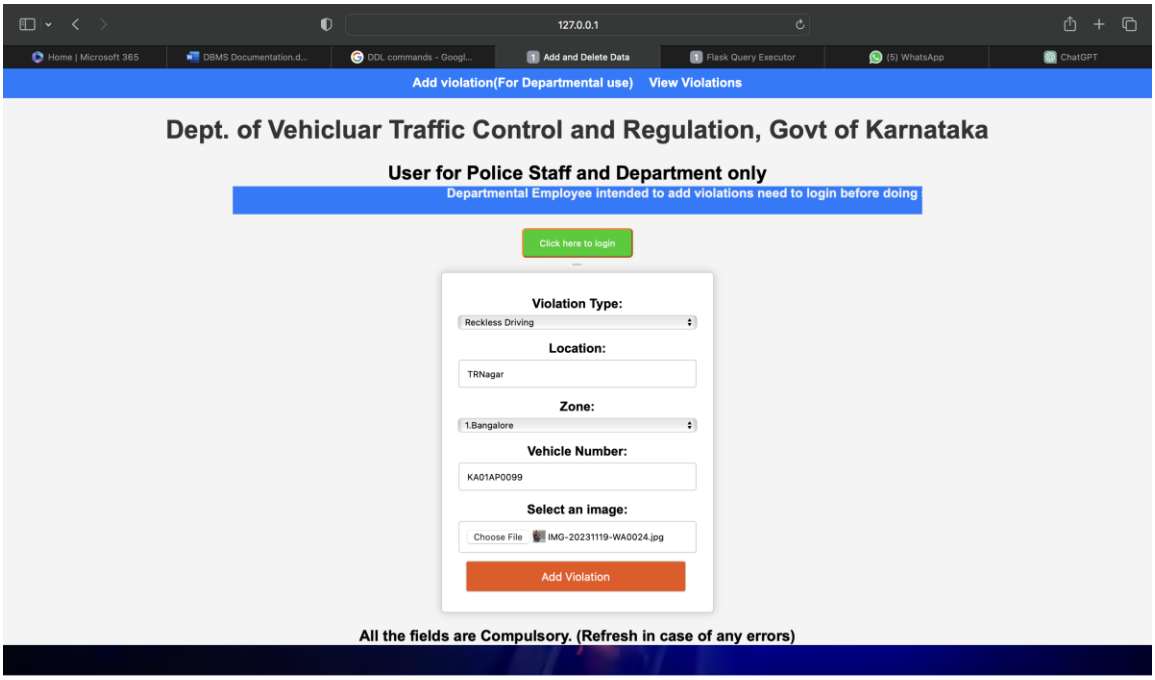


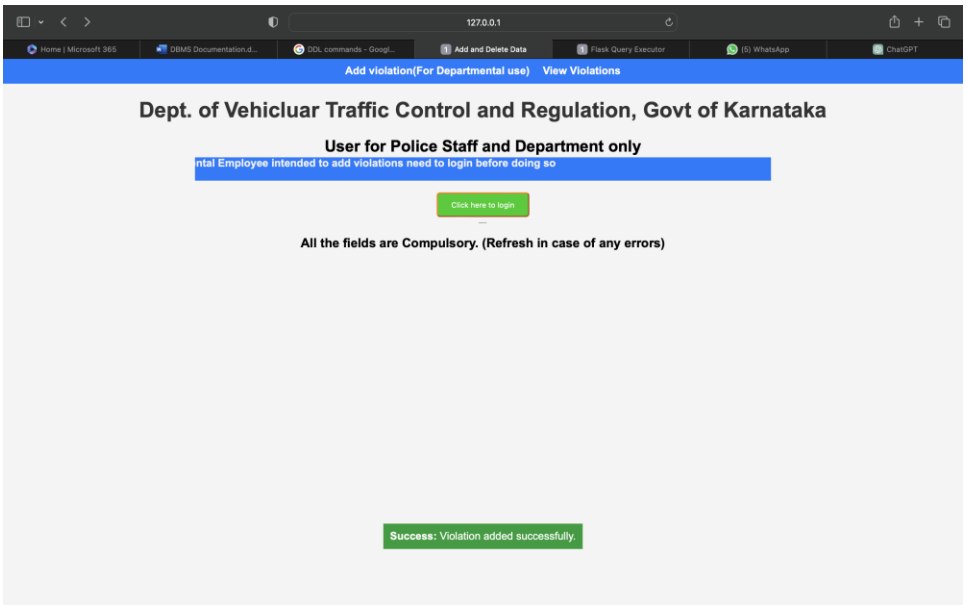
ADDITION OF VIOLATION:

Code:

```
@app.route('/add_data', methods=['POST'])
def add_data():

    db = pymysql.connect(host="localhost", user="project", password="project123", database="trafficfine")
    cursor = db.cursor()
    if request.method == 'POST':
        # Get data from the form
        viol_type=request.form['type']
        print(viol_type,type(viol_type))
        location=request.form['location']
        zone_id=int(request.form['zone_id'][0])
        vehicle_no = request.form['vehicle_no']
        if 'viol_img' in request.files:
            image=request.files['viol_img']
            if image.filename != '':
                image_data = image.read()
        fines = {}
        fines['Signal Jump']=500
        fines['Wrong Parking']=750
        fines['Wrong way driving']=1000
        fines['No helmet']=500
        fines['Triple riding']=1750
        fines['Using Mobile']=750
        fines['Reckless Driving'] = 2000
        a=random.randint(1000,10000)
        while(a in violist):
            a=random.randint(1000,10000)
    if login_id[0]==1:
        try:
            insert_query = "INSERT INTO violation (violation_id,viol_type,dt_time,amount,loc,zone_id,vehicle_no,viol_img) VALUES (%s,%s,NOW(),%s,%s, %s,%s,%s)"
            cursor.execute(insert_query, (a,viol_type,fines[viol_type], location,zone_id,vehicle_no,image_data))
            db.commit()
            cursor.close()
            db.close()
            time.sleep(1.5)
            return redirect('/')
        except Exception as e:
            return render_template("vehicle_not_found.html")

    else:
        db.commit()
        cursor.close()
        db.close()
        return redirect('/login_info')
```

FrontEnd Page:



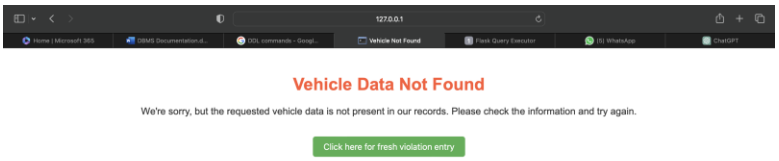


The violations added here go to violation table

```
mysql> select violation_id, viol_type, dt_time, amount, loc, zone_id, vehicle_no from violation;
+--------------+-----------------+---------------------+--------+-----------+---------+------------+
| violation_id | viol_type       | dt_time             | amount | loc       | zone_id | vehicle_no |
+--------------+-----------------+---------------------+--------+-----------+---------+------------+
|         3650 | Reckless Driving | 2023-11-23 21:20:54 |   2000 | TRNagar   |       1 | KA01AP0099 |
|         4478 | Triple riding    | 2023-11-23 17:55:36 |   1750 | GiriNagar |       1 | KA00EU8768 |
|         5449 | Using Mobile     | 2023-11-23 21:28:22 |    750 | Madivala  |       1 | KA01AP0099 |
|         6158 | No helmet        | 2023-11-23 21:29:13 |    500 | KR PURAM  |       1 | KA01IE3258 |
|         9038 | Signal Jump      | 2023-11-23 21:27:07 |    500 | MG Road   |       6 | KA01KS1120 |
+--------------+-----------------+---------------------+--------+-----------+---------+------------+
5 rows in set (0.00 sec)
```

Note: Image not shown in the database as it is converted to binary values which is very huge

All the vehicles present in the ‘rto_vehicle’ table can be added to the violation table as all these vehicles are the registered vehicles
On trying to add a wrond vehicle no we get an error:

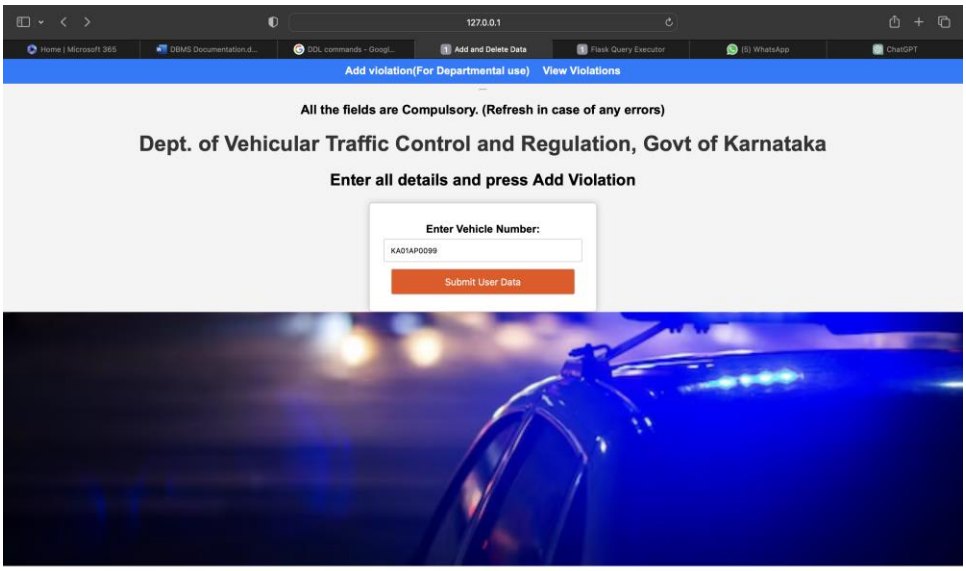


RETRIEVAL OF VIOLAITONS COMMITED BY A PARTICULAR VEHICLE:

Code:

```
@app.route("/search_violations", methods=["POST"])
def search_violations():
    import base64
    from PIL import Image
    from io import BytesIO

    db = pymysql.connect(
        host="localhost",
        user="project",
        password="project123",
        database="trafficfine",
        cursorclass=pymysql.cursors.DictCursor
    )


    vehicle_number = request.form["vehi_no"]
    with db.cursor() as cursor:
        query = "SELECT * FROM violation WHERE vehicle_no = %s"
        cursor.execute(query, (vehicle_number,))
        violations = cursor.fetchall()

        for violation in violations:
            image_data = violation.get("viol_img")

            if image_data:
                encoded_image = base64.b64encode(image_data).decode("utf-8")
                violation["encoded_image"] = encoded_image

    violations_list = list(violations)

    if len(violations_list)>0:
        return render_template("violations.html", violations=violations_list)
    else:
        return render_template("no_viol.html")
```

FrontEnd:



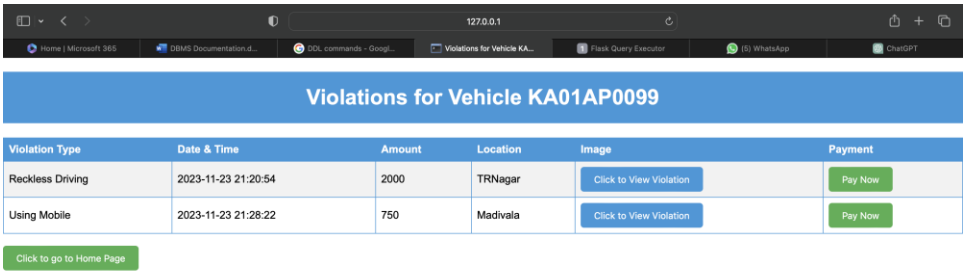


Image Display:

Code:

```python
@app.route('/show_image', methods=['POST'])
def show_image():
    # Get the violation_id from the form
    violation_id = request.form['violation_id']
    print(violation_id)
    # Query the database to get the image data
    connection = pymysql.connect(host="localhost", user="admin", password="likith", database="trafficfine")
    try:
        with connection.cursor() as cursor:
            # Assuming your table is named 'violations' and has columns 'violation_id' and 'violation_img'
            query = f"SELECT viol_img FROM violation WHERE violation_id = {violation_id}"
            cursor.execute(query)
            result = cursor.fetchone()
            if result:
                # Extract image data from the result
                image_data = result

                # Convert binary image data to base64 for rendering in HTML
                image_base64 = base64.b64encode(image_data[0]).decode('utf-8')

                # Render the HTML page with the image
                return render_template('show_image.html', image_base64=image_base64)
            else:
                return render_template('error.html', message='Violation ID not found')

    except Exception as e:
        return render_template('error.html', message=f'Error: {str(e)}')
    finally:
        connection.close()
```

On payment for a particular entry, it will be removed from the ‘violation’ table and will be added to the ‘paid_violation’ table and also the payment details will be added to the ‘payment’ table:

Code:

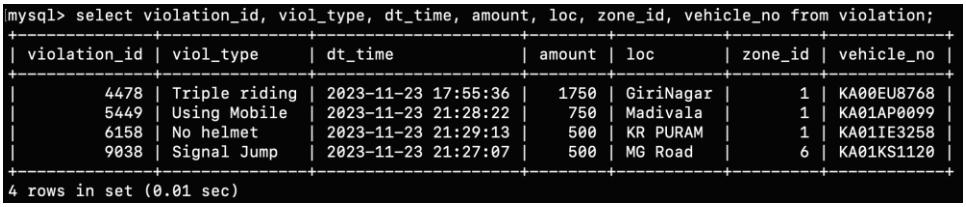
```python
@app.route("/add_payment", methods=['POST'])
def add_payment():
    db = pymysql.connect(host="localhost", user="project", password="project123", database="trafficfine")
    cursor = db.cursor()
    if request.method=='POST':
        violation_id=int(request.form["violation_id"])
        amount=int(request.form["amount"])
        vehi_no=request.form['vehi_no']
        a=random.randint(1000,10000)
        while(a in paymentlist):
            a=random.randint(1000,10000)
        print(vehi_no,type(vehi_no))
        print(a,type(a))
        print(violation_id,type(violation_id))
        print(amount,type(amount))
        query="INSERT INTO payment (payment_id,violation_id,amount,dt_time,stat) VALUES (%s,%s,%s,NOW(),'success')"
        cursor.execute(query,(a,violation_id,amount))
        db.commit()
        time.sleep(1.5)
        cursor.close()
        db.close()
        html="<h1 style='color: green;'>Successful</h1><p style='color: blue;'>Payment of rupees "+str(amount)+" is received</p><form id='redirect_form' method='POST' action='/search_violations'><input
    return (html)
```

Trigger used to add to ‘paid_violation’:

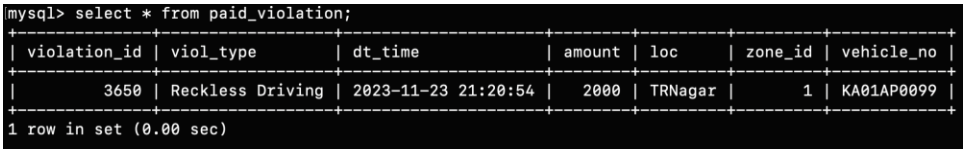
DELIMITER //

```
CREATE TRIGGER copy_to_paid_violation

AFTER INSERT ON payment

FOR EACH ROW

BEGIN

   -- Copy the corresponding entry from violation to paid_violation

   INSERT INTO paid_violation (violation_id, viol_type, dt_time, amount, loc, zone_id, vehicle_no)

   SELECT violation_id, viol_type, dt_time, amount, loc, zone_id, vehicle_no

   FROM violation

   WHERE violation_id = NEW.violation_id;

   -- Delete the entry from the violation table

   DELETE FROM violation

   WHERE violation_id = NEW.violation_id;

END;

//

DELIMITER ;
```

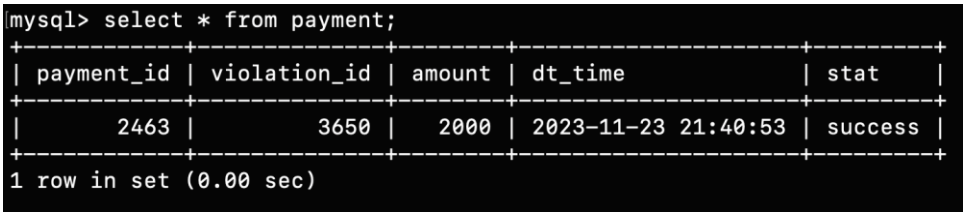
After paying for the first violation in the above image:



```
mysql> select violation_id, viol_type, dt_time, amount, loc, zone_id, vehicle_no from violation;
+-------------+--------------+---------------------+--------+-----------+---------+------------+
| violation_id | viol_type     | dt_time             | amount | loc       | zone_id | vehicle_no |
+-------------+--------------+---------------------+--------+-----------+---------+------------+
|         4478 | Triple riding | 2023-11-23 17:55:36 |   1750 | GiriNagar |       1 | KA00EU8768 |
|         5449 | Using Mobile  | 2023-11-23 21:28:22 |    750 | Madivala  |       1 | KA01AP0099 |
|         6158 | No helmet     | 2023-11-23 21:29:13 |    500 | KR PURAM  |       1 | KA01IE3258 |
|         9038 | Signal Jump   | 2023-11-23 21:27:07 |    500 | MG Road   |       6 | KA01KS1120 |
+-------------+--------------+---------------------+--------+-----------+---------+------------+
4 rows in set (0.01 sec)
```



```
mysql> select * from paid_violation;
+-------------+----------------+---------------------+--------+---------+---------+------------+
| violation_id | viol_type       | dt_time             | amount | loc     | zone_id | vehicle_no |
+-------------+----------------+---------------------+--------+---------+---------+------------+
|         3650 | Reckless Driving | 2023-11-23 21:20:54 |   2000 | TRNagar |       1 | KA01AP0099 |
+-------------+----------------+---------------------+--------+---------+---------+------------+
1 row in set (0.00 sec)
```



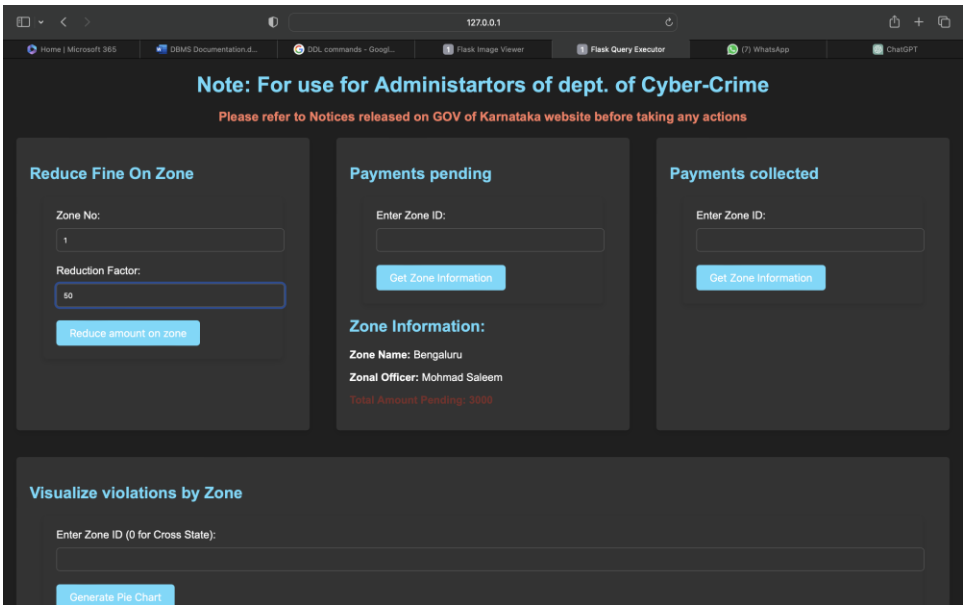
```
mysql> select * from payment;
+------------+--------------+--------+---------------------+---------+
| payment_id | violation_id | amount | dt_time             | stat    |
+------------+--------------+--------+---------------------+---------+
|       2463 |         3650 |   2000 | 2023-11-23 21:40:53 | success |
+------------+--------------+--------+---------------------+---------+
1 row in set (0.00 sec)
```

FRONT END FOR ADMIN:

Code:

```
@app.route('/reducefine',methods=['POST'])
def reduce():
    if request.method=='POST':
        zone=request.form['zone_no']
        percentage=request.form['reduction_factor']
        query="CALL ReduceAmountInZone(%s,%s);"
        connection = pymysql.connect(**db_config)
        cursor = connection.cursor()
        cursor.execute(query,(zone,percentage))
        connection.commit()
        connection.close()
    time.sleep(2)
    return render_template('admin_home.html')

@app.route('/pendingzoneinfo', methods=['POST'])
def zoneinfopending():
    if request.method == 'POST':
        try:
            zone_id = int(request.form['zone_id'])
            query = """
                SELECT zone_name, zonal_officer, calculate_total_amount_pending(%s) as total_amount
                FROM zone
                WHERE zone_id = %s
            """
            connection = pymysql.connect(**db_config)
            cursor = connection.cursor()
            cursor.execute(query, (zone_id, zone_id))
            zone_info = cursor.fetchone()
            print(zone_info,type(zone_info))
            if zone_info:
                # Construct the response
                response=zone_info
                return render_template('admin_home.html', zone_info_pending=response)
            else:
                return render_template('admin_home.html', zone_info_pending=None)

        except Exception as e:
            print(f"Error: {e}")
            return render_template('your_template_name.html', zone_info_pending=None)
@app.route('/paidzoneinfo', methods=['POST'])
def zoneinfopaid():
    if request.method == 'POST':
        try:
            zone_id = int(request.form['zone_id'])
            query = """
                SELECT zone_name, zonal_officer, calculate_total_amount_paid(%s) as total_amount
                FROM zone
                WHERE zone_id = %s
            """
            connection = pymysql.connect(**db_config)
            cursor = connection.cursor()
            cursor.execute(query, (zone_id, zone_id))
            zone_info = cursor.fetchone()
            print(zone_info,type(zone_info))
            if zone_info:
                # Construct the response
                response=zone_info
                return render_template('admin_home.html', zone_info_paid=response)
            else:
                return render_template('admin_home.html', zone_info_paid=None)

        except Exception as e:
            print(f"Error: {e}")
            return render_template('your_template_name.html', zone_info_paid=None)
```

Total pending payments of zone 1:



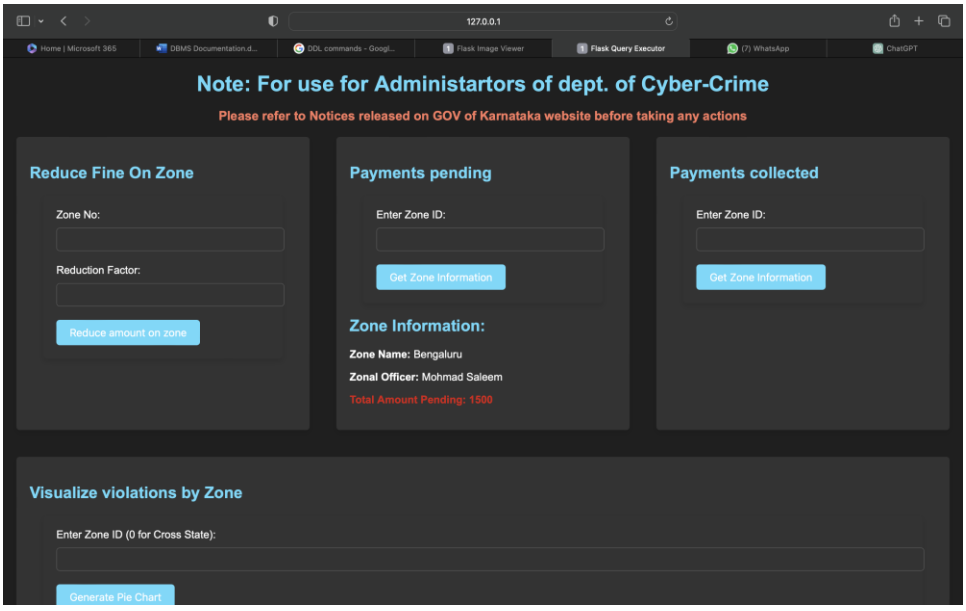
Function used to calculate the total amount pending:

```
DELIMITER//
CREATE FUNCTION calculate_total_amount_pending(zone_id_param INT)
RETURNS INT
READS SQL DATA
BEGIN
   DECLARE total_amount INT;
   SELECT SUM(amount)
   INTO total_amount
   FROM violation
   WHERE zone_id = zone_id_param;
   IF total_amount IS NULL THEN
      SET total_amount = 0;
   END IF;
   RETURN total_amount;
END //
DELIMITER ;
```

Procedure used to apply the reduction factor:

```
DELIMITER //
CREATE PROCEDURE ReduceAmountInZone(IN zoneIDParam INT, IN percentageParam INT)
BEGIN
    DECLARE reductionFactor DECIMAL(5,2);
    -- Convert percentage to a decimal factor
    SET reductionFactor = percentageParam / 100.0;
    -- Update the amount in the paid_violation table for the specified zone
    UPDATE violation
    SET amount = amount - (amount * reductionFactor)
    WHERE zone_id = zoneIDParam;
END //
DELIMITER ;
```

Total payments pending in zone 1 after applying the reduction factor:

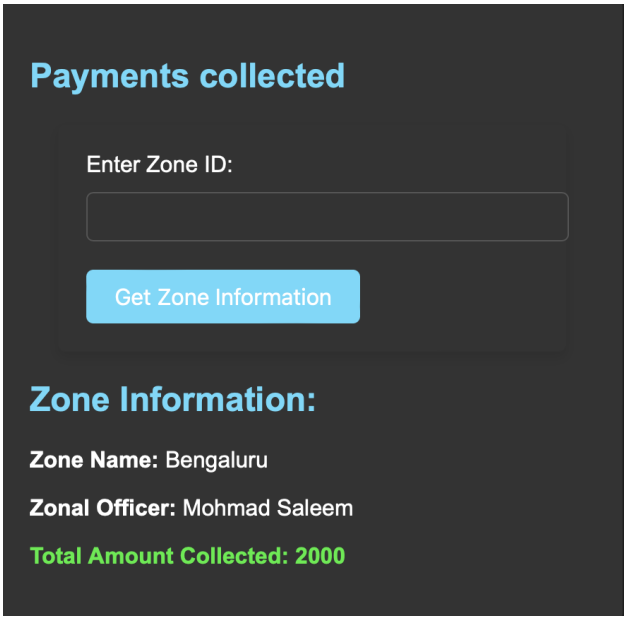


Payments already collected from zone 1:

Function used to calculate the total amount already paid from a particular zone:

```
DELIMITER //
CREATE FUNCTION calculate_total_amount_paid(zone_id_param INT)
RETURNS INT
READS SQL DATA
BEGIN
   DECLARE total_amount INT;
   SELECT SUM(amount)
   INTO total_amount
   FROM paid_violation
   WHERE zone_id = zone_id_param;
   IF total_amount IS NULL THEN
      SET total_amount = 0;
   END IF;
   RETURN total_amount;
END //
DELIMITER ;
```

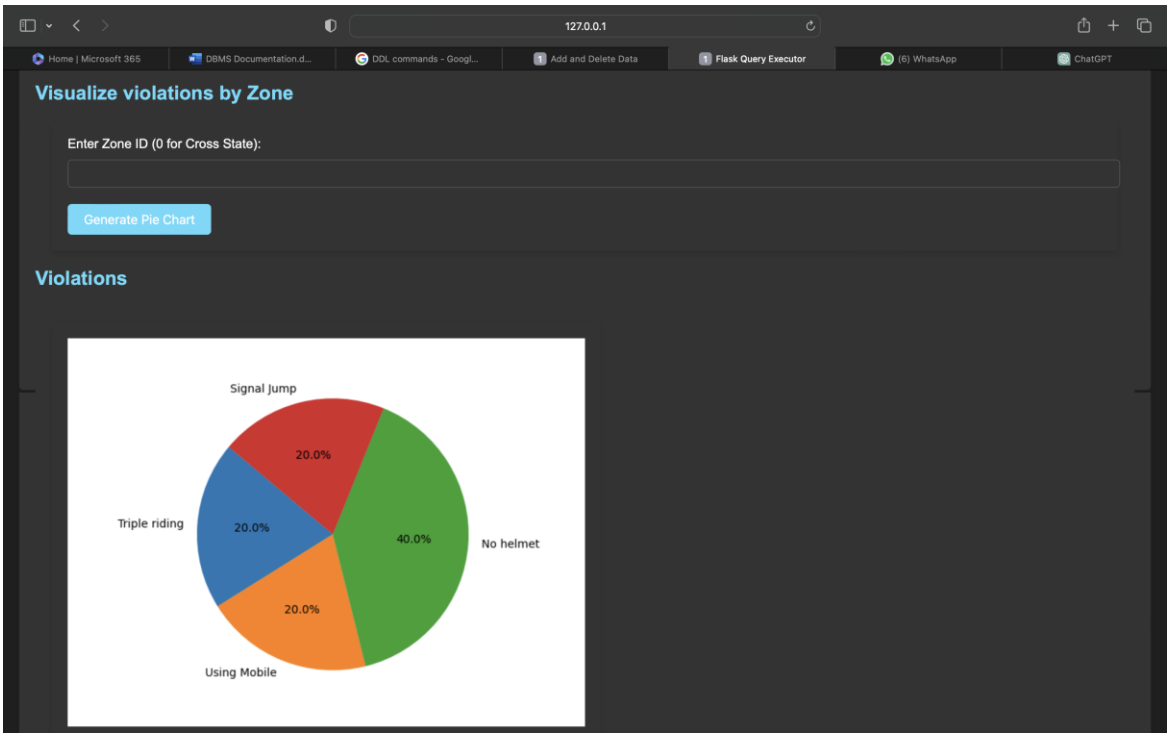
Total amount paid from zone 1:



PIE CHART OF THE DISTRIBUTION OF DIFFERENT TYPES OF VIOLATIONS ACCROSS STATE:

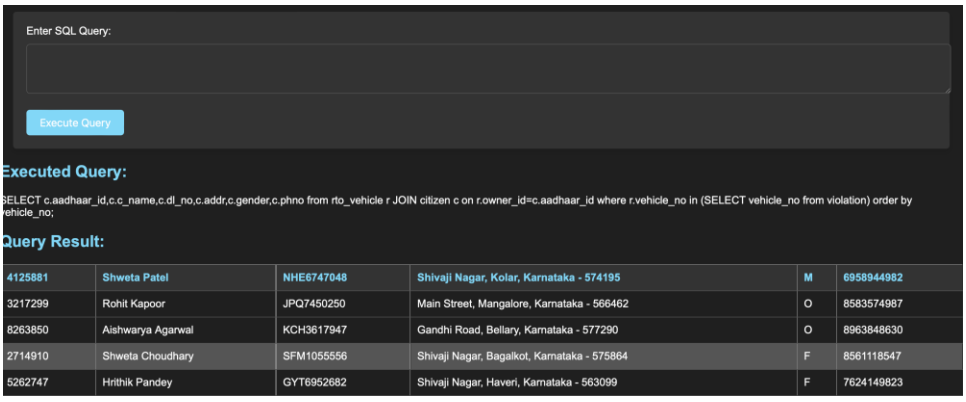
CODE:

```
# Function to get total violations by type for a given zone or all zones
def get_total_violations(zone_id):
    if zone_id == 0:
        query = """
            SELECT viol_type, COUNT(*) as total_count
            FROM violation
            GROUP BY viol_type;
        """
        connection = pymysql.connect(**db_config)
        cursor=connection.cursor()
        result = cursor.execute(query)
    else:
        query = """
            SELECT viol_type, COUNT(*) as total_count
            FROM violation
            WHERE zone_id = %s
            GROUP BY viol_type;
        """
        connection = pymysql.connect(**db_config)
        cursor=connection.cursor()
        result = cursor.execute(query, (zone_id))

    result = cursor.fetchall()
    cursor.close()
    connection.close()

    print(result)
    return result

@app.route('/pie_chart', methods=['POST'])
def generate_pie_chart():
    zone_id = int(request.form['zone_id'])
    data = get_total_violations(zone_id)

    # Extract data for plotting
    labels = [row[0] for row in data]
    values = [row[1] for row in data]

    # Plotting the pie chart using the 'Agg' backend
    plt.switch_backend('Agg')
    plt.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle

    # Convert plot to base64 image for embedding in HTML
    img_buffer = BytesIO()
    plt.savefig(img_buffer, format='png')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    plt.close()

    return render_template('admin_home.html', zone_id=zone_id, pie_chart=img_str)


if __name__ == '__main__':
    app.run(debug=True,port=8080)
```



EXAMPLE FOR EXECUTION OF DIFFERENT QUERIES BY THE ADMIN:

- To get details of the citezen who commited the violation:



4125881	Shweta Patel	NHE6747048	Shivaji Nagar, Kolar, Karnataka - 574195	M	6958944982
3217299	Rohit Kapoor	JPQ7450250	Main Street, Mangalore, Karnataka - 566462	O	8583574987
8263850	Aishwarya Agarwal	KCH3617947	Gandhi Road, Bellary, Karnataka - 577290	O	8963848630
2714910	Shweta Choudhary	SFM1055556	Shivaji Nagar, Bagalkot, Karnataka - 575864	F	8561118547
5262747	Hrithik Pandey	GYT6952682	Shivaji Nagar, Haveri, Karnataka - 563099	F	7624149823

- To get DL details of all the violators:

- Citizen who have their pending amount more than 1000