AUTOMATED UNIVERSITY GARAGE MANAGEMENT SYSTEM (AUGMS)

TEAM MEMBERS:

1- ALY HASSAN
2- HAMZA EL KODSH
3- MOHAMED EHAB
4- ALAA SHAABAN
5- KENZY ZEDAN

Deliverable #1 Template – Database Project (Fall 2025)

1. English Requirements (Business Rules)

The Automated University Garage Management System (AUGMS) is a smart parking management system that automates vehicle registration, entry/exit verification, real-time parking spot monitoring using existing sensors, service requests (EV charging, car cleaning), and generates operational reports. It serves three main user types: Students/Faculty (Parking Users), Garage Administrators, and University Management.

> System requirements here:

R1 — User Types and Hierarchy

- There are three main actor types: ParkingUser (Student or Faculty), GarageAdmin, and UniversityManager.
- A ParkingUser can be either a Student or a Faculty member (disjoint specialization of User).
- All users are controlled by UniversityManager (a UniversityManager can enable/disable the garage access).

R2 — Vehicle Registration

- Each ParkingUser may register one or more vehicles.
- A vehicle must have a unique licensePlate across the entire system.
- A vehicle must be approved by a GarageAdmin before it can access the garage.
- Only approved vehicles are granted entry.

R3 — Parking Garage and Spots

- The university has one or more ParkingGarages.
- Each ParkingGarage contains one or more parkingSpot(s).
- Each parkingSpot is monitored by exactly one OccupancySensor.
- A parkingSpot is either available or occupied at any time.

R4 — Entry and Exit Gates

- Each ParkingGarage has one or more Gates (entry/exit).
- Each Gate is monitored by one GateSensor that reads RFID.

R5 — Automated Entry/Exit

- When a vehicle approaches a gate, the GateSensor reads the RFID sticker returning the car's license plate.
- The system checks if the license plate belongs to an approved and access-enabled vehicle.
- If valid and accessStatus = true → gate opens and a Logging entry is created.
- If invalid or access disabled → access denied and incident logged.

R6 — Service Requests

- A registered ParkingUser can submit zero or more ServiceRequests (EV charging or car cleaning).
- Each ServiceRequest is of one ServiceType.
- A ServiceRequest must be approved and completed by staff → generates an Invoice.

R7 — Invoicing and Payment

- Every completed service generates exactly one Invoice.
- An Invoice is paid using one Payment (supports multiple payment methods).

R8 — Access Control

- UniversityManager can disable/enable garage access for any ParkingUser or GarageAdmin at any time (accessStatus flag).
- Disabled users/admins cannot enter the garage or perform admin actions.

R9 — Reporting

- The system automatically generates DailyOperationsReport every day.
- Weekly and monthly SummaryReports are generated on schedule.
- UsageReport can be requested on-demand by UniversityManager.

R10 — Occupancy Constraints

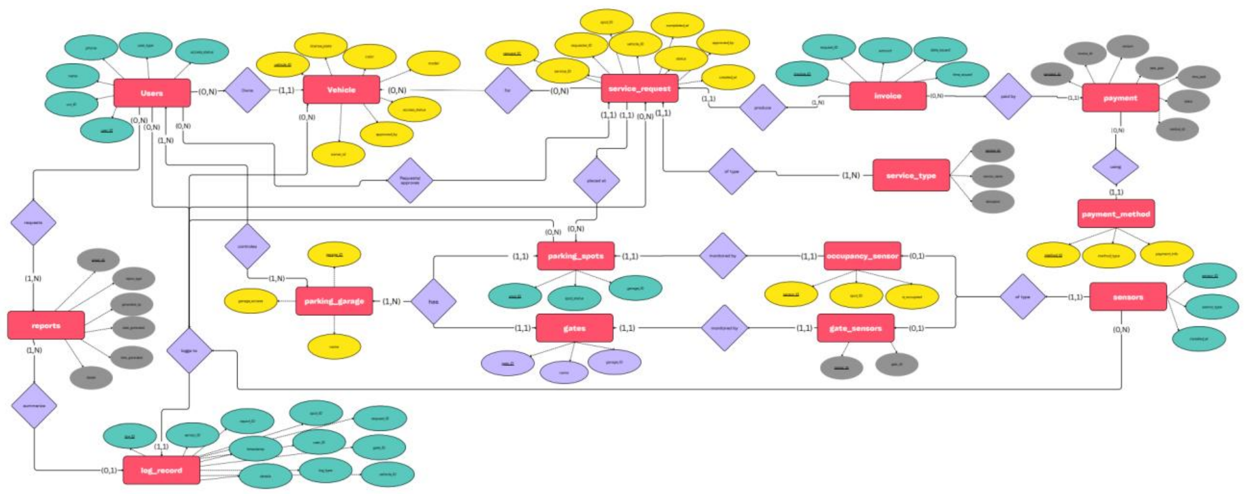
- Total occupied spots cannot exceed total spots in a garage.
- When occupancy reaches 100%, garage status becomes “FULL” and displayed accordingly.

R11 — Sensor Data Integrity

- Every change in spot occupancy or gate passage must be recorded in Logging with timestamp.

---

2. Entity–Relationship Diagram (ERD)



https://www.canva.com/design/DAG6e4lZSYY/1dMp89tpYW45d4TSHCA98g/edit

---

3. Relational Schema

Users

<u>user_id</u>	(PK)
uni_ID	(UNIQUE, NOT NULL)
name	(NOT NULL)
phone	

password

user_type

access_status

created_at

Vehicle

---

vehicle_id (PK)

license_plate (UNIQUE, NOT NULL)

color

user_id (FK → Users.user_id)

model

access_status

approved_by (FK → Users.user_id)

created_at

Service_type

---

service_id (PK)

service_name

description

price

Parking_garage

---

garage_id (PK)

name

garage_access

garage_status (CHECK: 'Open'/'Closed')

location

Gates

---

gate_id (PK)

garage_id (FK → Parking_garage.garage_id)

name

Parking_spots

---

spot_id (PK)

spot_number

spot_status

garage_id (FK → Parking_garage.garage_id)

Occupancy_sensor

---

sensor_id (PK)

spot_id (FK → Parking_spots.spot_id)

is_occupied

Gate_sensors

---

sensor_id (PK)

gate_id (FK → Gates.gate_id)

Sensors

---

sensor_id (PK)

sensor_type

spot_id (FK → Parking_spots.spot_id)

Service_request

---

request_id (PK)

user_id (FK → Users.user_id)

vehicle_id (FK → Vehicle.vehicle_id)

service_id (FK → Service_type.service_id)

status

approved_by (FK → Users.user_id)

created_at

completed_at

spot_id (FK → Parking_spots.spot_id)

Invoice

---

invoice_id (PK)

request_id (FK → Service_request.request_id)

amount

date_issued

time_issued

Payment_method

---

method_id (PK)

method_type

payment_info

Payment

---

payment_id (PK)

invoice_id (FK → Invoice.invoice_id)

amount

date_paid

time_paid

status

method_id (FK → Payment_method.method_id)

Reports

---

report_id (PK)

report_type

user_id (FK → Users.user_id)

date_generated

time_generated

details

Log_record

---

log_id	(PK)
spot_id	(FK → Parking_spots.spot_id)
gate_id	(FK → Gates.gate_id)
request_id	(FK → Service_request.request_id)
report_id	(FK → Reports.report_id)
vehicle_id	(FK → Vehicle.vehicle_id)
user_id	(FK → Users.user_id)
sensor_id	(FK → Sensors.sensor_id)
log_type	
details	
timestamp	

Requests (M:N between Users and Reports)

---

user_id	(PK, FK → Users.user_id)
report_id	(PK, FK → Reports.report_id)

Controls (M:N between Users and Parking_garage)

---

user_id	(PK, FK → Users.user_id)
garage_id	(PK, FK → Parking_garage.garage_id)

Owns (M:N between Vehicle and Service_request)

---

vehicle_id (PK, FK → Vehicle.vehicle_id)

request_id (PK, FK → Service_request.request_id)

4. List of all queries:

Query 1: Total Revenue Generated from Paid Invoices

Type: Aggregate Query

```
SELECT
    SUM(p.amount) AS Total_Revenue
FROM Payment p
WHERE p.status = 'Paid';
```

Explanation:
This query calculates the total revenue generated by the system by summing the payment amounts of all invoices that have a status of **Paid**. It is used to measure overall financial performance.

Query 2: Number of Service Requests per Service Type

Type: Aggregate Query with Join

```
SELECT
    st.service_name,
    COUNT(sr.request_id) AS Total_Requests
FROM Service_type st
JOIN Service_request sr ON st.service_id = sr.service_id
GROUP BY st.service_name;
```

Explanation:
This query counts how many service requests were made for each service type. It joins the `Service_type` and `Service_request` tables and groups the results by service name.

Query 3: Users Who Have Made More Than One Service Request

Type: Subquery with Aggregate

```
SELECT name
FROM Users
WHERE user_id IN (
    SELECT user_id
    FROM Service_request
    GROUP BY user_id
    HAVING COUNT(request_id) > 1
);
```

Explanation:
This query identifies users who have submitted more than one service request. A subquery is used to group service requests by user and filter those with multiple requests.

Query 4: Vehicles That Have Never Been Used in Any Service Request

Type: Subquery

```
SELECT license_plate
FROM Vehicle
WHERE vehicle_id NOT IN (
    SELECT DISTINCT vehicle_id
    FROM Service_request
);
```

Explanation:
This query retrieves vehicles that have never been associated with any service request. It is useful for identifying inactive or unused vehicles in the system.

Query 5 : Invoice, Payment, and Related Service Details

Type: Join (More Than Two Tables)

```
SELECT
    i.invoice_id AS 'Invoice ID',
    i.amount AS 'Amount',
    i.date_issued AS 'Date Issued',
    i.time_issued AS 'Time Issued',
    p.payment_id AS 'Payment ID',
    p.amount AS 'Paid Amount',
    p.date_paid AS 'Date Paid',
    p.time_paid AS 'Time Paid',
    p.status AS 'Payment Status',

    pm.method_type AS 'Payment Method Type',
    pm.payment_info AS 'Payment Credentials/Details',

    sr.request_id AS 'Related Service Request ID',
    u.uni_ID AS 'User University ID',
    u.name AS 'User Name',
    v.license_plate AS 'Vehicle License Plate'
FROM Invoice i
LEFT JOIN Payment p ON i.invoice_id = p.invoice_id
LEFT JOIN Payment_method pm ON p.method_id = pm.method_id
LEFT JOIN Service_request sr ON i.request_id = sr.request_id
LEFT JOIN Users u ON sr.user_id = u.user_id
LEFT JOIN Vehicle v ON sr.vehicle_id = v.vehicle_id
ORDER BY i.date_issued DESC, i.invoice_id DESC;
```

Explanation:

This query retrieves comprehensive billing and payment information for each invoice, including payment status, payment method details, related service requests, user information, and vehicle data.
`LEFT JOIN` is used to ensure that **all invoices are displayed**, even if no payment has been made yet, making the query suitable for financial auditing and tracking unpaid invoices.

Query 6: Total Payments per Payment Method

Type: Aggregate Query with Join

```
SELECT
    pm.method_type,
    SUM(p.amount) AS Total_Amount
FROM Payment p
JOIN Payment_method pm ON p.method_id = pm.method_id
GROUP BY pm.method_type;
```

Explanation:
This query calculates the total amount of money collected through each payment method, helping analyze payment trends and preferred methods.

New Query 7 : Number of Service Requests per Garage

Type: Aggregate Query with Multiple Joins

```
SELECT
    pg.name AS 'Garage Name',
    COUNT(sr.request_id) AS 'Total Service Requests'
FROM Parking_garage pg
JOIN Parking_spots ps ON pg.garage_id = ps.garage_id
JOIN Service_request sr ON ps.spot_id = sr.spot_id
GROUP BY pg.name
ORDER BY COUNT(sr.request_id) DESC;
```

Explanation:

This query calculates the total number of service requests handled in each parking garage.
It joins parking garages, parking spots, and service requests to analyze garage-level activity and workload distribution.

Query 8: Service Requests with Assigned Parking Spots and Garages

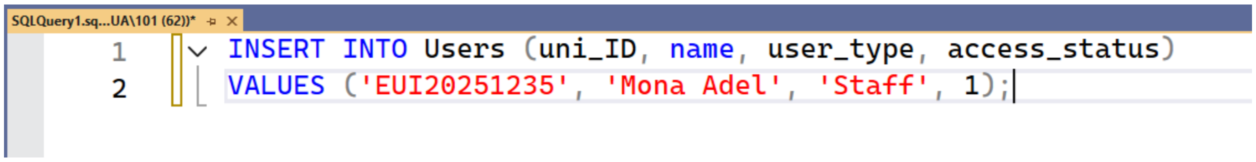
Type: Join (More Than Two Tables)

```
SELECT
    sr.request_id,
    u.name AS User_Name,
    v.license_plate,
    ps.spot_number,
    pg.name AS Garage_Name,
    sr.status,
    sr.created_at
FROM Service_request sr
JOIN Users u ON sr.user_id = u.user_id
JOIN Vehicle v ON sr.vehicle_id = v.vehicle_id
JOIN Parking_spots ps ON sr.spot_id = ps.spot_id
JOIN Parking_garage pg ON ps.garage_id = pg.garage_id;
```

Explanation:

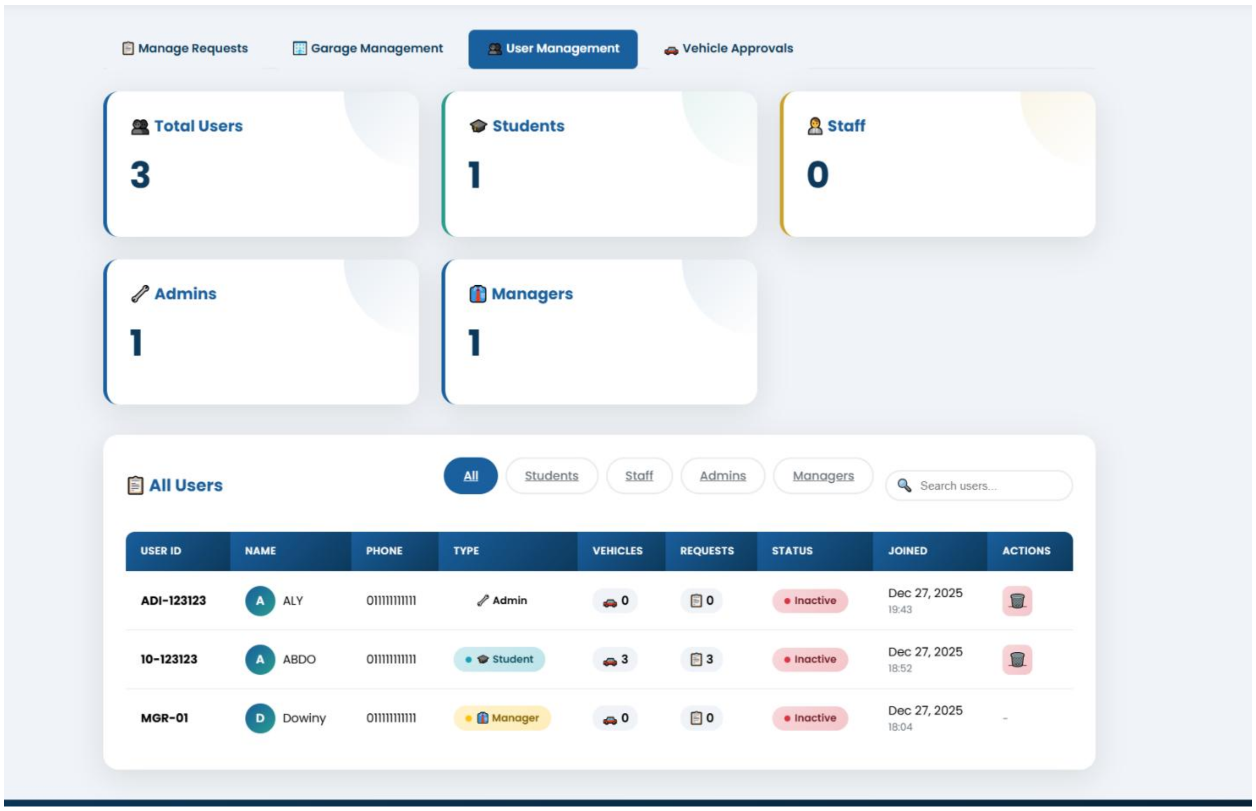
This query retrieves service requests along with their associated users, vehicles, assigned parking spots, and the garages in which the spots are located.
It joins **five related tables** to present a complete view of parking utilization without depending on the logging mechanism.

5. The GUI screenshots:



```sql
INSERT INTO Users (uni_ID, name, user_type, access_status)
VALUES ('EUI20251235', 'Mona Adel', 'Staff', 1);
```

Insert Statement



Total Users 3

Students 1

Staff 0

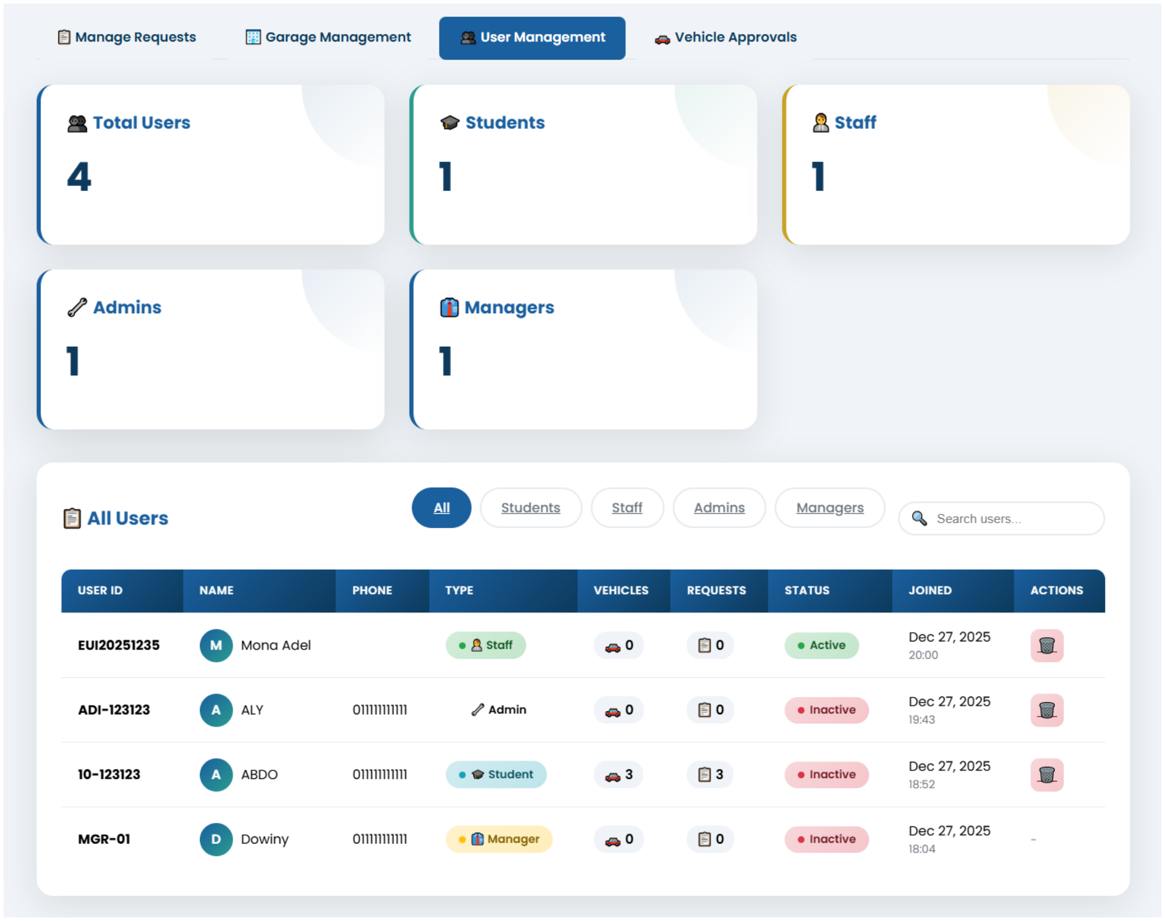
Admins 1

Managers 1

All Users

USER ID	NAME	PHONE	TYPE	VEHICLES	REQUESTS	STATUS	JOINED	ACTIONS
ADI-123123	A ALY	01111111111	Admin	0	0	● Inactive	Dec 27, 2025 19:43	🗑
10-123123	A ABDO	01111111111	● Student	3	3	● Inactive	Dec 27, 2025 18:52	🗑
MGR-01	D Dowiny	01111111111	● Manager	0	0	● Inactive	Dec 27, 2025 18:04	-

Users board before insert statement



Total Users 4

Students 1

Staff 1

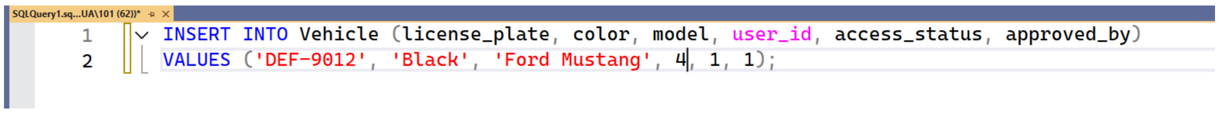
Admins 1

Managers 1

All Users

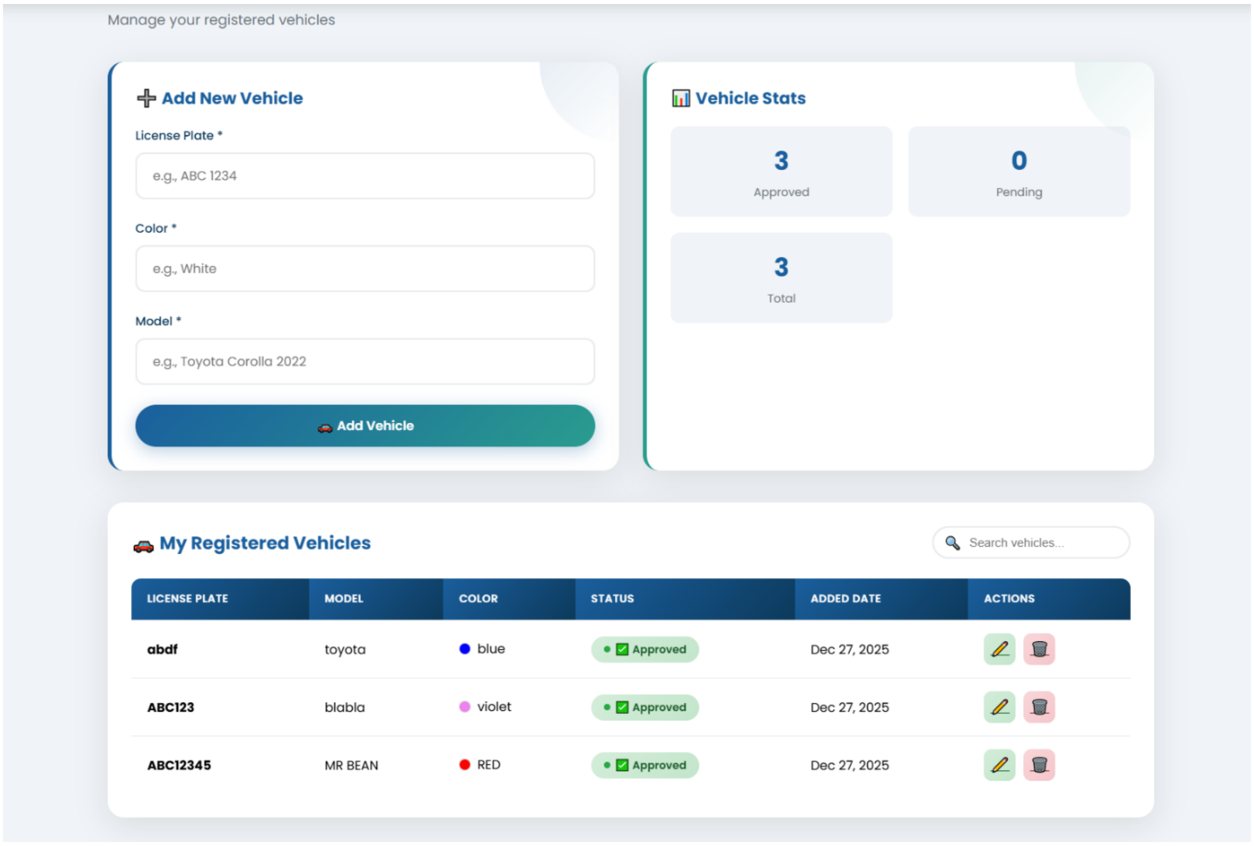
USER ID	NAME	PHONE	TYPE	VEHICLES	REQUESTS	STATUS	JOINED	ACTIONS
EUI20251235	M Mona Adel		● Staff	0	0	● Active	Dec 27, 2025 20:00	
ADI-123123	A ALY	01111111111	Admin	0	0	● Inactive	Dec 27, 2025 19:43	
10-123123	A ABDO	01111111111	● Student	3	3	● Inactive	Dec 27, 2025 18:52	
MGR-01	D Dowiny	01111111111	● Manager	0	0	● Inactive	Dec 27, 2025 18:04	-

User board after insert statement



```sql
INSERT INTO Vehicle (license_plate, color, model, user_id, access_status, approved_by)
VALUES ('DEF-9012', 'Black', 'Ford Mustang', 4, 1, 1);
```

Insert statement


Manage your registered vehicles

Add New Vehicle

License Plate *

e.g., ABC 1234

Color *

e.g., White

Model *

e.g., Toyota Corolla 2022

Vehicle Stats

3 Approved

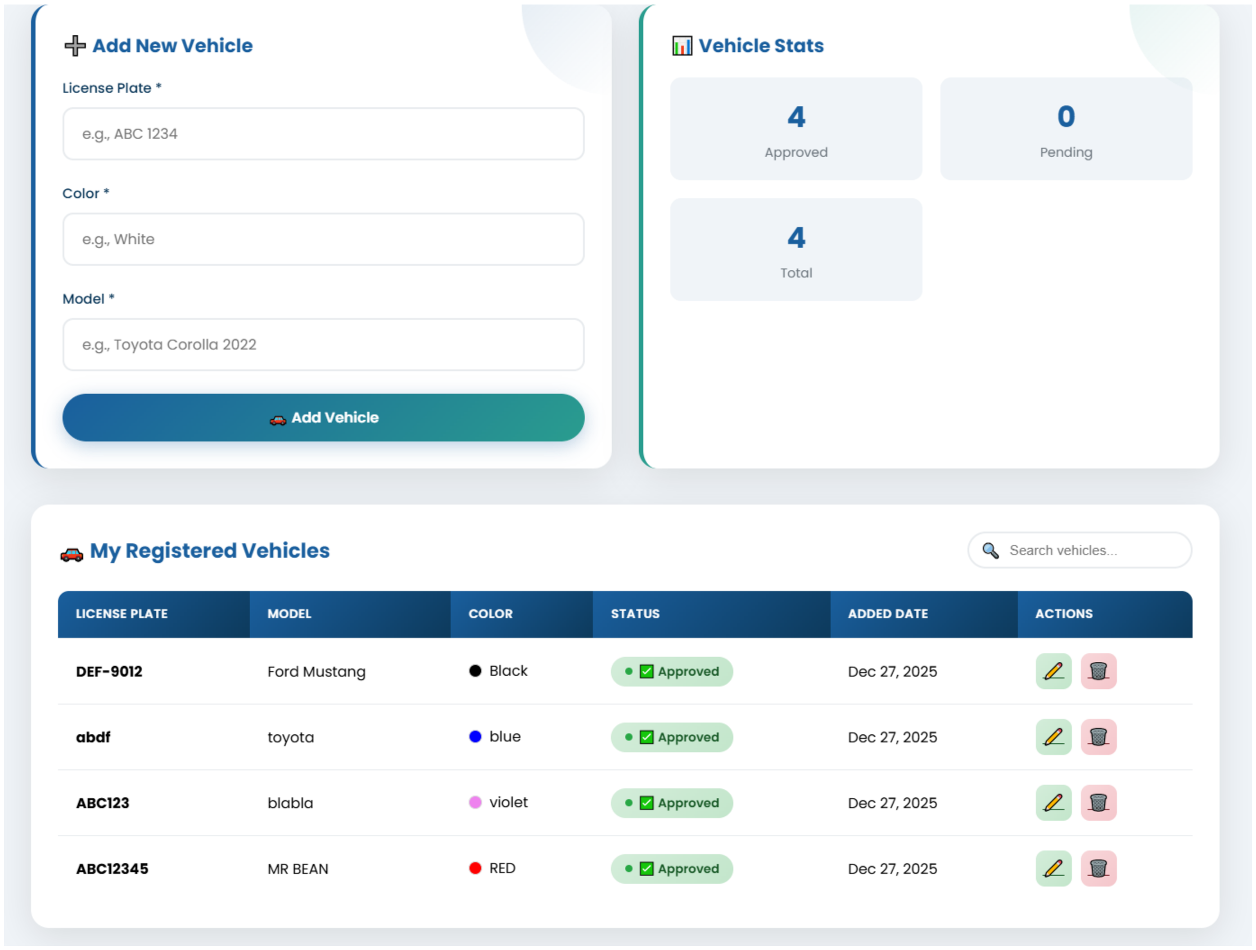
0 Pending

3 Total

My Registered Vehicles

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
abdf	toyota	● blue	● ✓ Approved	Dec 27, 2025	
ABC123	blabla	● violet	● ✓ Approved	Dec 27, 2025	
ABC12345	MR BEAN	● RED	● ✓ Approved	Dec 27, 2025	

Vehicle board before insert statement


➕ Add New Vehicle

License Plate *

e.g., ABC 1234

Color *

e.g., White

Model *

e.g., Toyota Corolla 2022

Add Vehicle

Vehicle Stats

4 Approved

0 Pending

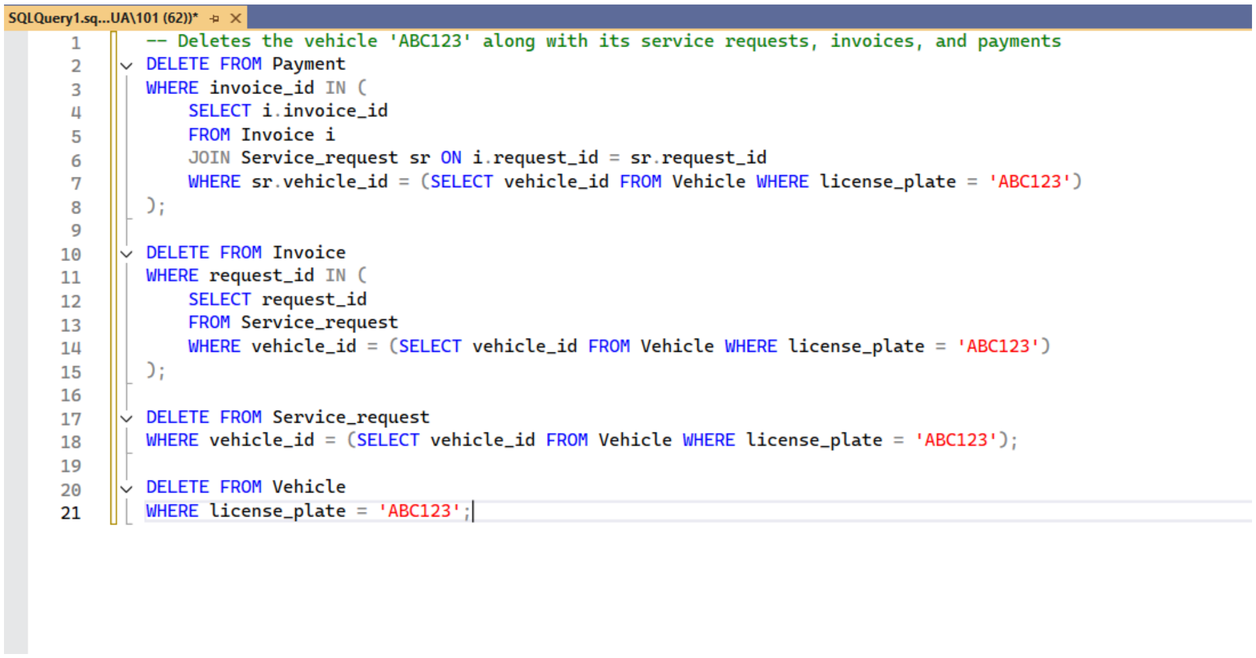
4 Total

My Registered Vehicles

Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	✓ Approved	Dec 27, 2025	
abdf	toyota	● blue	✓ Approved	Dec 27, 2025	
ABC123	blabla	● violet	✓ Approved	Dec 27, 2025	
ABC12345	MR BEAN	● RED	✓ Approved	Dec 27, 2025	

Vehicle board after insert statement


SQLQuery1.sq...UA\101 (62))*

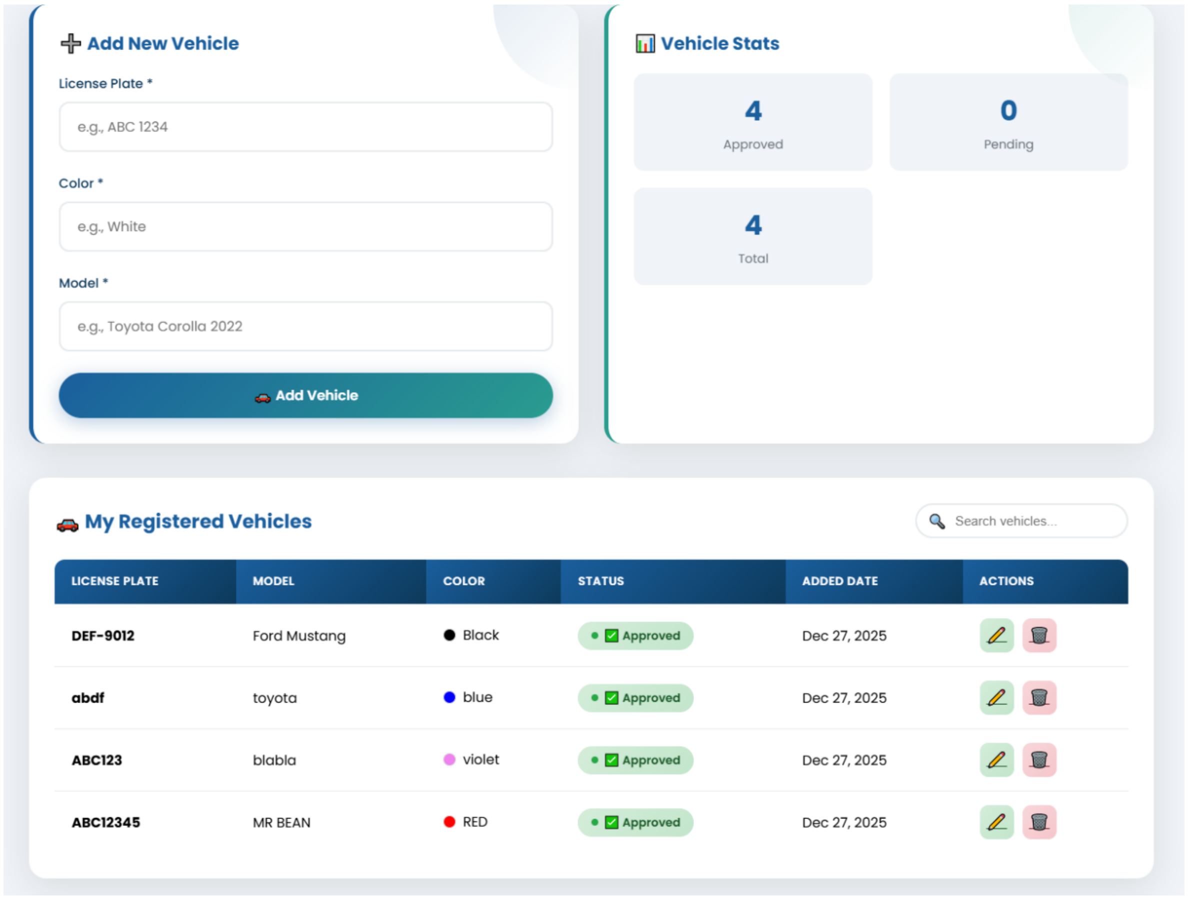
```sql
-- Deletes the vehicle 'ABC123' along with its service requests, invoices, and payments
DELETE FROM Payment
WHERE invoice_id IN (
    SELECT i.invoice_id
    FROM Invoice i
    JOIN Service_request sr ON i.request_id = sr.request_id
    WHERE sr.vehicle_id = (SELECT vehicle_id FROM Vehicle WHERE license_plate = 'ABC123')
);

DELETE FROM Invoice
WHERE request_id IN (
    SELECT request_id
    FROM Service_request
    WHERE vehicle_id = (SELECT vehicle_id FROM Vehicle WHERE license_plate = 'ABC123')
);

DELETE FROM Service_request
WHERE vehicle_id = (SELECT vehicle_id FROM Vehicle WHERE license_plate = 'ABC123');

DELETE FROM Vehicle
WHERE license_plate = 'ABC123';
```

Delete statements



➕ Add New Vehicle

License Plate *

e.g., ABC 1234

Color *

e.g., White

Model *

e.g., Toyota Corolla 2022

🚗 Add Vehicle

📊 Vehicle Stats

4 Approved

0 Pending

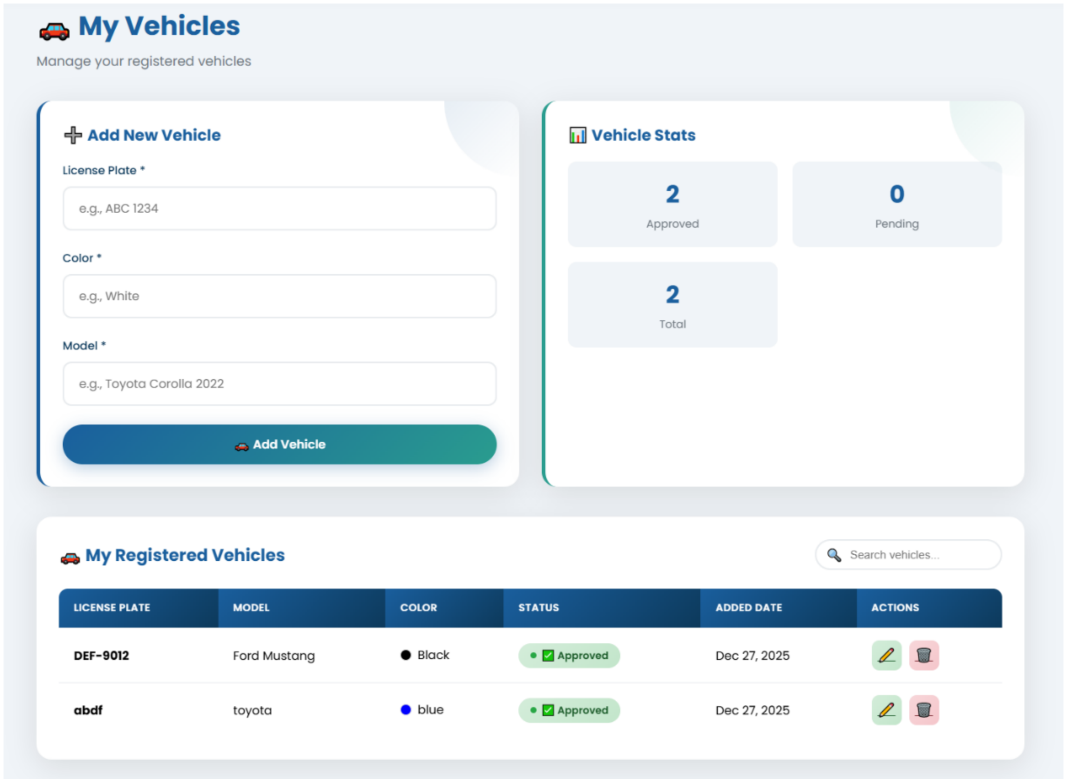
4 Total

🚗 My Registered Vehicles

🔍 Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	● ✅ Approved	Dec 27, 2025	✏️ 🗑️
abdf	toyota	● blue	● ✅ Approved	Dec 27, 2025	✏️ 🗑️
ABC123	blabla	● violet	● ✅ Approved	Dec 27, 2025	✏️ 🗑️
ABC12345	MR BEAN	● RED	● ✅ Approved	Dec 27, 2025	✏️ 🗑️

Vehicle board before delete statement



🚗 My Vehicles

Manage your registered vehicles

➕ Add New Vehicle

License Plate *

e.g., ABC 1234

Color *

e.g., White

Model *

e.g., Toyota Corolla 2022

🚗 Add Vehicle

📊 Vehicle Stats

2 Approved

0 Pending

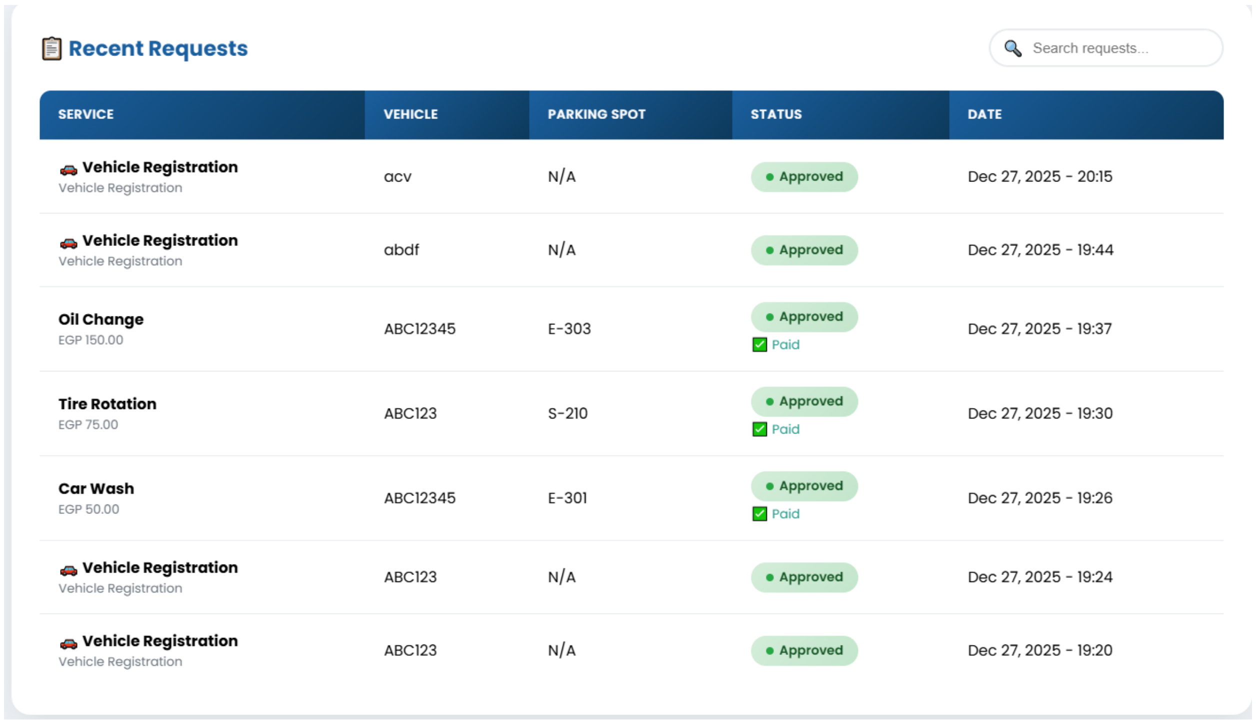
2 Total

🚗 My Registered Vehicles

🔍 Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	● ✅ Approved	Dec 27, 2025	✏️ 🗑️
abdf	toyota	● blue	● ✅ Approved	Dec 27, 2025	✏️ 🗑️

Vehicle board after delete statement

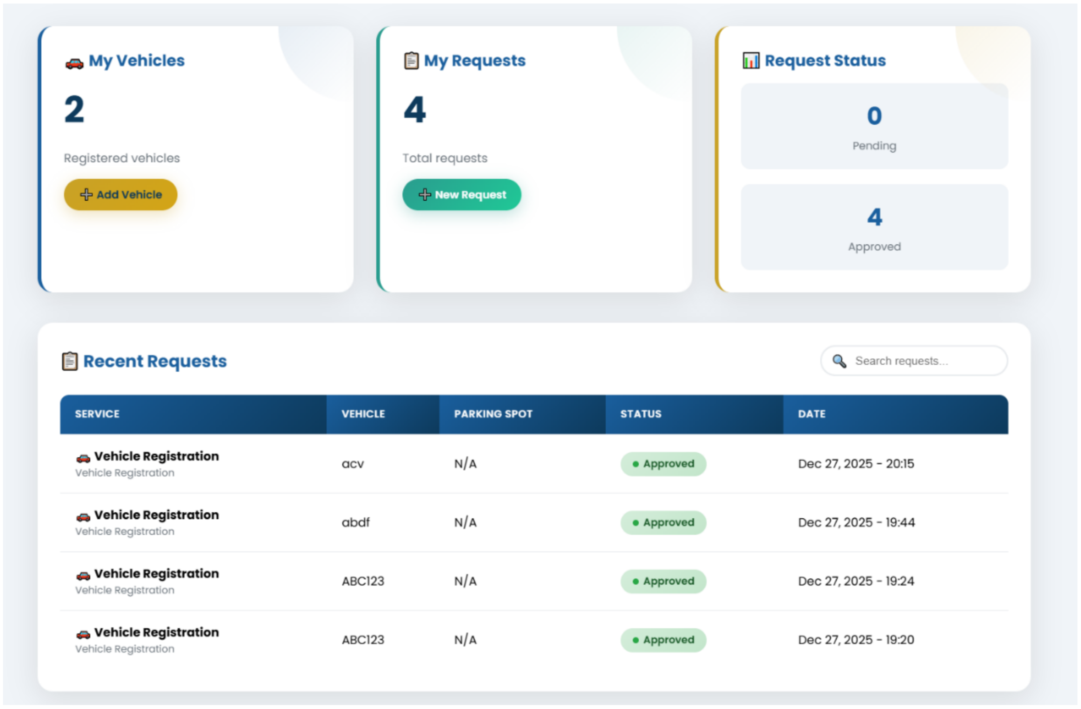


📋 **Recent Requests**

Search requests...

SERVICE	VEHICLE	PARKING SPOT	STATUS	DATE
🚗 **Vehicle Registration** Vehicle Registration	acv	N/A	● Approved	Dec 27, 2025 - 20:15
🚗 **Vehicle Registration** Vehicle Registration	abdf	N/A	● Approved	Dec 27, 2025 - 19:44
Oil Change EGP 150.00	ABC12345	E-303	● Approved ✅ Paid	Dec 27, 2025 - 19:37
Tire Rotation EGP 75.00	ABC123	S-210	● Approved ✅ Paid	Dec 27, 2025 - 19:30
Car Wash EGP 50.00	ABC12345	E-301	● Approved ✅ Paid	Dec 27, 2025 - 19:26
🚗 **Vehicle Registration** Vehicle Registration	ABC123	N/A	● Approved	Dec 27, 2025 - 19:24
🚗 **Vehicle Registration** Vehicle Registration	ABC123	N/A	● Approved	Dec 27, 2025 - 19:20

Requests board before delete statement



🚗 **My Vehicles**

2

Registered vehicles

Add Vehicle

📋 **My Requests**

4

Total requests

New Request

📊 **Request Status**

0 Pending

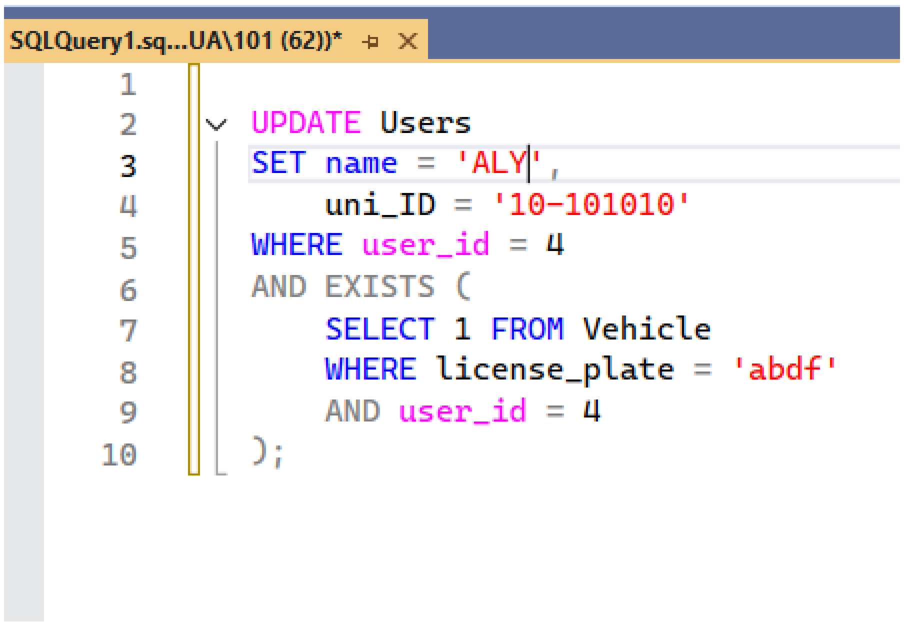
4 Approved

📋 **Recent Requests**

Search requests...

SERVICE	VEHICLE	PARKING SPOT	STATUS	DATE
🚗 **Vehicle Registration** Vehicle Registration	acv	N/A	● Approved	Dec 27, 2025 - 20:15
🚗 **Vehicle Registration** Vehicle Registration	abdf	N/A	● Approved	Dec 27, 2025 - 19:44
🚗 **Vehicle Registration** Vehicle Registration	ABC123	N/A	● Approved	Dec 27, 2025 - 19:24
🚗 **Vehicle Registration** Vehicle Registration	ABC123	N/A	● Approved	Dec 27, 2025 - 19:20

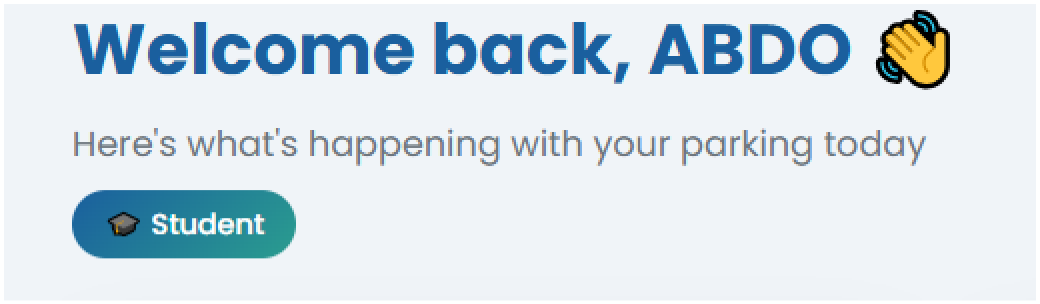
Vehicle board after delete statement



```sql

UPDATE Users
SET name = 'ALY',
    uni_ID = '10-101010'
WHERE user_id = 4
AND EXISTS (
    SELECT 1 FROM Vehicle
    WHERE license_plate = 'abdf'
    AND user_id = 4
);
```

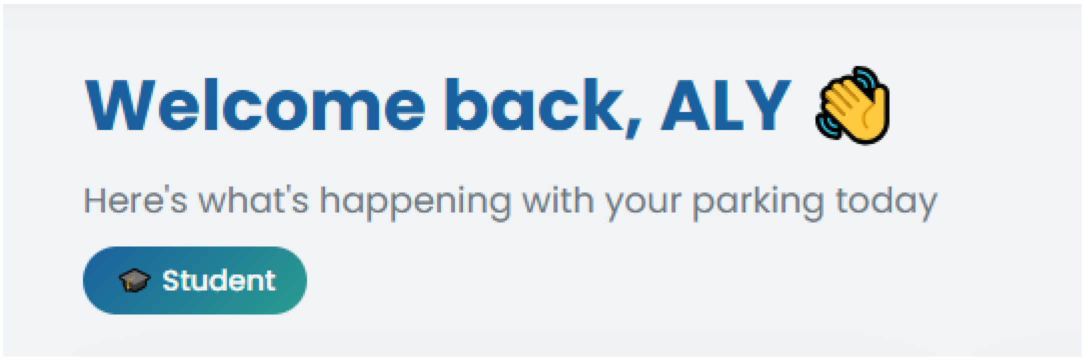
Name Update Statement



Welcome back, ABDO 👋

Here's what's happening with your parking today

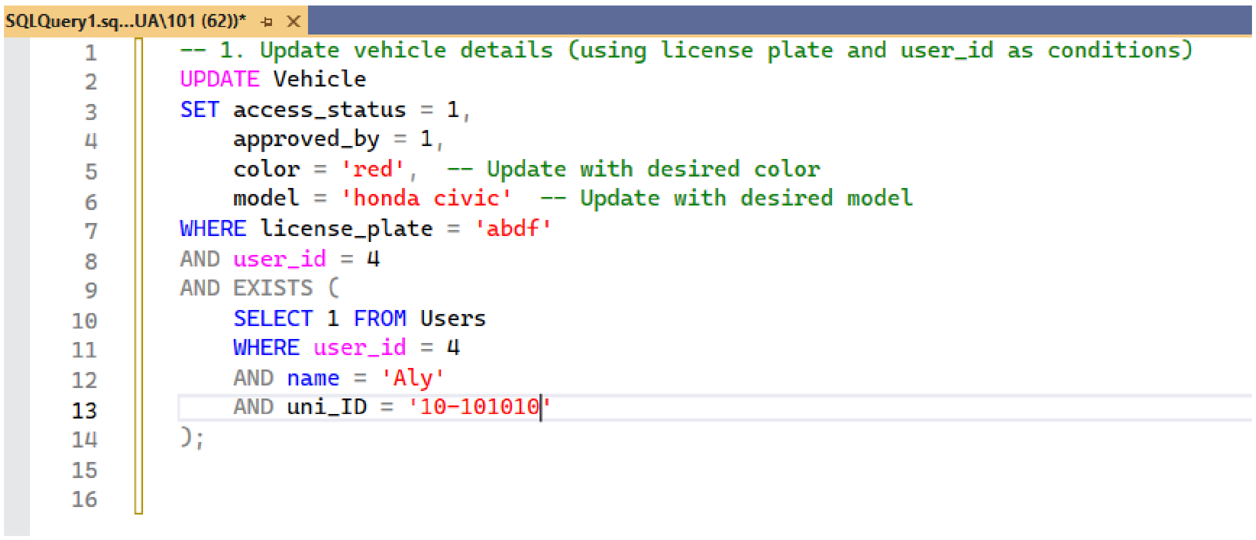
Name before Update statement



Welcome back, ALY 👋

Here's what's happening with your parking today

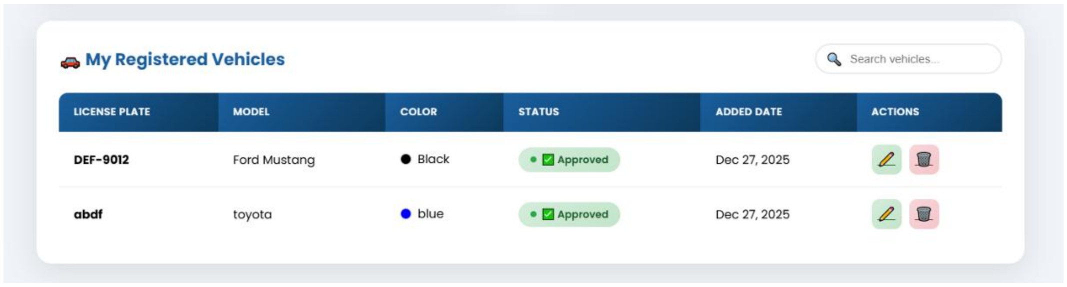
Name after Update statement



```sql
-- 1. Update vehicle details (using license plate and user_id as conditions)
UPDATE Vehicle
SET access_status = 1,
    approved_by = 1,
    color = 'red',   -- Update with desired color
    model = 'honda civic'   -- Update with desired model
WHERE license_plate = 'abdf'
AND user_id = 4
AND EXISTS (
    SELECT 1 FROM Users
    WHERE user_id = 4
    AND name = 'Aly'
    AND uni_ID = '10-101010'
);


```

Vehicle update statement

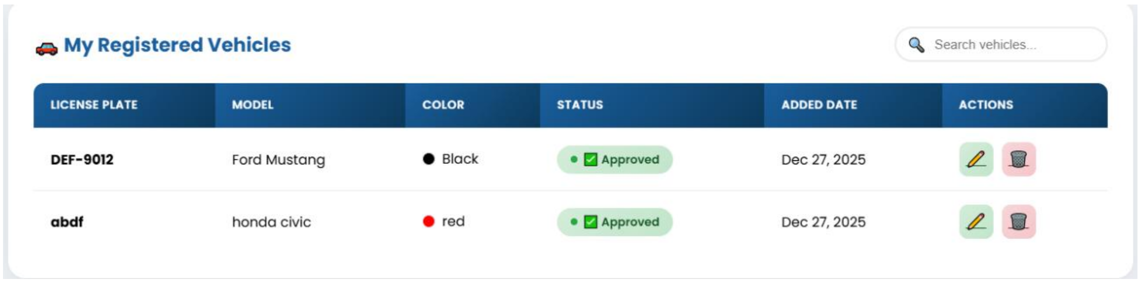


My Registered Vehicles

Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	● ✓ Approved	Dec 27, 2025	
abdf	toyota	● blue	● ✓ Approved	Dec 27, 2025	

Vehicle board before update statement

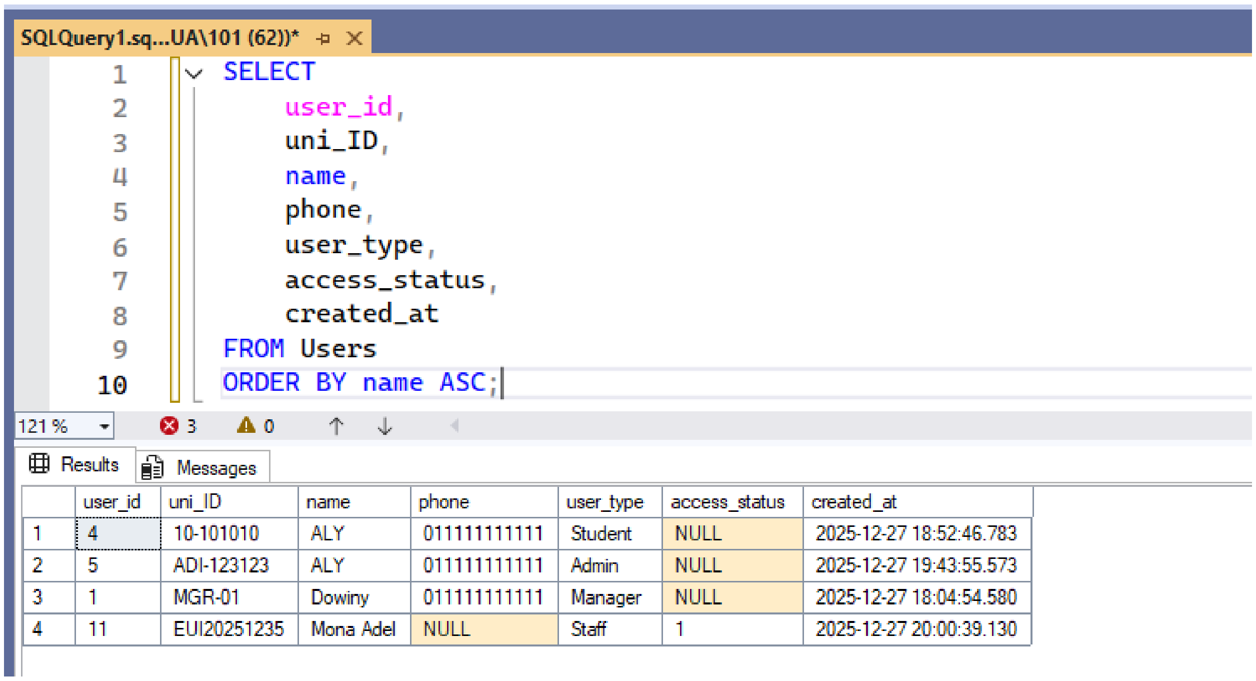


My Registered Vehicles

Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	● ✓ Approved	Dec 27, 2025	
abdf	honda civic	● red	● ✓ Approved	Dec 27, 2025	

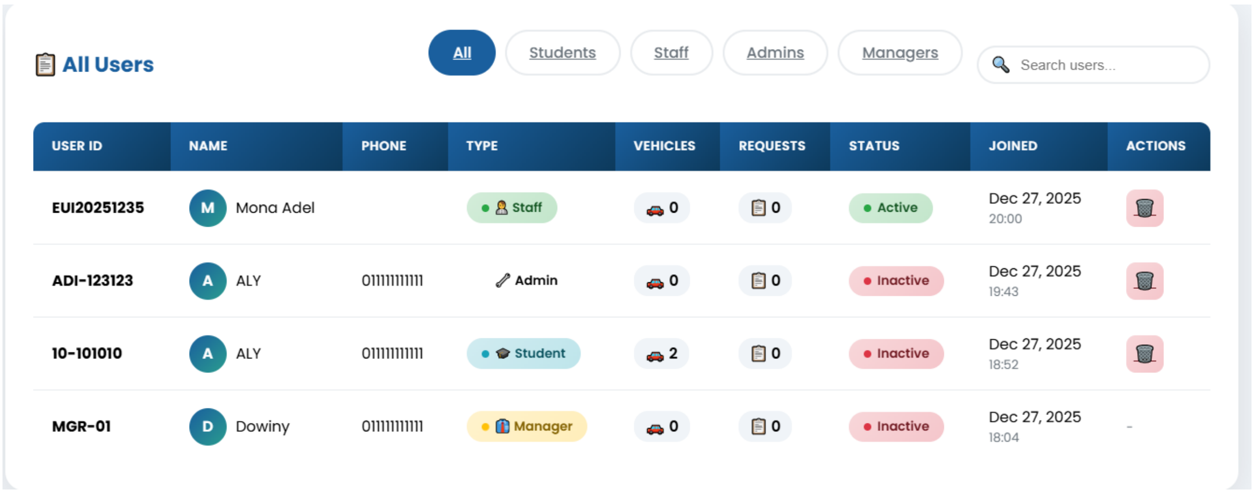
Vehicle board after update statement



```sql
SELECT
    user_id,
    uni_ID,
    name,
    phone,
    user_type,
    access_status,
    created_at
FROM Users
ORDER BY name ASC;
```

	user_id	uni_ID	name	phone	user_type	access_status	created_at
1	4	10-101010	ALY	011111111111	Student	NULL	2025-12-27 18:52:46.783
2	5	ADI-123123	ALY	011111111111	Admin	NULL	2025-12-27 19:43:55.573
3	1	MGR-01	Dowiny	011111111111	Manager	NULL	2025-12-27 18:04:54.580
4	11	EUI20251235	Mona Adel	NULL	Staff	1	2025-12-27 20:00:39.130

User Select Statement

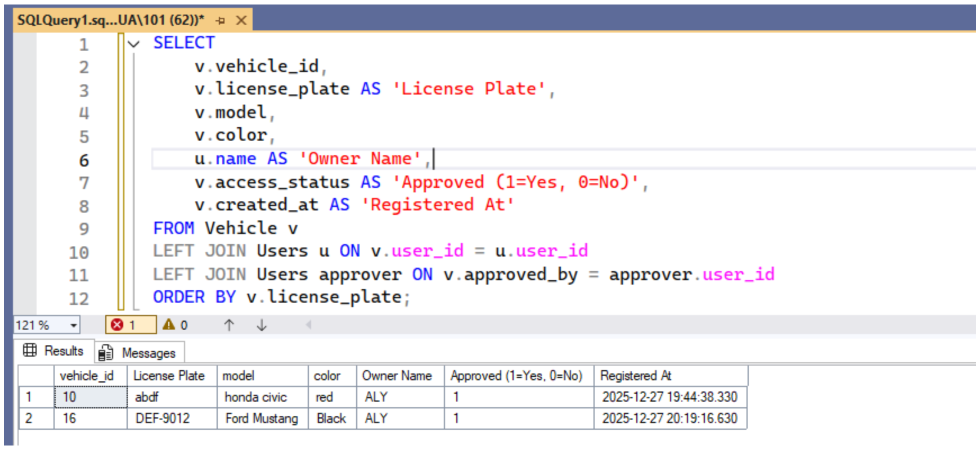


All Users

All | Students | Staff | Admins | Managers | Search users...

USER ID	NAME	PHONE	TYPE	VEHICLES	REQUESTS	STATUS	JOINED	ACTIONS
EUI20251235	M Mona Adel		Staff	0	0	Active	Dec 27, 2025 20:00	
ADI-123123	A ALY	01111111111	Admin	0	0	Inactive	Dec 27, 2025 19:43	
10-101010	A ALY	01111111111	Student	2	0	Inactive	Dec 27, 2025 18:52	
MGR-01	D Dowiny	01111111111	Manager	0	0	Inactive	Dec 27, 2025 18:04	-

GUI User Board



SQLQuery1.sq...UA\101 (62))*

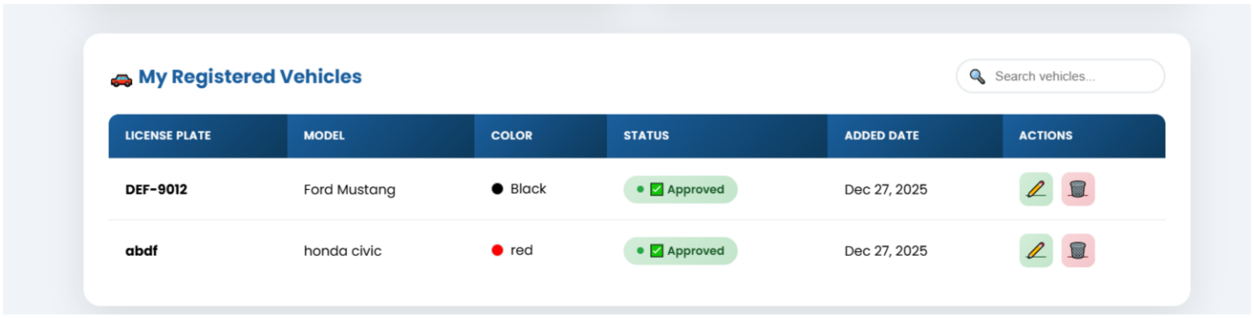
```sql
SELECT
    v.vehicle_id,
    v.license_plate AS 'License Plate',
    v.model,
    v.color,
    u.name AS 'Owner Name',
    v.access_status AS 'Approved (1=Yes, 0=No)',
    v.created_at AS 'Registered At'
FROM Vehicle v
LEFT JOIN Users u ON v.user_id = u.user_id
LEFT JOIN Users approver ON v.approved_by = approver.user_id
ORDER BY v.license_plate;
```

121 % | 1 | 0

Results | Messages

	vehicle_id	License Plate	model	color	Owner Name	Approved (1=Yes, 0=No)	Registered At
1	10	abdf	honda civic	red	ALY	1	2025-12-27 19:44:38.330
2	16	DEF-9012	Ford Mustang	Black	ALY	1	2025-12-27 20:19:16.630

Select Statement involving more than one table of the database- (using joins).



My Registered Vehicles

Search vehicles...

LICENSE PLATE	MODEL	COLOR	STATUS	ADDED DATE	ACTIONS
DEF-9012	Ford Mustang	● Black	Approved	Dec 27, 2025	
abdf	honda civic	● red	Approved	Dec 27, 2025	

GUI of the above select statement.