

**Database Foundations**  
**ICCS225**

**Project Report: Garage Management System**

Chawinthon Kittivacharaphong

December 3, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Pain Points</b>	<b>4</b>
<b>3</b>	<b>Solution Overview</b>	<b>4</b>
<b>4</b>	<b>Entity-Relationship Diagram</b>	<b>5</b>
<b>5</b>	<b>Applications Flow</b>	<b>6</b>
5.1	Login & Sign up . . . . .	6
5.1.1	Customer authenticate . . . . .	7
5.1.2	Employee authenticate . . . . .	8
5.1.3	Owner authenticate . . . . .	9
5.2	Admin side . . . . .	10
5.2.1	Owner management . . . . .	11
5.2.2	Subscription plan management . . . . .	12
5.2.3	Subscription transaction management . . . . .	13
5.3	Owner side . . . . .	15
5.3.1	Approved requests and create employee . . . . .	16
5.3.2	yearly financial summary . . . . .	18
5.3.3	Appointment management . . . . .	20
5.3.4	Customer management . . . . .	22
5.3.5	Membership plan management . . . . .	24
5.3.6	Employment request management . . . . .	25
5.3.7	Employee schedule management . . . . .	26
5.3.8	Employee management . . . . .	27
5.3.9	Financial summary . . . . .	29
5.3.10	Membership transaction . . . . .	31
5.3.11	Service Record . . . . .	33
5.3.12	Subscription plan . . . . .	35
5.4	Employee side . . . . .	36
5.4.1	Change appointment status . . . . .	37
5.5	Customer side . . . . .	38
5.5.1	Companys menu . . . . .	39
5.5.2	Service menu . . . . .	41
5.5.3	Vehicle management . . . . .	42
5.5.4	Remove vehicle . . . . .	43
<b>6</b>	<b>Table details</b>	<b>46</b>
6.1	Company table . . . . .	46
6.1.1	Purpose . . . . .	46
6.1.2	Important Columns . . . . .	46
6.1.3	Keys & Constraints . . . . .	46
6.2	Owner table . . . . .	47
6.2.1	Purpose . . . . .	47
6.2.2	Important Columns . . . . .	47
6.2.3	Keys & Constraints . . . . .	47
6.3	Customer table . . . . .	48
6.3.1	Purpose . . . . .	48
6.3.2	Important Columns . . . . .	48

6.3.3	Keys & Constraints . . . . .	48
6.4	Employee table . . . . .	49
6.4.1	Purpose . . . . .	49
6.4.2	Important Columns . . . . .	49
6.4.3	Keys & Constraints . . . . .	49
6.5	Employee Schedule table . . . . .	50
6.5.1	Purpose . . . . .	50
6.5.2	Important Columns . . . . .	50
6.5.3	Keys & Constraints . . . . .	50
6.6	Equipment table . . . . .	51
6.6.1	Purpose . . . . .	51
6.6.2	Important Columns . . . . .	51
6.6.3	Keys & Constraints . . . . .	51
6.7	Vehicle table . . . . .	52
6.7.1	Purpose . . . . .	52
6.7.2	Important Columns . . . . .	52
6.7.3	Keys & Constraints . . . . .	52
6.8	Service table . . . . .	53
6.8.1	Purpose . . . . .	53
6.8.2	Important Columns . . . . .	53
6.8.3	Keys & Constraints . . . . .	53
6.9	appointment . . . . .	54
6.9.1	Purpose . . . . .	54
6.9.2	Important Columns . . . . .	54
6.9.3	Keys & Constraints . . . . .	54
6.10	Service Record table . . . . .	55
6.10.1	Purpose . . . . .	55
6.10.2	Important Columns . . . . .	55
6.10.3	Keys & Constraints . . . . .	55
6.11	Spares table . . . . .	56
6.11.1	Purpose . . . . .	56
6.11.2	Important Columns . . . . .	56
6.11.3	Keys & Constraints . . . . .	56
6.12	Membership Plan table . . . . .	57
6.12.1	Purpose . . . . .	57
6.12.2	Important Columns . . . . .	57
6.12.3	Keys & Constraints . . . . .	57
6.13	Membership table . . . . .	58
6.13.1	Purpose . . . . .	58
6.13.2	Important Columns . . . . .	58
6.13.3	Keys & Constraints . . . . .	58
6.14	Membership Transaction table . . . . .	59
6.14.1	Purpose . . . . .	59
6.14.2	Important Columns . . . . .	59
6.14.3	Keys & Constraints . . . . .	59
6.15	Subscription Plan table . . . . .	60
6.15.1	Purpose . . . . .	60
6.15.2	Important Columns . . . . .	60
6.15.3	Keys & Constraints . . . . .	60
6.16	Subscription Transaction table . . . . .	61
6.16.1	Purpose . . . . .	61

6.16.2	Important Columns . . . . .	61
6.16.3	Keys & Constraints . . . . .	61
6.17	Transaction table . . . . .	62
6.17.1	Purpose . . . . .	62
6.17.2	Important Columns . . . . .	62
6.17.3	Keys & Constraints . . . . .	62
6.18	Request Form table . . . . .	63
6.18.1	Purpose . . . . .	63
6.18.2	Important Columns . . . . .	63
6.18.3	Keys & Constraints . . . . .	63
<b>7</b>	<b>Database Design Quality, Security, and efficiency</b>	<b>64</b>
7.1	Normalization Design . . . . .	64
7.2	Security Mindset & Implementation . . . . .	64
7.3	Dummy Database . . . . .	64
7.4	Query Efficiency Implementation . . . . .	65
<b>8</b>	<b>Financial Considerations</b>	<b>66</b>
<b>9</b>	<b>Database access</b>	<b>66</b>
<b>10</b>	<b>Github Repository</b>	<b>66</b>

## 1 Introduction

This report outlines the development of a comprehensive Garage Management System designed to streamline operations for independent automotive service centers. The platform centralizes critical administrative functions, including mechanic scheduling, spare parts inventory tracking, equipment tracking, financial reporting, and customer vehicle history. The primary objective is to replace fragmented, manual processes—such as paper-based logs—with a unified digital solution to enhance operational efficiency and profitability.

## 2 Pain Points

Small garages face several problems that this project solves:

- **Budget Constraints:** Small garages cannot afford to hire professional programmers to build custom software.
- **Data Inconsistency:** Writing everything by hand is slow and tiring.
- **Prevent Data Loss:** Paper records get lost, damaged, or are hard to read. This system keeps data safe.

## 3 Solution Overview

The platform helps manage the entire garage through these features:

- **Work & Service:**
  - **Job Requests:** Create forms for new repair jobs.
  - **Service Management:** Track the status of vehicle repairs.
  - **Appointments:** Schedule dates and times for customers.
- **People Management:**
  - **Customers:** Keep a list of clients and their car history.
  - **Employees:** Manage staff details and shifts.
  - **Memberships:** Track loyal customers and their subscription status.
- **Shop & Finance:**
  - **Stock Tracker:** Count spare parts and equipment so you don't run out.
  - **Financial Reports:** See how much money is made from services and memberships.

## 4 Entity-Relationship Diagram

The Entity-Relationship Diagram (ERD) helps visualize the database structure for the Garage Management System. It shows how different entities like Customers, Employees, Vehicles, Services, Spare Parts, and Equipment relate to each other.

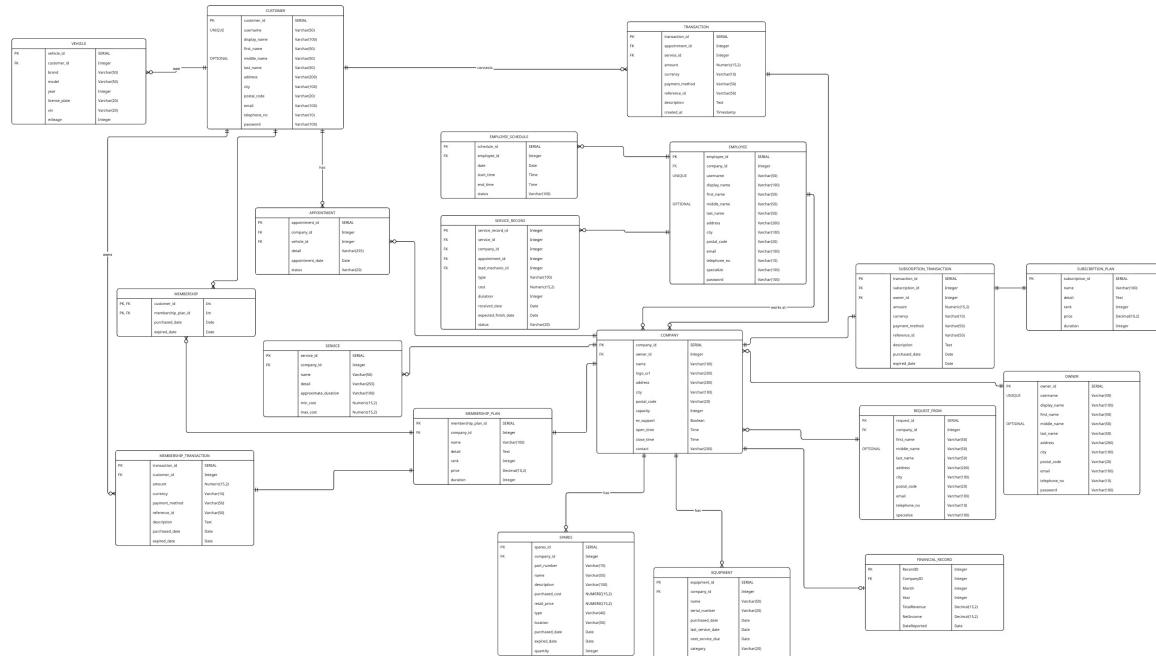
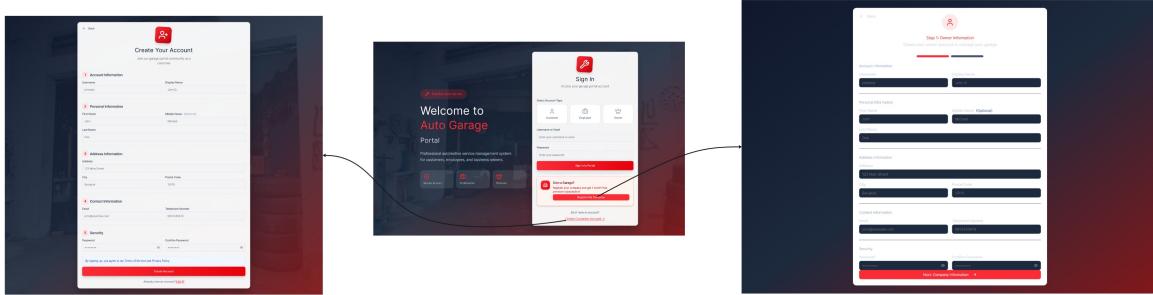


Figure 1: Entity-Relationship Diagram for Garage Management System

## 5 Applications Flow

### 5.1 Login & Sign up



Users can sign up as an owner, employee, or customer. After signing up, they can log in to access their respective dashboards. Every user type has a dedicated authentication function in the database to verify their credentials during login.

### 5.1.1 Customer authenticate

```
1 create function authenticate_customer(p_identifier character varying,
2                                     p_password character varying) returns integer
3 language plpgsql
4 as
5 $$
6 DECLARE
7     v_user_id INT;
8 BEGIN
9     SELECT id INTO v_user_id
10    FROM "customer"
11   WHERE (username = p_identifier OR email = p_identifier)
12     -- Check if the input password matches the stored hash
13     AND password = crypt(p_password, password);
14
15    RETURN v_user_id; -- return null if user doesn't exist or input not
16    match
17
18 END;
19 $$;
20
21 alter function authenticate_customer(varchar, varchar) owner to
22 sc_db_project_user;
```

#### Explanation (authenticate\_customer)

**Purpose:** Validates a customer attempting to log in using either their username or email plus a plaintext password. Returns the internal customer id if credentials match, otherwise NULL (no row selected).

**Logic:** Single SELECT on the `customer` table with an OR predicate for identifier matching. Uses PostgreSQL's `crypt` to hash the supplied password and compare with the stored salted hash.

### 5.1.2 Employee authenticate

```
1 create function authenticate_employee(p_identifier character varying,
2                                     p_password character varying) returns integer
3   language plpgsql
4 as
5 $$
6 DECLARE
7   v_user_id INT;
8 BEGIN
9   SELECT id INTO v_user_id
10  FROM "employee"
11  WHERE (username = p_identifier OR email = p_identifier)
12    -- Check if the input password matches the stored hash
13    AND password = crypt(p_password, password);
14
15  RETURN v_user_id;
16 END;
17 $$;
18 alter function authenticate_employee(varchar, varchar) owner to
  sc_db_project_user;
```

#### Explanation (authenticate\_employee)

**Purpose:** Authenticates an employee for access to internal (non-owner) operational features.

**Logic:** Mirrors the customer approach; performs a lookup with OR on username/email and validates password using `crypt`.

### 5.1.3 Owner authenticate

```
1 create function authenticate_owner(p_identifier character varying,
2                                     p_password character varying) returns integer
3                                     language plpgsql
4 as
5 $$
6 DECLARE
7     v_user_id INT;
8 BEGIN
9     SELECT id
10    INTO v_user_id
11   FROM "owner"
12  WHERE (username = p_identifier OR email = p_identifier)
13    -- Check if the input password matches the stored hash
14    AND password = crypt(p_password, password);
15
16    RETURN v_user_id;
17 END;
18 $$;
19 alter function authenticate_owner(varchar, varchar) owner to
sc_db_project_user;
```

#### Explanation (authenticate\_owner)

**Purpose:** Grants owners access to administrative dashboards (financials, employee management) after verifying credentials.

**Logic:** SELECT + password hash comparison via `crypt`.

## 5.2 Admin side

The image displays three separate screenshots of a web-based administrative dashboard, likely for a service provider like GarageHub.

**Screenshot 1: Subscription Plan Management**

This screen shows a list of available subscription plans:

Plan ID	Plan Name	Description	Price (USD)	Duration	Status
P001	Standard Monthly	Standard monthly plan for most garages. Includes basic management features.	\$10.00	1 Month	Active
P002	Professional Monthly	Advanced features for professionals. Includes inventory tracking.	\$14.99	1 Month	Active
P003	Enterprise Annual	Complete solution for large operations. All features included plus reporting.	\$100.00	1 Year	Active
P004	Prosumer Monthly	Prosumer plan with enhanced features. Perfect for medium-sized garages.	\$20.00	1 Month	Active

**Screenshot 2: Subscription Transactions**

This screen shows a list of completed transactions:

Trans ID	Subscription Plan	Owner Name	Amount	Payment Method	Reference ID	Description	Purchased	Expires
T001	Standard Monthly	Michael Johnson	\$10.00	Credit Card	NFT-1234-5678	Monthly subscription renewal for AutoGarage	01-Nov-2023	01-Dec-2023
T002	Professional Monthly	Sarah Williams	\$14.99	Credit Card	NFT-1234-5678	Annual subscription for Premium Auto Services	01-Nov-2023	01-Dec-2023
T003	Enterprise Annual	David Chen	\$100.00	Credit Card	NFT-1234-5678	Annual enterprise plan for Meggarage Network	01-Nov-2023	01-Nov-2024
T004	Prosumer Monthly	Emma Thompson	\$20.00	Credit Card	NFT-1234-5678	Premium monthly plan for Speedy Garage	01-Nov-2023	01-Dec-2023
T005	Prosumer Monthly	James Anderson	\$20.00	Credit Card	NFT-1234-5678	Basic user subscription for Quick Auto Repair	01-Nov-2023	01-Dec-2023
T006	Prosumer Quarterly	Lisa Martinez	\$60.00	Credit Card	NFT-1234-5678	Quarterly subscription renewal for Elite Garage	01-Nov-2023	01-Feb-2024

**Screenshot 3: Owner Management**

This screen shows a list of garage owners:

ID	Name	Full Name	Email	Phone	Action
O001	GarageHub	GarageHub	info@garagehub.com	0987654321	View Details
O002	GarageHub Admin	GarageHub Admin	admin@garagehub.com	0987654321	View Details
O003	GarageHub Owners	GarageHub Owners	owners@garagehub.com	0987654321	View Details
O004	GarageHub Support	GarageHub Support	support@garagehub.com	0987654321	View Details

### 5.2.1 Owner management

```

1  create function show_owner()
2      returns TABLE(id integer, username character varying, display_name
3                      character varying, full_name character varying, city character
4                      varying, email character varying, telephone_no character varying)
5              language plpgsql
6  as
7  $$
8  BEGIN
9      RETURN QUERY
10     SELECT
11         o.id,
12         o.username,
13         o.display_name,
14         (o.first_name || ' ' || COALESCE(o.middle_name || ' ', '')) ||
15         o.last_name)::VARCHAR,
16         o.city,
17         o.email,
18         o.telephone_no
19     FROM
20         owner o
21     ORDER BY
22         o.id;
23 END;
24 $$;
25
26 alter function show_owner() owner to sc_db_project_user;

```

Listing 1: show\_owner.sql

The screenshot shows a dashboard titled "Owner Management". At the top, there are three summary cards: "Total Owners" (3), "Active Garages" (3), and "Verified Accounts" (3). Below these is a section titled "Garage Owners" with a sub-instruction "Manage all registered garage owner accounts". A table lists three garage owners with their details:

ID	Username	Display Name	Full Name	Email	Telephone	Actions
#1	garage_owner1	AutoPro Garage	Somchai Sukhumvit	somchai@autopro.com	0812345678	
#2	speedfix_motors	SpeedFix Motors	Nattapong K. Ratchada	nattapong@speedfix.com	0823456789	
#3	premier_auto	Premier Auto Service	Siriporn Pattanakarn	siriporn@premierauto.com	0834567890	

### Explanation (show\_owner.sql)

**Purpose:** Provides an overview of platform owners for administrative monitoring (support, billing, compliance).

**Logic:** Straightforward SELECT with ordered output by id for stable pagination/export. Middle name is conditionally appended using COALESCE.

### 5.2.2 Subscription plan management

```

1  create function show_subscription_plans()
2      returns TABLE(plan_id integer, plan_name character varying, details
3                      text, price_thb numeric, duration_days integer)
4          language plpgsql
5
6  as
7  $$
8  BEGIN
9      RETURN QUERY
10         SELECT
11             subscription_id,
12             name,
13             detail,
14             price,
15             duration
16         FROM "subscription_plan"
17         ORDER BY rank;
18
19 END;
$$;
20
21 alter function show_subscription_plans() owner to sc_db_project_user;

```

Listing 2: show\_subscription\_plans.sql

**Subscription Plan Management**

Create and manage subscription tiers for garage owners

Total Plans: 4      Monthly Plans: 2 (30D)      Quarterly Plans: 1 (90D)      Annual Plans: 1 (365D)

+ Create New Plan

**Subscription Plans**

Manage all subscription plans available to garage owners

Plan ID	Plan Name	Details	Price (THB)	Duration	Actions
PLAN001	Basic Monthly	Essential features for small garages. Includes customer management,	฿1,990	1 Month	<input type="checkbox"/> <input type="checkbox"/>
PLAN002	Professional Quarterly	Advanced features for growing businesses. Includes inventory manag...	฿5,490	3 Months	<input type="checkbox"/> <input type="checkbox"/>
PLAN003	Enterprise Annual	Complete solution for large operations. All features included plus prio...	฿19,990	1 Year	<input type="checkbox"/> <input type="checkbox"/>
PLAN004	Premium Monthly	Premium tier with unlimited features. Perfect for medium-sized garag...	฿3,990	1 Month	<input type="checkbox"/> <input type="checkbox"/>

#### Explanation (show\_subscription\_plans.sql listing)

**Purpose:** Displays all subscription plans available for purchase by owners, supporting SaaS monetization tiers.

**Logic:** Simple ordered SELECT (by rank) enabling prioritized display of featured plans.

### 5.2.3 Subscription transaction management

```
1 create function show_subscription_transaction()
2     returns TABLE(trans_id integer, owner_name character varying,
3                     plan_name character varying, amount numeric, payment_method
4                     character varying, date_purchased date, date_expired date, status
5                     text)
6     language plpgsql
7 as
8 $$
9 BEGIN
10    RETURN QUERY
11    SELECT
12        st.transaction_id,
13        o.display_name::VARCHAR,
14        sp.name::VARCHAR,
15        st.amount,
16        st.payment_method,
17        st.purchased_date,
18        st.expired_date,
19        CASE
20            WHEN st.expired_date < CURRENT_DATE THEN 'Expired'
21            ELSE 'Active'
22        END
23    FROM
24        subscription_transaction st
25        JOIN
26            subscription_plan sp ON st.subscription_id = sp.
27        subscription_id
28        JOIN
29            owner o ON st.owner_id = o.id
30    ORDER BY
31        st.purchased_date DESC;
32 END;
33 $$;
34
35 alter function show_subscription_transaction() owner to
36     sc_db_project_user;
```

Listing 3: show\_subscription\_transaction.sql

The screenshot shows a dashboard titled "Subscription Transactions". At the top, there are four summary cards: "Total Transactions 10", "Total Revenue ₧70,400", "This Month ₧0", and "Active Subscriptions 0". Below these are search and filter fields for "Search by owner, reference, or plan...", "All Methods", and "All Plans". A large section titled "All Transactions" displays a table of 6 rows, each representing a transaction with columns for Trans. ID, Subscription Plan, Owner Name, Amount, Payment Method, Reference ID, Description, Purchased, and Expires.

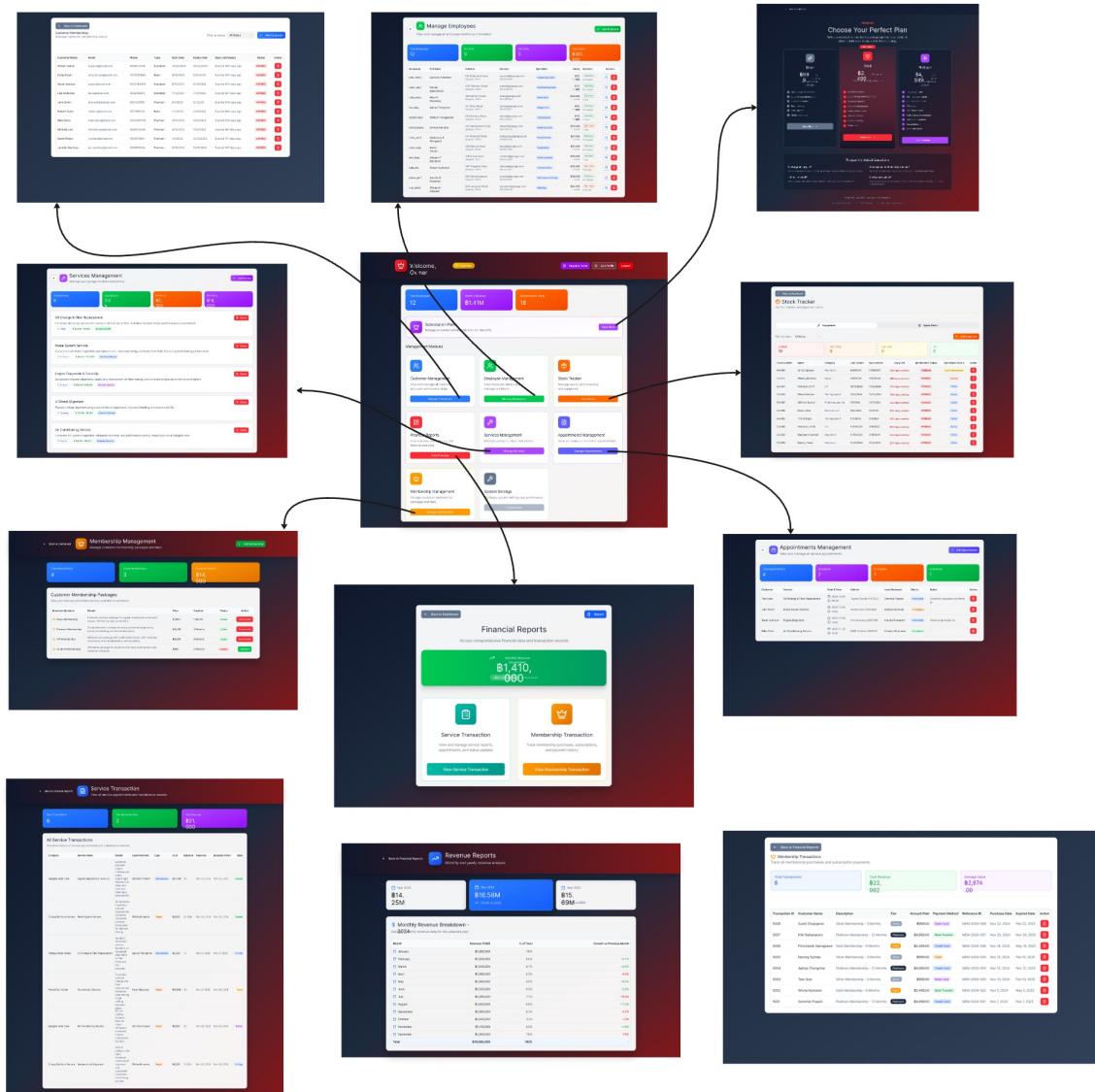
Trans. ID	Subscription Plan	Owner Name	Amount	Payment Method	Reference ID	Description	Purchased	Expires
#1001	Basic Monthly	Michael Johnson	฿1,990 THB	Credit Card	REF-2024-001	Monthly subscription renewal for AutoCare Garage	15 Nov 2024	15 Dec 2024 Expired
#1002	Professional Quarterly	Sarah Williams	฿5,490 THB	Bank Transfer	REF-2024-002	3-month subscription for Premium Auto Services	10 Nov 2024	10 Feb 2025 Expired
#1003	Enterprise Annual	David Chen	฿19,990 THB	Credit Card	REF-2024-003	Annual enterprise plan for MegaGarage Network	01 Nov 2024	01 Nov 2025 Expired
#1004	Premium Monthly	Emma Thompson	฿3,990 THB	PromptPay	REF-2024-004	Premium monthly plan for SpeedFix Garage	20 Nov 2024	20 Dec 2024 Expired
#1005	Basic Monthly	James Anderson	฿1,990 THB	Credit Card	REF-2024-005	Basic plan subscription for Quick Auto Repair	18 Nov 2024	18 Dec 2024 Expired
#1006	Professional Quarterly	Lisa Martinez	฿5,490 THB	Bank Transfer	REF-2024-006	Quarterly subscription renewal for Elite Motors	12 Nov 2024	12 Feb 2025 Pending

### Explanation (show\_subscription\_transaction.sql)

**Purpose:** Central audit view of subscription purchases supporting revenue tracking, churn analysis, and compliance reviews.

**Logic:** Joins transaction with plan and owner; CASE expression derives status relative to current date.

### 5.3 Owner side



### 5.3.1 Approved requests and create employee

```
1  create function approve_request_and_create_employee(p_email character
2      varying, p_salary numeric,
3          p_contract_type
4      character varying,
5          p_contract_period
6      character varying) returns integer
7      language plpgsql
8  as
9  $$
10 DECLARE
11 r RECORD;
12     v_new_username
13 VARCHAR;
14     v_new_password
15 VARCHAR;
16     v_employee_id
17 INTEGER;
18 BEGIN
19     SELECT *
20     INTO r
21     FROM "request_form"
22     WHERE request_form.email = p_email;
23
24     v_new_username
25 := lower(r.first_name || '.' || r.last_name || r.request_id);
26     -- Password: Default temporary password
27     v_new_password
28 := 'changeme1234';
29
30     v_employee_id
31 := add_employee(
32         r.company_id,
33         v_new_username,
34         r.first_name,
35         r.middle_name,
36         r.last_name,
37         r.address,
38         r.city,
39         r.postal_code,
40         r.email,
41         r.telephone_no,
42         r.specialize,
43         v_new_password,      -- Generated Password
44         p_salary,
45         CURRENT_DATE,
46         p_contract_type,
47         p_contract_period
48     );
49
50     -- delete the request
51     DELETE
52     FROM "request_form"
53     WHERE email = p_email;
54
55     RETURN v_employee_id;
56 END;
```

```
54 $$;  
55  
56 alter function approve_request_and_create_employee(varchar, numeric,  
           varchar, varchar) owner to sc_db_project_user;
```

Listing 4: approve\_request\_and\_create\_employee.sql

Pending Applications					
Request ID	Applicant Name	Contact Info	Specialize	Location	Action
#1001	Apinya Thongchai	apinya.t@email.com   0845678901	Brake Systems	Nonthaburi	<button>Approve</button>
#1002	Wichai Pongpanich	wichai.p@email.com   0856789012	Transmission	Samut Prakan	<button>Approve</button>
#1003	Siriwan Kamolrat	siriwan.k@email.com   0867890123	Diagnostics	Pathum Thani	<button>Approve</button>

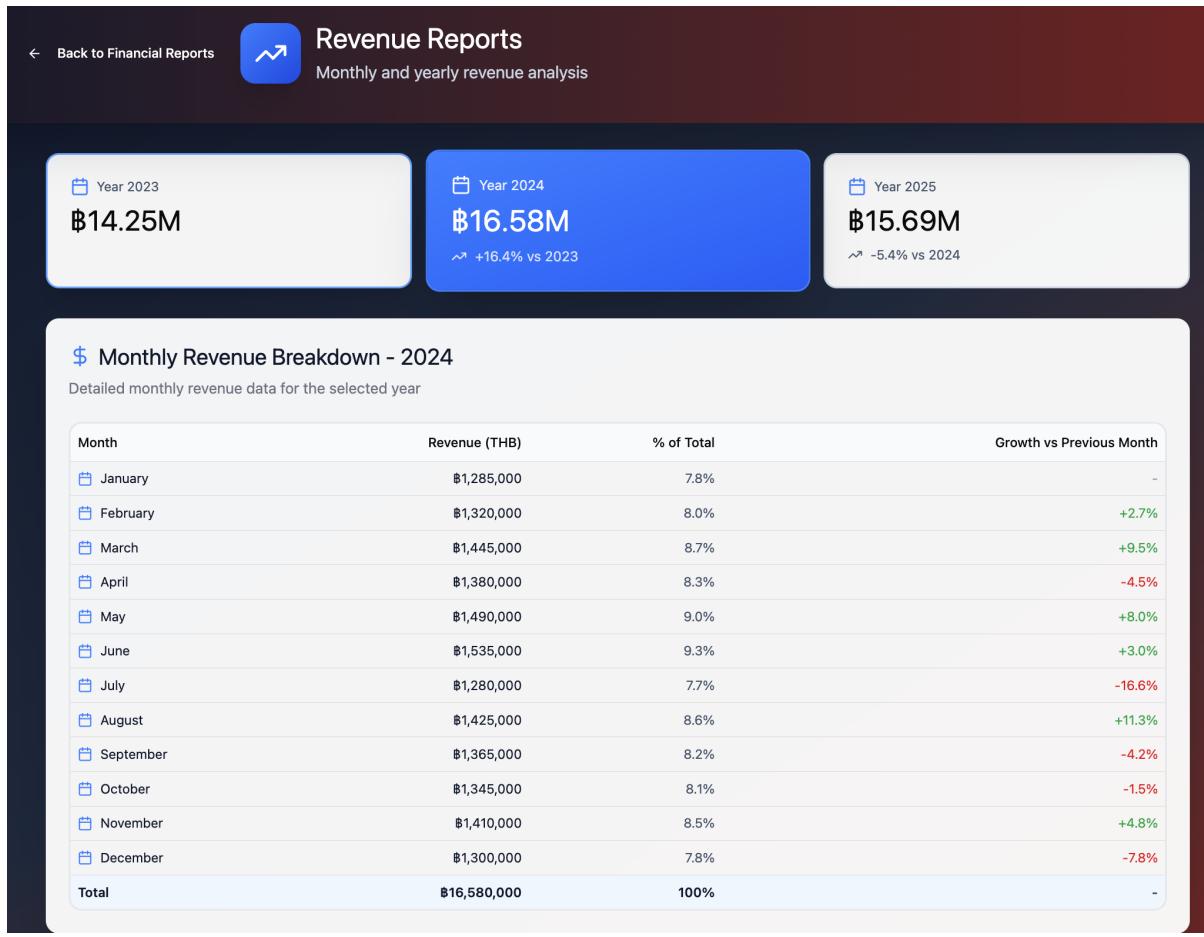
### Explanation (approve\_request\_and\_create\_employee.sql)

**Purpose:** Automates hiring: transforms a pending applicant (request form) into an employee record, then removes the source request to prevent duplication.

### 5.3.2 yearly financial summary

```
1 create function get_yearly_financial_summary(p_company_id integer,
2     p_year integer)
3     returns TABLE
4     (
5         year              integer,
6         total_revenue_year character varying,
7         total_net_income_year character varying
8     )
9     language plpgsql
10    as
11 $$
12 BEGIN
13     RETURN QUERY
14     SELECT f.year,
15            (COALESCE(SUM(total_revenue), 0.00)::VARCHAR || ' THB')
16            ::VARCHAR,
17            (COALESCE(SUM(net_income), 0.00)::VARCHAR || ' THB')::
18            VARCHAR
19     FROM "financial_record" AS f
20     WHERE company_id = p_company_id
21         AND f.year = p_year
22     group by f.year;
23 END;
24 $$;
25
26 alter function get_yearly_financial_summary(integer, integer) owner to
27     sc_db_project_user;
```

Listing 5: get\_yearly\_financial\_summary.sql



### Explanation (get\_yearly\_financial\_summary.sql)

**Purpose:** Produces aggregated yearly totals (revenue, net income) for dashboard totals and comparative analytics.

**Logic:** COALESCE + SUM across financial\_record filtered by company and year; formatted with currency suffix for direct UI display.

### 5.3.3 Appointment management

```

1  create function show_appointments(p_company_id integer)
2      returns TABLE(appointment_id integer, customer_name text,
3                      vehicle_info text, date_str text, time_str text, service_detail
4                      character varying, status character varying)
5              language plpgsql
6
7  as
8  $$
9  BEGIN
10     RETURN QUERY
11     SELECT
12         a.id AS appointment_id,
13         (c.first_name || ' ' || c.last_name)::TEXT,
14         (v.brand || ' ' || v.model)::TEXT,
15         to_char(a.appointment_date, 'MM/DD/YYYY')::TEXT,
16         to_char(a.appointment_date, 'HH12:MI AM')::TEXT,
17         a.details,
18         a.status
19     FROM "appointment" a
20             JOIN "vehicle" v ON a.vehicle_id = v.id
21             JOIN "customer" c ON v.owner_id = c.id
22     WHERE a.company_id = p_company_id
23
24     ORDER BY
25     CASE
26         WHEN a.status = 'pending' THEN 1
27         WHEN a.status = 'accepted' THEN 2
28         ELSE 3
29     END ,
30         a.appointment_date;
31
32 END;
33 $$;
34
35 alter function show_appointments(integer) owner to sc_db_project_user;

```

Listing 6: show\_appointments.sql

Customer	Service	Date & Time	Vehicle	Lead Mechanic	Status	Notes	Action
Test User	Oil Change & Filter Replacement	2025-12-01 09:00	Toyota Corolla (XYZ123)	Somchai Tanaka	Scheduled	Customer requested synthetic oil...	
John Smith	Brake System Service	2025-12-01 14:00	Honda Civic (ABC456)	Siriwan Kamolrat	In Progress	-	
Sarah Johnson	Engine Diagnostic	2025-12-02 10:00	Ford Mustang (DEF789)	Anucha Prasertsri	Scheduled	Check engine light on	
Mike Chen	Air Conditioning Service	2025-11-28 11:00	BMW 3 Series (GHI012)	Chalerm Boonmee	Completed	-	

#### Explanation (show\_appointments.sql)

**Purpose:** Retrieves enriched appointment listing for operational scheduling and status tracking.

ing.

**Logic:** Multi-table join (appointment, vehicle, customer) plus CASE order block ensuring pending/accepted appear first.

### 5.3.4 Customer management

```

1  create function show_company_customers(p_company_id integer)
2    returns TABLE(customer_name character varying, email character
3                  varying, phone character varying, membership_type character varying,
4                  start_date date, expiry_date date, days_until_expiry character
5                  varying, status character varying)
6      language plpgsql
7  as
8  $$
9  BEGIN
10    RETURN QUERY
11      SELECT (c.first_name || ' ' || c.last_name)::character varying
12        AS customer_name,
13          c.email,
14          c.telephone_no,
15          mp.name,
16          m.purchased_date,
17          m.expired_date,
18          CASE
19            WHEN m.expired_date < CURRENT_DATE THEN
20              'Expired ' || (CURRENT_DATE - m.expired_date)::VARCHAR || ' days ago'
21            WHEN m.expired_date = CURRENT_DATE THEN
22              'Expires today'
23            ELSE
24              (m.expired_date - CURRENT_DATE)::VARCHAR || ' days left'
25          END::VARCHAR,
26          CASE
27            WHEN m.expired_date < CURRENT_DATE THEN 'EXPIRED'
28            ELSE 'ACTIVE'
29          END::VARCHAR
30
31      FROM "customer" c
32        JOIN "membership" m ON c.id = m.customer_id
33        JOIN "membership_plan" mp ON m.membership_plan_id = mp
34        .membership_plan_id
35        WHERE mp.company_id = p_company_id
36        ORDER BY mp.rank DESC, m.expired_date DESC;
37  END;
38  $$;
39
40  alter function show_company_customers(integer) owner to
41  sc_db_project_user;

```

Listing 7: show\_company\_customers.sql

Customer Memberships  
Manage customer membership status

Filter by status: All Status + Add Customer

Customer Name	Email	Phone	Type	Start Date	Expiry Date	Days Until Expiry	Status	Action
William Garcia	w.garcia@email.com	5559012345	Standard	10/22/2023	10/22/2024	Expired 405 days ago	EXPIRED	
Emily Brown	emily.brown@email.com	5554567890	Basic	9/20/2023	9/20/2024	Expired 437 days ago	EXPIRED	
Sarah Johnson	sarah.j@email.com	5552345678	Standard	8/15/2023	8/15/2024	Expired 473 days ago	EXPIRED	
Lisa Anderson	lisa.a@email.com	5556789012	Standard	7/12/2023	7/12/2024	Expired 507 days ago	EXPIRED	
John Smith	john.smith@email.com	5551234567	Premium	6/1/2023	6/1/2024	Expired 548 days ago	EXPIRED	
Robert Taylor	robert.t@email.com	5557890123	Basic	11/1/2023	2/10/2024	Expired 660 days ago	EXPIRED	
Mike Davis	mike.davis@email.com	5553456789	Premium	3/10/2023	1/25/2024	Expired 676 days ago	EXPIRED	
Michelle Lee	michelle.lee@email.com	5550123456	Premium	4/14/2023	1/20/2024	Expired 681 days ago	EXPIRED	
David Wilson	d.wilson@email.com	5555678901	Premium	1/5/2023	12/31/2023	Expired 701 days ago	EXPIRED	
Jennifer Martinez	jen.martinez@email.com	5558901234	Premium	5/18/2023	11/15/2023	Expired 747 days ago	EXPIRED	

### Explanation (show\_company\_customers.sql)

**Purpose:** Surfaces membership retention context (expiry countdown, active state) for owner renewal strategies.

**Logic:** Joins customer, membership, and membership plan; CASE generates human-readable expiry messages.

### 5.3.5 Membership plan management

```

1  create function show_company_membership_plans(p_company_id integer)
2    returns TABLE(plan_id integer, plan_name character varying,
3                  plan_detail text, plan_rank integer, price_thb numeric,
4                  duration_days integer)
5    language plpgsql
6  as
7  $$
8  BEGIN
9    RETURN QUERY
10   SELECT
11     membership_plan_id,
12     name,
13     detail,
14     rank,
15     price,
16     duration
17   FROM "membership_plan"
18   WHERE company_id = p_company_id
19   ORDER BY rank ;
20
21 END;
22 $$;
23
24
25 alter function show_company_membership_plans(integer) owner to
26 sc_db_project_user;

```

Listing 8: show\_company\_membership\_plans.sql

Membership Name	Details	Price	Duration	Status	Action
Basic Membership	Essential services package for regular maintenance and basic repairs. Perfect for daily commuters.	\$1,500	1 Month	Active	<button>Deactivate</button>
Premium Membership	Comprehensive coverage including advanced diagnostics, priority scheduling, and discounted parts.	\$3,500	3 Months	Active	<button>Deactivate</button>
VIP Membership	Ultimate care package with unlimited services, 24/7 roadside assistance, and complimentary vehicle pickup.	\$9,500	6 Months	Active	<button>Deactivate</button>
Student Membership	Affordable package for students with basic maintenance and seasonal checkups.	\$800	2 Months	Inactive	<button>Activate</button>

### Explanation (show\_company\_membership\_plans.sql)

**Purpose:** Displays company-specific loyalty offerings with rank-based ordering.

### 5.3.6 Employment request management

```

1  create function show_company_requests(p_company_id integer)
2      returns TABLE(req_id integer, applicant_name text, contact_info
3                      text, specialize character varying, location text)
4      language plpgsql
5
6  as
7  $$
8  BEGIN
9      RETURN QUERY
10     SELECT
11         r.request_id,
12         (r.first_name || ' ' || r.last_name)::TEXT,
13         (r.email || ' | ' || r.telephone_no)::TEXT,
14         r.specialize,
15         (r.city || ' , ' || r.postal_code)::TEXT
16     FROM "request_form" r
17     WHERE r.company_id = p_company_id;
18
19 END;
$$;
20
21 alter function show_company_requests(integer) owner to
22 sc_db_project_user;

```

Listing 9: show\_company\_requests.sql

Pending Applications					
Request ID	Applicant Name	Contact Info	Specialize	Location	Action
#1001	Apinya Thongchai	apinya.t@email.com   0845678901	Brake Systems	Nonthaburi	<button>Approve</button>
#1002	Wichai Pongpanich	wichai.p@email.com   0856789012	Transmission	Samut Prakan	<button>Approve</button>
#1003	Siriwan Kamolrat	siriwan.k@email.com   0867890123	Diagnostics	Pathum Thani	<button>Approve</button>

#### Explanation (show\_company\_requests.sql)

**Purpose:** Lists employment application requests.

### 5.3.7 Employee schedule management

```

1  create function show_employee_schedule(p_employee_id integer)
2      returns TABLE(work_date text,
3                      day_name text,
4                      shift_time text,
5                      status character varying)
6      language plpgsql
7  as
8  $$
9  BEGIN
10     RETURN QUERY
11        SELECT
12            to_char(s.date, 'DD/MM/YYYY')::TEXT,
13            TRIM(to_char(s.date, 'Day'))::TEXT,
14            (to_char(s.start_time, 'HH24:MI') || ' - ' || to_char(s.
end_time, 'HH24:MI'))::TEXT,
15            s.status
16        FROM employee_schedule s
17        WHERE s.employee_id = p_employee_id
18        ORDER BY s.date;
19    END;
20  $$;
21
22 alter function show_employee_schedule(integer) owner to
sc_db_project_user;

```

Listing 10: show\_employee\_schedule.sql

Work Date	Day	Shift Time	Status
January 15, 2024	Monday	08:00 - 17:00	Working
January 16, 2024	Tuesday	08:00 - 17:00	Working
January 17, 2024	Wednesday	08:00 - 17:00	Working
January 18, 2024	Thursday	08:00 - 17:00	Sick
January 19, 2024	Friday	08:00 - 17:00	Working
January 20, 2024	Saturday	08:00 - 12:00	Working
January 22, 2024	Monday	08:00 - 17:00	Working
January 23, 2024	Tuesday	08:00 - 17:00	Leave

#### Explanation (show\_employee\_schedule.sql)

**Purpose:** Provides employee shift visibility—date, day name, time window, status.

### 5.3.8 Employee management

```
1 create function show_employee(p_company_id integer)
2     returns TABLE(company_name character varying, username character
3                     varying, first_name character varying, middle_name character varying
4                     , last_name character varying, address character varying, city
5                     character varying, postal_code character varying, email character
6                     varying, telephone_no character varying, specialize character
7                     varying)
8     language plpgsql
9
10    as
11    $$
12    BEGIN
13        RETURN QUERY
14            SELECT
15                (SELECT c.name FROM company AS c
16                 WHERE c.id = e.company_id),
17                e.username,
18                e.first_name,
19                e.middle_name,
20                e.last_name,
21                e.address,
22                e.city,
23                e.postal_code,
24                e.email,
25                e.telephone_no,
26                e.specialize
27            FROM employee e
28            WHERE e.company_id = p_company_id;
29    END;
30    $$;
31
32    alter function show_employee(integer) owner to sc_db_project_user;
```

Listing 11: show\_employee.sql

Username	Full Name	Address	Contact	Specialize	Salary	Contract	Actions
john_mech	Somchai A Rattana	123 Sukhumvit Road Bangkok, 10110	somchai@garage.com 0812345678	Engine Specialist	\$35,000 / month	Full-time Permanent	
sarah_elec	Pranee Suwannarat	456 Petchburi Road Bangkok, 10400	pranee@garage.com 0823456789	Electrical Systems	\$32,000 / month	Full-time Permanent	
mike_body	Niran K Chaiwong	789 Rama IV Road Bangkok, 10330	niran@garage.com 0834567890	Body Work	\$28,000 / month	Full-time 1 Year	
lisa_diag	Apinya Thongchai	321 Silom Road Bangkok, 10500	apinya@garage.com 0845678901	Diagnostics	\$30,000 / month	Full-time Permanent	
robert_trans	Wichai P Pongpanich	654 Wireless Road Bangkok, 10330	wichai@garage.com 0856789012	Transmission	\$33,000 / month	Full-time 2 Years	
emma_brake	Siriwan Kamolrat	987 Ratchadamri Road Bangkok, 10330	siriwan@garage.com 0867890123	Brake Systems	\$29,000 / month	Part-time 1 Year	
chris_paint	Nattapong S Wongwian	147 Ploenchit Road Bangkok, 10330	nattapong@garage.com 0878901234	Paint & Finish	\$27,000 / month	Full-time Permanent	
anna_susp	Kanya Somjai	258 Sathorn Road Bangkok, 10120	kanya@garage.com 0889012345	Suspension	\$31,000 / month	Full-time Permanent	

### Explanation (show\_employee.sql)

**Purpose:** Roster listing with personal + specialization data for management views.

**Logic:** Single table filtered by company id; potential to extend with role or permission columns.

### 5.3.9 Financial summary

```

1  create function show_financial_records(p_company_id integer)
2      returns TABLE(record_id integer, company_name character varying,
3                      period character varying, total_revenue character varying,
4                      net_income character varying, date_reported date)
5              language plpgsql
6  as
7  $$
8  BEGIN
9      RETURN QUERY
10         SELECT fr.record_id,
11             c.name::character varying,
12             (fr.month || '/' || fr.year)::character varying,
13             (fr.total_revenue::character varying || ' THB'),
14             (fr.net_income::character varying || ' THB'),
15             fr.date_reported
16         FROM "financial_record" fr
17             JOIN "company" c ON fr.company_id = c.id
18         WHERE fr.company_id = p_company_id
19         ORDER BY fr.year DESC, fr.month DESC;
20
21 END;
22 $$;
23
24 alter function show_financial_records(integer) owner to
25     sc_db_project_user;

```

Listing 12: show\_financial\_records.sql

The screenshot shows a web-based financial reporting interface. At the top, there's a header with a back button, a logo, and the title "Revenue Reports". Below the header, there are three large cards representing annual revenue:

- Year 2023:** \$14.25M
- Year 2024:** \$16.58M (with a note "+16.4% vs 2023")
- Year 2025:** \$15.69M (with a note "-5.4% vs 2024")

Below these cards, there's a section titled "\$ Monthly Revenue Breakdown - 2024" which says "Detailed monthly revenue data for the selected year". This section contains a table with the following data:

Month	Revenue (THB)	% of Total	Growth vs Previous Month
January	\$1,285,000	7.8%	-
February	\$1,320,000	8.0%	+2.7%
March	\$1,445,000	8.7%	+9.5%
April	\$1,380,000	8.3%	-4.5%
May	\$1,490,000	9.0%	+8.0%
June	\$1,535,000	9.3%	+3.0%
July	\$1,280,000	7.7%	-16.6%
August	\$1,425,000	8.6%	+11.3%
September	\$1,365,000	8.2%	-4.2%
October	\$1,345,000	8.1%	-1.5%
November	\$1,410,000	8.5%	+4.8%
December	\$1,300,000	7.8%	-7.8%
Total	\$16,580,000	100%	-

**Explanation (show\_financial\_records.sql)**

**Purpose:** Monthly ledger for trend analysis and gross/net performance tracking.

**Logic:** Join with company for naming; reverse chronological ordering surfaces recent performance first.

### 5.3.10 Membership transaction

```
1 create function show_membership_transactions(p_company_id integer)
2     returns TABLE
3     (
4         transaction_id integer,
5         customer_name text,
6         amount_paid text,
7         payment_method character varying,
8         reference_id character varying,
9         purchased_date date,
10        expired_date date,
11        description text
12    )
13    language plpgsql
14 as
15 $$$
16 BEGIN
17     RETURN QUERY
18     SELECT mt.id,
19             (c.first_name || ' ' || c.last_name)::TEXT AS
20         customer_name,
21             (mt.amount::TEXT || ' ' || mt.currency),
22             mt.payment_method,
23             mt.reference_id,
24             mt.purchased_date,
25             mt.expired_date,
26             mt.description
27
28     FROM "membership_transaction" mt
29             JOIN "membership" m ON mt.member_id = m.id
30             JOIN "customer" c ON m.id = c.member_id
31             JOIN "membership_plan" mp ON m.membership_plan_id = mp
32 .membership_plan_id
33
34     WHERE mp.company_id = p_company_id
35     ORDER BY mt.purchased_date DESC;
36
37 END;
38 $$;
39
40 alter function show_membership_transactions(integer) owner to
41 sc_db_project_user;
```

Listing 13: show\_membership\_transactions.sql

[← Back to Financial Reports](#)**👑 Membership Transactions**

Track all membership purchases and subscription payments

Total Transactions

8

Total Revenue

\$22,992

Average Value

\$2,874.00

Transaction ID	Customer Name	Description	Tier	Amount Paid	Payment Method	Reference ID	Purchase Date	Expired Date	Action
5008	Suwit Chaiyapruk	Silver Membership - 3 Months	Silver	\$999.00	Debit Card	MEM-2024-008	Nov 22, 2024	Feb 22, 2025	
5007	Kitti Rattanakorn	Platinum Membership - 12 Months	Platinum	\$4,999.00	Bank Transfer	MEM-2024-007	Nov 20, 2024	Nov 20, 2025	
5006	Pimchanok Saengkaew	Gold Membership - 6 Months	Gold	\$2,499.00	Credit Card	MEM-2024-006	Nov 18, 2024	May 18, 2025	
5005	Narong Suthep	Silver Membership - 3 Months	Silver	\$999.00	Cash	MEM-2024-005	Nov 15, 2024	Feb 15, 2025	
5004	Apinya Thongchai	Platinum Membership - 12 Months	Platinum	\$4,999.00	Credit Card	MEM-2024-004	Nov 12, 2024	Nov 12, 2025	
5003	Test User	Silver Membership - 3 Months	Silver	\$999.00	Debit Card	MEM-2024-003	Nov 10, 2024	Feb 10, 2025	
5002	Wichai Kumaran	Gold Membership - 6 Months	Gold	\$2,499.00	Bank Transfer	MEM-2024-002	Nov 5, 2024	May 5, 2025	
5001	Somchai Prasert	Platinum Membership - 12 Months	Platinum	\$4,999.00	Credit Card	MEM-2024-001	Nov 1, 2024	Nov 1, 2025	

**Explanation (show\_membership\_transactions.sql)**

**Purpose:** Detailed purchase history for membership monetization and customer lifecycle tracking.

**Logic:** Joins membership transaction with membership, customer, and plan; denormalizes names and descriptions.

### 5.3.11 Service Record

```

1  create function show_service_records(p_company_id integer)
2      returns TABLE
3      (
4          company_name      character varying,
5          service_name      character varying,
6          appointment_detail character varying,
7          lead_mechanic_name character varying,
8          record_type       character varying,
9          cost              integer,
10         duration_minutes integer,
11         received_date     date,
12         expected_finish_date date,
13         status             character varying
14     )
15     language plpgsql
16 as
17 $$$
18 BEGIN
19     RETURN QUERY
20         SELECT c.name::VARCHAR
21             AS
22             company_name,
23             s.name::VARCHAR
24             AS
25             service_name,
26             COALESCE(a.details, 'N/A')::VARCHAR
27             AS
28             appointment_detail,
29             (e.first_name || ' ' || e.last_name)::VARCHAR AS
30             lead_mechanic_name,
31
32             sr.type,
33             sr.cost,
34             sr.duration,
35             sr.received_date,
36             sr.expected_finish_date,
37             sr.status
38
39         FROM "service_record" sr
40             JOIN "service" s ON sr.service_id = s.id
41             JOIN "company" c ON s.company_id = c.id
42             JOIN "appointment" a ON sr.appointment_id = a.id
43             LEFT JOIN "employee" e ON sr.lead_mechanic_id = e.id
44
45         WHERE s.company_id = p_company_id
46         ORDER BY sr.received_date DESC;
47     END;
48 $$;
49
50 alter function show_service_records(integer) owner to
51     sc_db_project_user;

```

Listing 14: show\_service\_records.sql

**Service Transaction**  
View all service appointments and maintenance records

Company	Service Name	Details	Lead Mechanic	Type	Cost	Duration	Received	Expected Finish	Status
Bangkok Auto Care	Engine Diagnostic & Tune-Up	Customer reported engine misfiring and check engine light. Requires full diagnostic scan and spark plug replacement.	Somchai Prasert	Maintenance	\$12,500	3h	Nov 25, 2025	Nov 25, 2025	Completed
Chiang Mai Auto Service	Brake System Service	Annual brake inspection and pad replacement. Customer requested premium brake pads for highway driving.	Wichai Kumaran	Repair	\$8,500	2h 30m	Nov 26, 2025	Nov 26, 2025	Completed
		Standard 10,000 km service.							

### Explanation (show\_service\_records.sql)

**Purpose:** Operational and historical view of service execution, including lead mechanic and timing.

**Logic:** Multi-table joins; COALESCE safeguards missing appointment details.

### 5.3.12 Subscription plan

```

1  create function show_subscription_plans()
2    returns TABLE(plan_id integer, plan_name character varying, details
3      text, price_thb numeric, duration_days integer)
4      language plpgsql
5
6  as
7  $$
8  BEGIN
9    RETURN QUERY
10   SELECT
11     subscription_id,
12     name,
13     detail,
14     price,
15     duration
16   FROM "subscription_plan"
17   ORDER BY rank;
18
19 END;
$$;
20
21 alter function show_subscription_plans() owner to sc_db_project_user;

```

Listing 15: show\_subscription\_plans.sql

**Premium Plans**

## Choose Your Perfect Plan

Select a subscription plan that fits your garage business needs. All plans include core features with flexible pricing.

Plan	Price	Duration	Description
Silver	\$999	/ 3 months	Perfect for small garages
Gold	\$2,499	/ 6 months	Perfect for growing businesses
Platinum	\$4,999	/ 12 months	Perfect for enterprise garages

**Most Popular**

**Silver**

**Gold**

**Platinum**

**Features:**

- Basic garage management
- Up to 50 appointments/month
- Customer database
- Basic reporting
- Email support
- Mobile app access
- Everything in Silver
- Up to 200 appointments/month
- Advanced analytics
- Inventory management
- Priority email support
- SMS notifications
- Custom branding
- Export data
- Everything in Gold
- Unlimited appointments
- Multi-location support
- API access
- 24/7 phone support
- Dedicated account manager
- Advanced integrations
- Custom features
- White-label option

**Select Silver →**

**Select Gold →**

**Select Platinum →**

### Explanation (show\_subscription\_plans.sql owner view)

**Purpose:** Owner-facing list of available subscription options with pricing and duration.

## 5.4 Employee side

The image displays the employee dashboard interface, which includes the following components:

- Today's Appointments:** Shows 12 appointments from yesterday.
- Active Members:** Shows 248 members with a 98% retention rate.
- Parts in Stock:** Shows 1,247 parts available.
- Appointments:** A card to view and manage customer appointments.
- Customer Membership:** A card to manage customer memberships and status.
- Inventory & Storage:** A card to manage spare parts inventory and stock.

Below the dashboard, there are two additional tables:

- Customer Memberships:** A table listing customer names, emails, phone numbers, and membership types.
- Stock Tracker:** A table listing equipment and spare parts with details like serial number, name, category, last service date, next service date, days left, maintenance status, operational status, and actions.

They shared some features with the owner.

eg. show\_appointments.sql, show\_employee\_schedule.sql, show\_service\_records.sql

#### 5.4.1 Change appointment status

```
1 create function update_appointment_status(p_appointment_id integer ,
2     p_new_status character varying) returns integer
3 language plpgsql
4 as
5 $$
6 DECLARE
7     v_rows_updated INTEGER;
8 BEGIN
9     UPDATE "appointment"
10    SET status = p_new_status
11   WHERE id = p_appointment_id;
12   GET DIAGNOSTICS v_rows_updated = ROW_COUNT;
13   IF v_rows_updated > 0 THEN
14       RETURN 1;
15   ELSE
16       RETURN 0;
17   END IF;
18 END;
19 $$;
20 alter function update_appointment_status(integer , varchar) owner to
sc_db_project_user;
```

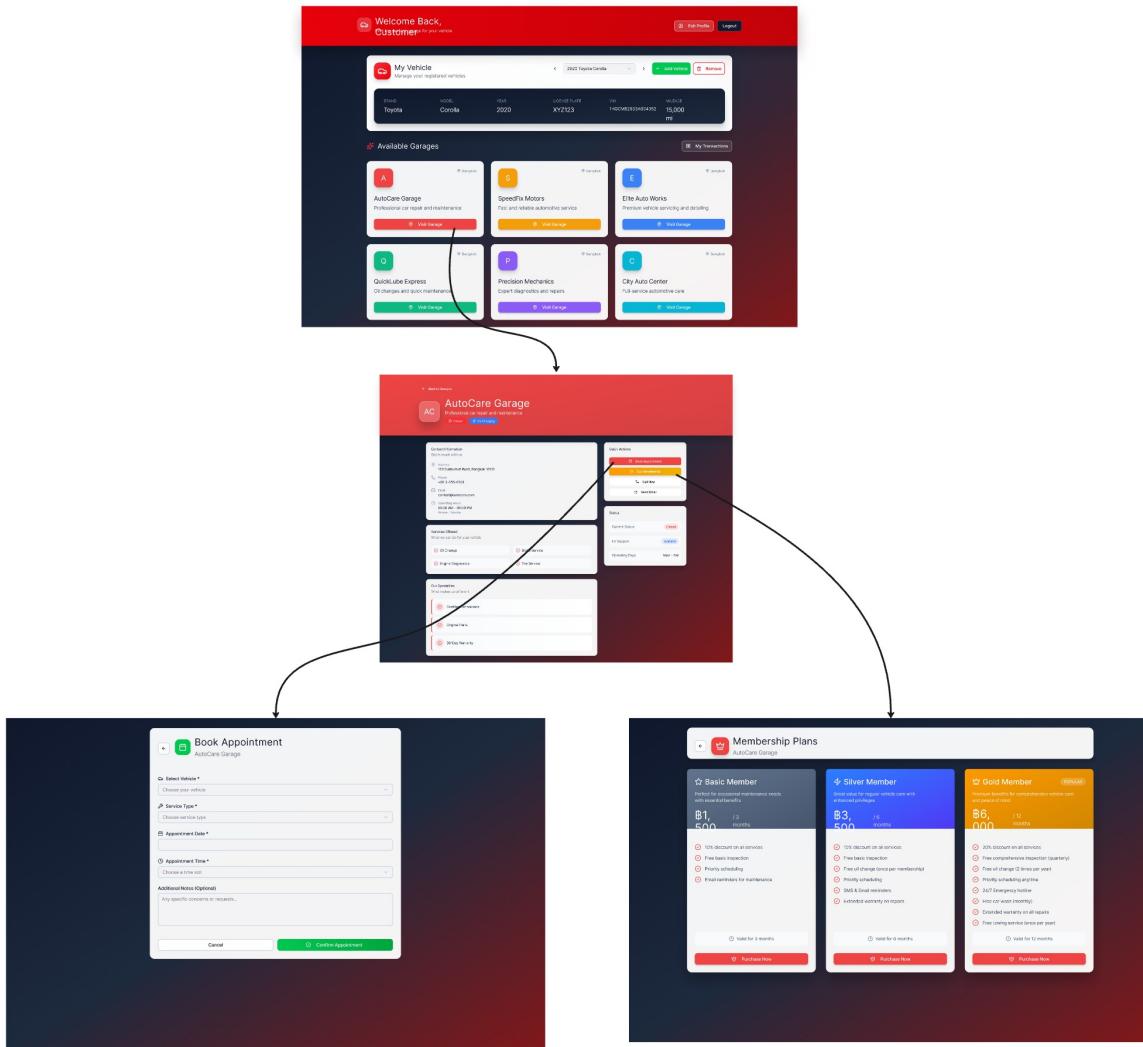
Listing 16: update\_appointment\_status.sql

#### Explanation (update\_appointment\_status.sql)

**Purpose:** Performs controlled state mutation on a single appointment row.

**Logic:** UPDATE followed by row count diagnostic; returns 1 (success) or 0 (no row changed).

## 5.5 Customer side



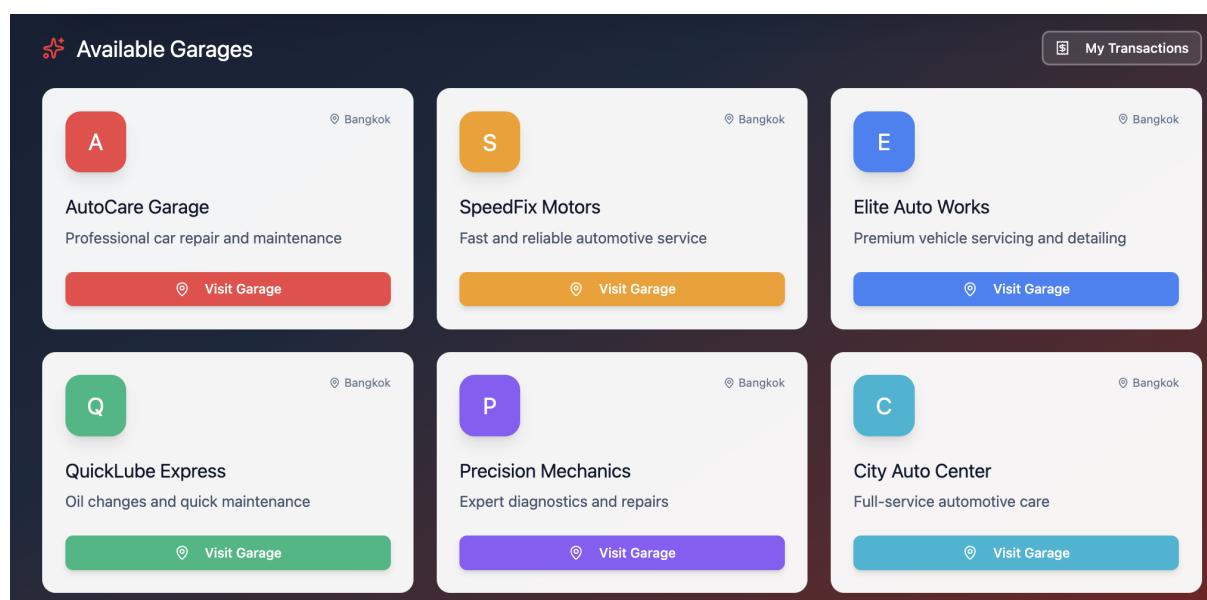
### 5.5.1 Companys menu

```

1 create function show_company()
2     returns TABLE(owner_full_name character varying, company_name
3         character varying, logo_url character varying, address character
4         varying, city character varying, postal_code character varying,
5         capacity integer, open_time time without time zone, close_time time
6         without time zone, social_media_contact character varying,
7         ev_support boolean)
8     language plpgsql
9
10    as
11    $$
12    BEGIN
13        RETURN QUERY
14            SELECT
15                (e.first_name || ' ' || e.last_name)::VARCHAR AS
16                owner_full_name,
17                c.name,
18                c.logo_url,
19                c.address,
20                c.city,
21                c.postal_code,
22                c.capacity,
23                c.open_time,
24                c.close_time,
25                c.social_media_contact,
26                c.ev_support
27
28        FROM company c
29                JOIN employee e ON c.employee_id = e.id;
30    END;
31    $$;
32
33    alter function show_company() owner to sc_db_project_user;

```

Listing 17: show\_company.sql



### Explanation (show\_company.sql)

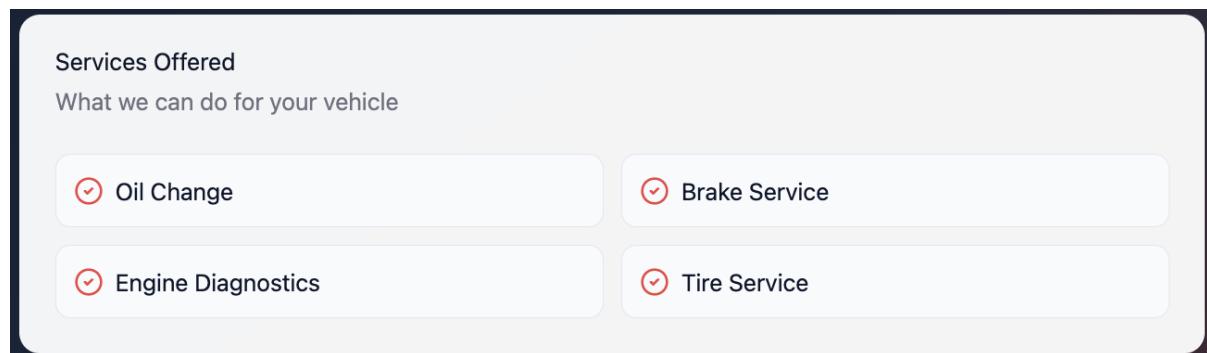
**Purpose:** Customer discovery of available garages with key operational attributes.

**Logic:** Join company with an employee (likely owner mapping) to present ownership context.

### 5.5.2 Service menu

```
1 create function show_company_services(p_company_id integer)
2     returns TABLE(service_name character varying, service_detail
3                     character varying, duration_hour numeric, minimum_cost numeric,
4                     maximum_cost numeric)
5             language plpgsql
6 as
7 $$
8 BEGIN
9     RETURN QUERY
10    SELECT
11        name,
12        detail,
13        approximate_duration,
14        min_cost,
15        max_cost
16    FROM "service"
17    WHERE company_id = p_company_id;
18 END;
19 $$;
20 alter function show_company_services(integer) owner to
21 sc_db_project_user;
```

Listing 18: show\_company\_services.sql



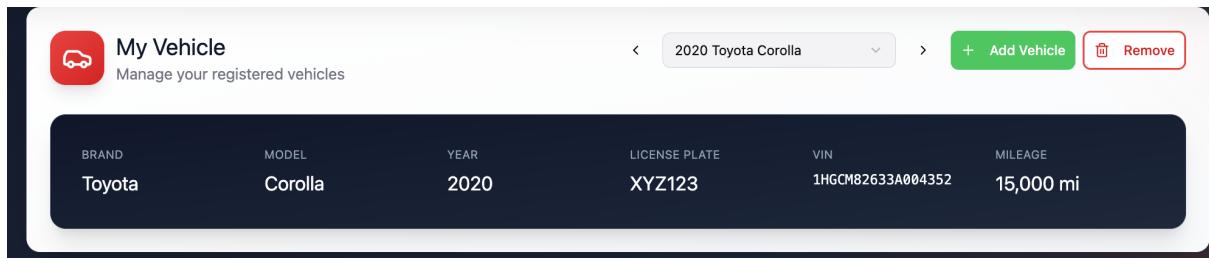
#### Explanation (show\_company\_services.sql)

**Purpose:** Displays service menu with estimated time and price band for booking decisions.

### 5.5.3 Vehicle management

```
1 create function show_vehicle_by_owner_id(p_owner_id integer)
2     returns TABLE(p_brand character varying, p_model character varying,
3                     p_year integer, p_license_plate character varying, p_vin character
4                     varying, p_mileage integer)
5             language plpgsql
6 as
7 $$
8 BEGIN
9     RETURN QUERY
10    SELECT
11        v.brand,
12        v.model,
13        v.year,
14        v.license_plate,
15        v.vin,
16        v.mileage
17    FROM "vehicle" v
18    WHERE v.owner_id = p_owner_id;
19
20 END;
21 $$;
22
23 alter function show_vehicle_by_owner_id(integer) owner to
24 sc_db_project_user;
```

Listing 19: show\_vehicle\_by\_owner\_id.sql



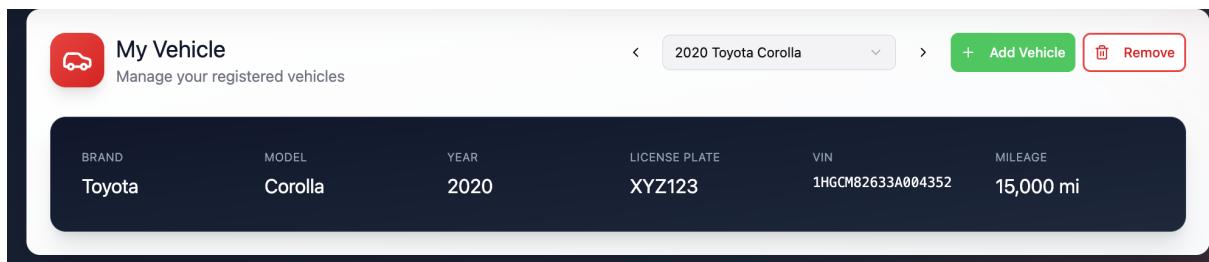
#### Explanation (show\_vehicle\_by\_owner\_id.sql)

**Purpose:** Enables customers to select from their registered vehicles during appointment creation.

#### 5.5.4 Remove vehicle

```
1 create function remove_vehicle(p_vin character varying) returns boolean
2     language plpgsql
3 as
4 $$
5 DECLARE
6     v_deleted_count INT;
7 BEGIN
8     DELETE FROM "vehicle"
9     WHERE vin = p_vin;
10
11    -- Check if a row was actually deleted
12    GET DIAGNOSTICS v_deleted_count = ROW_COUNT;
13
14    -- Return TRUE if deleted, FALSE if not found or mismatch
15    RETURN v_deleted_count > 0;
16 END;
17 $$;
18
19 alter function remove_vehicle(varchar) owner to sc_db_project_user;
```

Listing 20: remove\_vehicle.sql



#### Explanation (remove\_vehicle.sql)

**Purpose:** Deletes a vehicle no longer owned or mistakenly registered.

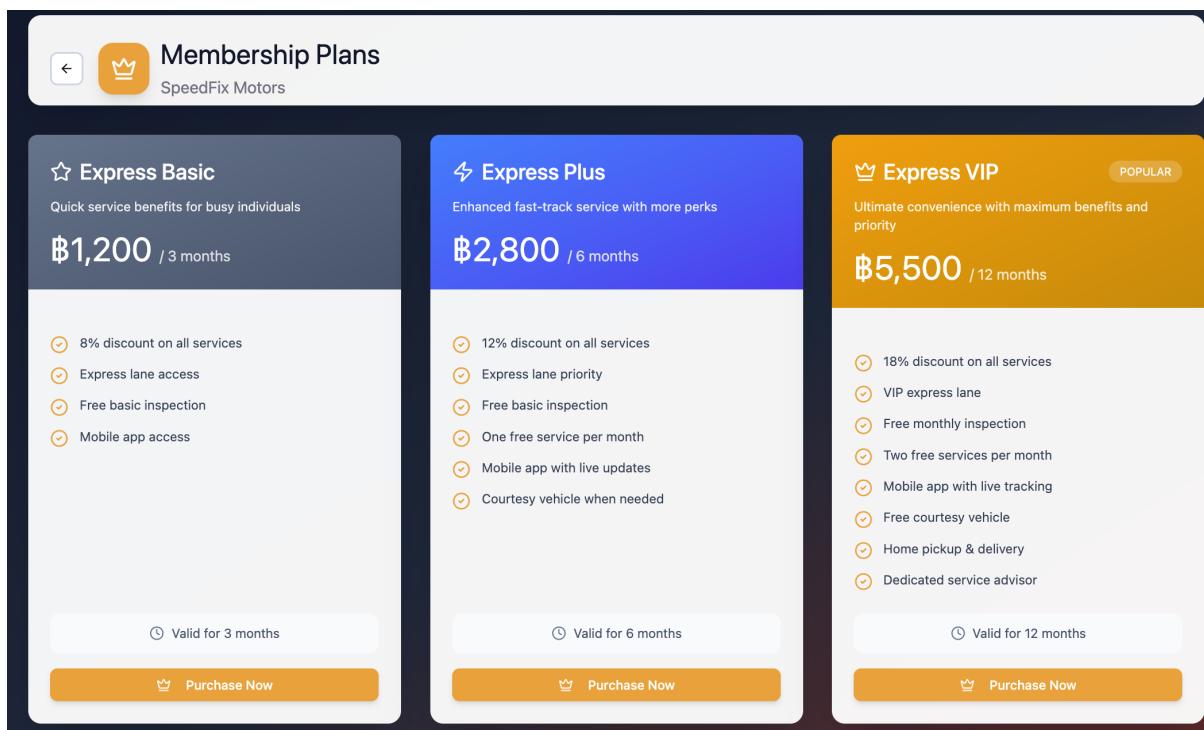
**Logic:** DELETE by VIN, returns TRUE if any row removed.

```

1  create function show_membership_plans_for_customers(p_company_id
2      integer)
3      returns TABLE
4      (
5          plan_name      character varying,
6          plan_detail   text,
7          plan_rank     integer,
8          price_thb     numeric,
9          duration_days integer
10         )
11    language plpgsql
12  as
13 $$$
14 BEGIN
15     RETURN QUERY
16     SELECT name,
17            detail,
18            rank,
19            price,
20            duration
21     FROM "membership_plan"
22     WHERE company_id = p_company_id
23     AND is_active = TRUE -- Only show active plans
24     ORDER BY rank;
25 END;
26 $$;
27 alter function show_membership_plans_for_customers(integer) owner to
sc_db_project_user;

```

Listing 21: show\_membership\_plans\_for\_customers.sql

**Explanation (show\_membership\_plans\_for\_customers.sql)**

**Purpose:** Customer-facing list of purchasable membership plans filtered to active ones for a company.

**Logic:** Ordered by rank to highlight premium tiers; exposes duration for expiry calculation post-purchase.

## 6 Table details

### 6.1 Company table

	<code>id</code>	<code>owner_id</code>	<code>name</code>	<code>logo_url</code>	<code>address</code>	<code>city</code>
1	1	1	AutoCare Garage	<a href="https://img.icons8.com/color/48/garage.png">https://img.icons8.com/color/48/garage.png</a>	123 Sukhumvit Road	Bangkok
2	2	2	SpeedFix Motors	<a href="https://img.icons8.com/color/48/f1-car.png">https://img.icons8.com/color/48/f1-car.png</a>	456 Phetchaburi Road	Bangkok
3	3	3	Elite Auto Works	<a href="https://img.icons8.com/color/48/expensive-price.png">https://img.icons8.com/color/48/expensive-price.png</a>	789 Thong Lor	Bangkok
4	4	1	Grand Opening Test Garage	<a href="http://test-logo.com/img.png">http://test-logo.com/img.png</a>	999 Testing Blvd	Bangkok
5	5	4	Pattaya Wheels	<null>	404 Wheel Ave	Pattaya
6	6	5	Drift King Garage	<a href="http://img.com/drift.png">http://img.com/drift.png</a>	505 Drift Soi	Bangkok
7	7	6	Phuket Body Shop	<null>	606 Paint Blvd	Phuket
8	8	7	Isan Oil Change	<null>	707 Oil Way	Khon Kaen
9	9	8	Mystery Machine Fix	<null>	808 Mystery Ln	Bangkok
10	10	9	Fast Lane Service	<a href="http://img.com/fast.png">http://img.com/fast.png</a>	909 Fast St	Bangkok
11	11	10	Northern Tires	<null>	1010 Weed St	Chiang Rai

#### 6.1.1 Purpose

Stores all companies (garages) using the platform.

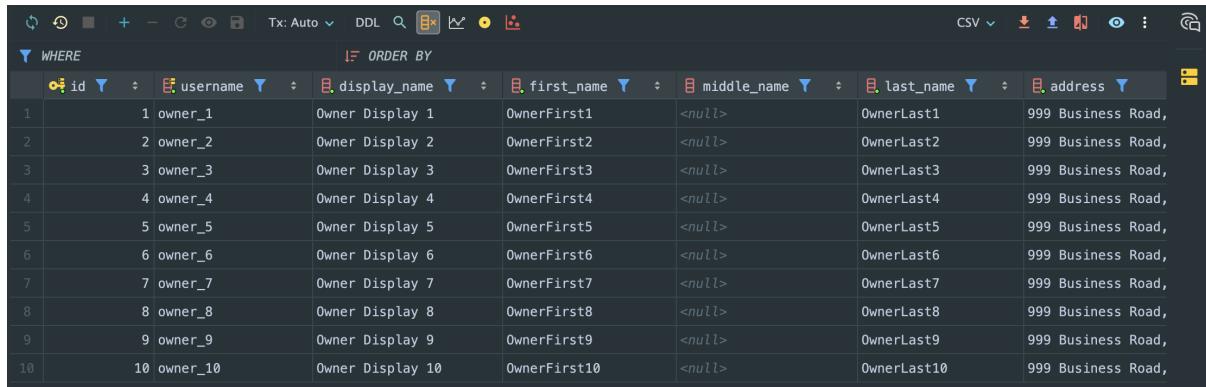
#### 6.1.2 Important Columns

- `id`: Surrogate identifier.
- `owner_id`: Owning account reference.
- `name`: Company/garage name.
- `logo_url`: Company logo URL.
- `address`: Physical address.
- `city`: City location.
- `postal_code`: Postal code.
- `capacity`: Service bay/workload capacity.
- `open_time`: Operating hours.
- `close_time`: Operating hours.
- `social_media_contact`: Public handle/contact.
- `ev_support`: EV servicing capability flag.

#### 6.1.3 Keys & Constraints

- PK: `id`
- FK: `owner_id` → `owner.id`

## 6.2 Owner table



The screenshot shows a database table named 'Owner' with 10 rows of data. The columns are: id, username, display\_name, first\_name, middle\_name, last\_name, and address. The data is as follows:

	id	username	display_name	first_name	middle_name	last_name	address
1	1	owner_1	Owner Display 1	OwnerFirst1	<null>	OwnerLast1	999 Business Road,
2	2	owner_2	Owner Display 2	OwnerFirst2	<null>	OwnerLast2	999 Business Road,
3	3	owner_3	Owner Display 3	OwnerFirst3	<null>	OwnerLast3	999 Business Road,
4	4	owner_4	Owner Display 4	OwnerFirst4	<null>	OwnerLast4	999 Business Road,
5	5	owner_5	Owner Display 5	OwnerFirst5	<null>	OwnerLast5	999 Business Road,
6	6	owner_6	Owner Display 6	OwnerFirst6	<null>	OwnerLast6	999 Business Road,
7	7	owner_7	Owner Display 7	OwnerFirst7	<null>	OwnerLast7	999 Business Road,
8	8	owner_8	Owner Display 8	OwnerFirst8	<null>	OwnerLast8	999 Business Road,
9	9	owner_9	Owner Display 9	OwnerFirst9	<null>	OwnerLast9	999 Business Road,
10	10	owner_10	Owner Display 10	OwnerFirst10	<null>	OwnerLast10	999 Business Road,

### 6.2.1 Purpose

Stores account information for company owners who administer garages.

### 6.2.2 Important Columns

- **id**: Surrogate identifier.
- **username** (unique): Username for login.
- **email** (unique): Email address for contact.
- **display\_name**: Name shown publicly.
- **first\_name**: Owner's first name.
- **middle\_name**: Owner's middle name.
- **last\_name**: Owner's last name.
- **address**: Owner's address.
- **city**: Owner's city.
- **postal\_code**: Owner's postal code.
- **telephone\_no**: Owner's telephone number.
- **password**: Hashed password.

### 6.2.3 Keys & Constraints

- PK: **id**
- Unique: **username**, **email**

### 6.3 Customer table

	<code>id</code>	<code>username</code>	<code>display_name</code>	<code>first_name</code>	<code>middle_name</code>	<code>last_name</code>	<code>address</code>
1	1	john_d	Johnny	John	<null>	Doe	123 Main St
2	2	jane_s	Janey	Jane	<null>	Smith	456 Silom
3	3	bob_b	Bobby	Bob	A	Brown	789 Sukhumvit
4	4	alice_w	Alice	Alice	<null>	White	321 Thonglo
5	6	david_b	Dave	David	<null>	Black	987 Rama 9
6	7	eve_p	Eve	Eve	<null>	Purple	147 Ladprao
7	8	frank_y	Frank	Frank	<null>	Yellow	258 Bangna
8	9	grace_o	Grace	Grace	<null>	Orange	369 Vibhavadi
9	10	hank_r	Hank	Henry	<null>	Red	741 Asoke

#### 6.3.1 Purpose

End-user accounts (vehicle owners) who book services.

#### 6.3.2 Important Columns

- `id`: Surrogate identifier.
- `username` (unique): Username for login.
- `email` (unique): Email address for contact.
- `display_name`: Name shown publicly.
- `first_name`: Customer's first name.
- `middle_name`: Customer's middle name.
- `last_name`: Customer's last name.
- `address`: Customer's address.
- `city`: Customer's city.
- `postal_code`: Customer's postal code.
- `telephone_no`: Customer's telephone number.
- `password`: Hashed password.

#### 6.3.3 Keys & Constraints

- PK: `id`
- Unique: `username`, `email`

## 6.4 Employee table

		id	company_id	username	first_name	middle_name	last_name	address
1		1	2	emp_mike	Mike	<null>	Mechanic	1 Garage Rd
2		2	2	emp_sarah	Sarah	<null>	Fix	2 Garage Rd
3		3	1	emp_tom	Tom	<null>	Tool	3 Garage Rd
4		4	3	emp_jerry	Jerry	<null>	Can	4 Garage Rd
5		5	3	emp_polly	Polly	<null>	Paint	5 Garage Rd
6		6	1	emp_andy	Andy	<null>	Air	6 Garage Rd
7		7	2	emp_betty	Betty	<null>	Brake	7 Garage Rd
8		8	1	emp_carl	Carl	<null>	Clutch	8 Garage Rd
9		9	3	emp_danny	Danny	<null>	Dent	9 Garage Rd
10		10	2	emp_eddie	Eddie	<null>	Engine	10 Garage Rd
11		11	1	somchai_jaidee2	Somchai	<null>	Jaidee	44 Sukhumvit Rd

### 6.4.1 Purpose

Company staff records with identity and employment details.

### 6.4.2 Important Columns

- **id**: Surrogate identifier.
- **company\_id**: Employer company reference.
- **username** (unique): Username for login.
- **email** (unique): Email address for contact.
- **first\_name**: Employee's first name.
- **middle\_name**: Employee's middle name.
- **last\_name**: Employee's last name.
- **address**: Employee's address.
- **city**: Employee's city.
- **postal\_code**: Employee's postal code.
- **telephone\_no**: Employee's telephone number.
- **specialize**: Employee's specialization.
- **password**: Hashed password.
- **salary** (default 15000.00): Monthly salary.
- **hire\_date** (default CURRENT\_DATE): Employment start date.
- **contract\_type** (default 'Full-Time'): Employment contract type.
- **contract\_period**: Duration of the contract.

### 6.4.3 Keys & Constraints

- PK: **id**
- FK: **company\_id** → **company.id**
- Unique: **username**, **email**

## 6.5 Employee Schedule table

	WHERE	ORDER BY				
	schedule_id	employee_id	date	start_time	end_time	status
1	1	1	2025-11-27	09:00:00	18:00:00	Working
2	2	1	2025-11-28	09:00:00	18:00:00	Working
3	3	1	2025-11-29	09:00:00	18:00:00	Working
4	4	1	2025-11-30	09:00:00	18:00:00	Working
5	5	1	2025-12-01	09:00:00	18:00:00	Working
6	6	1	2025-12-02	09:00:00	18:00:00	Working
7	7	1	2025-12-03	09:00:00	18:00:00	Working
8	8	2	2025-11-27	09:00:00	18:00:00	Working
9	9	2	2025-11-28	09:00:00	18:00:00	Working
10	10	2	2025-11-29	09:00:00	18:00:00	Working

### 6.5.1 Purpose

Work schedules and shift assignments for employees.

### 6.5.2 Important Columns

- **schedule\_id**: Surrogate identifier.
- **employee\_id**: Employee reference.
- **date**: Scheduled date.
- **start\_time**: Shift start time.
- **end\_time**: Shift end time.
- **status**: Schedule status (e.g., 'scheduled', 'completed', 'canceled').

### 6.5.3 Keys & Constraints

- PK: **schedule\_id**
- FK: **employee\_id** → **employee.id**

## 6.6 Equipment table

	<code>id</code>	<code>company_id</code>	<code>name</code>	<code>serial_number</code>	<code>purchase_cost</code>	<code>last_service_date</code>
1	1	1	Two-Post Hydraulic Lift	LFT-2023-001	120000.00	2024-01-15
2	2	1	Industrial Air Compressor	AIR-COMP-55	35000.00	2024-06-01
3	3	1	OBD-II Advanced Scanner	OBD-SCAN-99	15000.00	2023-11-20
4	4	2	3D Wheel Alignment System	WHEEL-3D-X	250000.00	2024-03-10
5	5	2	Automatic Tire Changer	TIRE-CH-200	85000.00	2024-05-20
6	6	2	Digital Wheel Balancer	BAL-DIGI-50	60000.00	2024-02-15
7	7	3	Dust-Free Paint Booth	PNT-BOOTH-01	500000.00	2024-01-01
8	8	3	Dual Action Polisher	POL-DA-PRO	12000.00	2024-07-01
9	9	3	ECU Remapping Tool	ECU-MAP-V3	45000.00	2024-04-10
10	10	1	Hydraulic Lift	LFT-001	50000.00	2023-01-15

### 6.6.1 Purpose

Durable equipment/assets tracked per company.

### 6.6.2 Important Columns

- `id`: Surrogate identifier.
- `company_id`: Company reference.
- `name`: Equipment name.
- `serial_number`: Unique serial number.
- `purchase_cost`: Cost at purchase.
- `last_service_date`: Date of last service.
- `category`: Equipment category.
- `quantity` (default 1): Number of units.
- `purchased_date` (default CURRENT\_DATE): Date purchased.
- `next_service_date`: Scheduled next service.
- `operational_status` (default 'active'): Current status.

### 6.6.3 Keys & Constraints

- PK: `id`
- FK: `company_id` → `company.id` (ON DELETE CASCADE)

## 6.7 Vehicle table

	<code>id</code>	<code>owner_id</code>	<code>brand</code>	<code>model</code>	<code>year</code>	<code>license_plate</code>	<code>vin</code>	<code>mileage</code>
1	1	1	Toyota	Camry	2020	1ABC1234	VIN1234567890001	50
2	2	2	Honda	Civic	2019	2DEF5678	VIN1234567890002	60
3	3	3	Mazda	3	2021	3GHI9012	VIN1234567890003	30
4	4	4	Nissan	Almera	2018	4JKL3456	VIN1234567890004	80
5	5	5	Ford	Ranger	2022	5MNN07890	VIN1234567890005	20
6	6	6	Isuzu	D-Max	2020	6PQR1234	VIN1234567890006	55
7	7	7	BMW	320d	2021	7STU5678	VIN1234567890007	15
8	8	8	Benz	C200	2019	8VWX9012	VIN1234567890008	40
9	9	9	Suzuki	Swift	2017	9YZA3456	VIN1234567890009	90
10	10	10	MG	ZS	2022	1BCD7890	VIN1234567890010	10

### 6.7.1 Purpose

Vehicles registered by customers; used for appointments and services.

### 6.7.2 Important Columns

- `id`: Surrogate identifier.
- `owner_id`: Vehicle owner reference.
- `brand`: Vehicle brand/make.
- `model`: Vehicle model.
- `year`: Manufacturing year.
- `mileage`: Current mileage.
- `license_plate`: Unique license plate number.
- `vin`: Unique vehicle identification number.

### 6.7.3 Keys & Constraints

- PK: `id`
- FK: `owner_id` → `customer.id` (ON DELETE SET NULL)
- Unique: `license_plate`, `vin`

## 6.8 Service table

	<code>id</code>	<code>company_id</code>	<code>name</code>	<code>detail</code>	<code>approximate_duration</code>	<code>min_cost</code>	<code>max_cost</code>
1	1	1	Oil Change	Synthetic Oil	45	1200.00	1800.00
2	2	1	Brake Check	Check Pads	60	500.00	800.00
3	3	2	Wheel Alignment	Laser Align	60	800.00	1500.00
4	4	3	Ceramic Coating	3 Layers	360	8000.00	15000.00
5	5	1	Standard Oil Change	Replace oil and filter	45	800.00	1500.00
6	6	1	Brake Inspection	Check pads and rotors	30	300.00	500.00
7	7	2	EV Battery Health Check	Scan battery cells	60	1500.00	2000.00
8	8	3	Tire Rotation	Rotate 4 wheels	40	400.00	600.00
9	9	4	Full Body Polish	Wash, clay bar, polish	240	2500.00	4000.00
10	10	5	ECU Remap	Stage 1 Tuning	120	9000.00	15000.00

### 6.8.1 Purpose

Service catalog entries offered by a company.

### 6.8.2 Important Columns

- `id`: Surrogate identifier.
- `company_id`: Offering company reference.
- `name`: Service name.
- `detail`: Service description.
- `approximate_duration`: Estimated time to complete.
- `min_cost` (numeric(10,2)): Minimum service cost.
- `max_cost` (numeric(15,2)): Maximum service cost.

### 6.8.3 Keys & Constraints

- PK: `id`
- FK: `company_id` → `company.id` (ON DELETE SET NULL)

## 6.9 appointment

	<code>id</code>	<code>vehicle_id</code>	<code>details</code>	<code>appointment_date</code>	<code>status</code>	<code>company_id</code>
1	4	7	Regular Service Checkup #4	2025-11-02 06:39:11.441300	accepted	
2	5	3	Regular Service Checkup #5	2025-11-09 16:56:13.691379	accepted	
3	6	4	Regular Service Checkup #6	2025-11-21 12:02:14.575321	accepted	
4	7	7	Regular Service Checkup #7	2025-11-07 17:43:37.007861	accepted	
5	8	6	Regular Service Checkup #8	2025-11-09 17:29:04.575759	accepted	
6	9	6	Regular Service Checkup #9	2025-11-21 09:15:23.366415	accepted	
7	10	7	Regular Service Checkup #10	2025-11-08 20:33:36.411441	accepted	
8	12	1	Regular Service Checkup #12	2025-11-25 01:44:17.334287	accepted	
9	13	5	Regular Service Checkup #13	2025-11-07 15:22:44.131702	accepted	
10	14	1	Regular Service Checkup #14	2025-11-21 00:24:52.743512	accepted	

### 6.9.1 Purpose

Booked service appointments for specific vehicles and companies.

### 6.9.2 Important Columns

- `id`: Surrogate identifier.
- `vehicle_id`: Vehicle reference.
- `company_id`: Company reference.
- `details`: Appointment details/notes.
- `appointment_date` (timestamp): Scheduled date and time.
- `status` (default 'pending'): Appointment status.

### 6.9.3 Keys & Constraints

- PK: `id`
- FK: `vehicle_id` → `vehicle.id`
- FK: `company_id` → `company.id`

## 6.10 Service Record table

	service_id	appointment_id	lead_mechanic_id	type	cost	duration	received_date
1	2	1	8	Standard Repair	500	60	2025-11-18
2	1	2	6	Standard Repair	1200	45	2025-11-21
3	3	3	1	Standard Repair	800	60	2025-11-01
4	1	4	6	Standard Repair	1200	45	2025-11-02
5	1	5	3	Standard Repair	1200	45	2025-11-09
6	3	6	7	Standard Repair	800	60	2025-11-21
7	2	7	3	Standard Repair	500	60	2025-11-07
8	3	8	7	Standard Repair	800	60	2025-11-09
9	3	9	1	Standard Repair	800	60	2025-11-21
10	3	10	10	Standard Repair	800	60	2025-11-08

### 6.10.1 Purpose

Historical records of performed services.

### 6.10.2 Important Columns

- **service\_id**: Service reference.
- **appointment\_id**: Appointment reference.
- **lead\_mechanic\_id**: Responsible mechanic reference.
- **type**: Service type/category.
- **cost**: Final service cost.
- **duration**: Actual time taken.
- **received\_date**: Date service was received.
- **expected\_finish\_date**: Projected completion date.
- **status** (default 'in progress'): Current service status.

### 6.10.3 Keys & Constraints

- PK: composite (**service\_id**, **appointment\_id**)
- FK: **service\_id** → **service.id**
- FK: **appointment\_id** → **appointment.id**
- FK: **lead\_mechanic\_id** → **employee.id**

## 6.11 Spares table

	<code>id</code>	<code>company_id</code>	<code>part_number</code>	<code>name</code>	<code>description</code>	<code>purchase_cost</code>	<code>retail_</code>
1	2		1 FIL-001	Oil Filter	Standard	100.00	
2	3		2 TIRE-MIC	Michelin Pilot	245/45/R18	4000.00	
3	1		1 OIL-001	Synthetic Oil 5W-30	4L Bottle	800.00	
4	4		1 FIL-OIL-001	Oil Filter	Standard Oil Filter	100.00	
5	5		1 OIL-SYN-5W30	Synthetic Oil	5W30 Full Synthetic 4L	800.00	
6	6		2 BAT-12V-45	Car Battery	12V 45Ah Maintenance F...	1500.00	
7	7		2 EV-CHG-CBL	Type 2 Cable	EV Charging Cable	3000.00	
8	8		3 TIR-265-65	Offroad Tire	265/65 R17 All Terrain	3500.00	
9	9		4 BRK-PAD-F	Brake Pads Front	Ceramic Brake Pads	800.00	
10	10		5 TURBO-KIT	Turbocharger	Stage 1 Turbo Kit	15000.00	

### 6.11.1 Purpose

Spare parts inventory and stock details per company.

### 6.11.2 Important Columns

- `id`: Surrogate identifier.
- `company_id`: Owning company reference.
- `part_number`: Unique part number.
- `name`: Spare part name.
- `description`: Spare part description.
- `purchase_cost`: Cost at purchase.
- `retail_price`: Selling price.
- `quantity` (default 1): Stock quantity.
- `expired_date`: Expiration date.
- `reorder_level`: Minimum stock before reorder.
- `location`: Storage location.
- `purchased_date` (default CURRENT\_DATE): Date purchased.

### 6.11.3 Keys & Constraints

- PK: `id`
- FK: `company_id` → `company.id` (ON DELETE CASCADE)

## 6.12 Membership Plan table

	membership_plan_id	company_id	name	detail	rank	price	duration
1	1	1	Bronze Care	5% Labor Discount	1	499.00	
2	2	1	Gold Premium	Priority Queue + 10% Discount	2	2500.00	
3	3	2	Street Racer	Discount on Tires & Oil	1	1200.00	
4	4	2	Track Pro	Free Alignment + 15% Parts	2	3500.00	
5	5	3	Elite Access	Valet Service & Free Coffee	1	2000.00	
6	6	3	Black Card	All Access + Concierge	2	50000.00	
7	27	1	Bronze Care	5% Labor Discount	1	500.00	
8	28	1	Silver Care	10% Labor Disc	2	1500.00	
9	29	2	Street Racer	Tire Discount	1	800.00	
10	30	2	Track Pro	Alignment Free	2	2500.00	

### 6.12.1 Purpose

Tiered membership plan definitions for companies.

### 6.12.2 Important Columns

- `membership_plan_id`: Surrogate identifier.
- `company_id`: Owning company reference.
- `name`: Plan name.
- `detail`: Plan description.
- `rank`: Plan tier/rank.
- `price`: Plan price.
- `duration`: Validity duration in days.
- `is_active`: Active status flag.

### 6.12.3 Keys & Constraints

- PK: `membership_plan_id`
- FK: `company_id` → `company.id` (ON DELETE CASCADE)

## 6.13 Membership table

	customer_id	membership_plan_id	purchased_date	expired_date
1		1	4 2025-11-29	2026-11-29
2		3	3 2025-11-29	2026-05-28
3		5	5 2025-11-29	2025-12-29
4		6	3 2025-11-29	2026-05-28
5		7	2 2025-11-29	2026-11-29
6		8	3 2025-11-29	2026-05-28
7		9	5 2025-11-29	2025-12-29
8		10	1 2025-11-29	2025-12-29
9		12	5 2025-11-29	2025-12-29
10		13	5 2025-11-29	2025-12-29

### 6.13.1 Purpose

Active membership instances linking customers to plans.

### 6.13.2 Important Columns

- `customer_id`: Customer reference.
- `membership_plan_id`: Membership plan reference.
- `purchased_date` (default CURRENT\_DATE): Date of purchase.
- `expired_date`: Expiration date

### 6.13.3 Keys & Constraints

- PK: composite (`customer_id, membership_plan_id`)
- FK: `customer_id` → `customer.id`
- FK: `membership_plan_id` → `membership_plan.membership_plan_id`

## 6.14 Membership Transaction table

	<code>id</code>	<code>customer_id</code>	<code>amount</code>	<code>currency</code>	<code>payment_method</code>	<code>reference_id</code>	<code>description</code>
1	1	1	500.00	THB	Cash	REC-001	Monthly Fee Payme
2	3	3	2500.00	THB	Cash	REC-001	Monthly Fee Payme
3	4	1	3500.00	THB	Credit Card	REF-TEST-1	New Member Signup
4	5	3	1200.00	THB	Credit Card	REF-TEST-3	New Member Signup
5	6	5	2000.00	THB	Credit Card	REF-TEST-5	New Member Signup
6	7	6	1200.00	THB	Credit Card	REF-TEST-6	New Member Signup
7	8	7	2500.00	THB	Credit Card	REF-TEST-7	New Member Signup
8	9	8	1200.00	THB	Credit Card	REF-TEST-8	New Member Signup
9	10	9	2000.00	THB	Credit Card	REF-TEST-9	New Member Signup
10	11	10	499.00	THB	Credit Card	REF-TEST-10	New Member Signup

### 6.14.1 Purpose

Financial entries for customer membership purchases and renewals.

### 6.14.2 Important Columns

- `id`: Surrogate identifier.
- `customer_id`: Customer reference.
- `amount`: Transaction amount.
- `currency` (default 'THB'): Transaction currency.
- `payment_method`: Payment method used.
- `reference_id`: External payment reference.
- `description`: Transaction description.
- `purchased_date` (default): Date of purchase.
- `expired_date`: Membership expiration date.

### 6.14.3 Keys & Constraints

- PK: `id`
- FK: `customer_id` → `customer.id` (ON DELETE CASCADE)

## 6.15 Subscription Plan table

	subscription_id	name	detail	rank	price	duration
1	1	Starter	Basic features for small garages (up to 50 jobs/mo)	1	4995.00	
2	3	Enterprise	All features + 24/7 Priority Support + API Access	3	125000.00	
3	2	Professional	Unlimited jobs + Analytics Dashboard	2	12495.00	

### 6.15.1 Purpose

Platform-level SaaS subscription plans for companies.

### 6.15.2 Important Columns

- **subscription\_id**: Surrogate identifier.
  - **name**: Plan name.
  - **detail**: Plan description.
  - **rank**: Plan tier/rank.
  - **price**: Plan price.
  - **duration**: Validity duration in days (default 30 days).
- `subscription_id, name, detail, rank, price, duration` (default 30 days).

### 6.15.3 Keys & Constraints

- PK: `subscription_id`

## 6.16 Subscription Transaction table

	transaction_id	subscription_id	owner_id	amount	currency	payment_method	reference_id
1		1	3	1	25000.00 THB	Bank Transfer	REF-P
2		2	2	2	2499.00 THB	PromptPay	REF-P
3		3	3	3	25000.00 THB	Credit Card	REF-P
4		4	2	4	2499.00 THB	Credit Card	REF-P
5		5	1	5	999.00 THB	PromptPay	REF-P
6		6	1	6	999.00 THB	PromptPay	REF-P
7		7	2	7	2499.00 THB	Bank Transfer	REF-P
8		8	1	8	999.00 THB	PromptPay	REF-P
9		9	3	9	25000.00 THB	Credit Card	REF-P
10		10	1	10	999.00 THB	Bank Transfer	REF-P

### 6.16.1 Purpose

Payments for company subscriptions to platform plans.

### 6.16.2 Important Columns

- `transaction_id`: Surrogate identifier.
- `subscription_id`: Subscription plan reference.
- `owner_id`: Paying owner reference.
- `amount`: Transaction amount.
- `currency` (default 'THB'): Transaction currency.
- `payment_method`: Payment method used.
- `reference_id`: External payment reference.
- `description`: Transaction description.
- `purchased_date` (default): Date of purchase.
- `expired_date`: Subscription expiration date.

### 6.16.3 Keys & Constraints

- PK: `transaction_id`
- FK: `subscription_id` → `subscription_plan.subscription_id`
- FK: `owner_id` → `owner.id`

## 6.17 Transaction table

	transaction_id	appointment_id	service_id	amount	currency	payment_method	r
1	1	1	1	1038.31	THB	Credit Card	TXN-
2	2	2	1	1060.83	THB	Credit Card	TXN-
3	3	3	3	2030.51	THB	Credit Card	TXN-
4	4	4	1	1348.34	THB	Credit Card	TXN-
5	5	5	1	837.74	THB	Credit Card	TXN-
6	6	6	3	1586.65	THB	Credit Card	TXN-
7	7	7	1	1298.84	THB	Credit Card	TXN-
8	8	8	3	1539.98	THB	Credit Card	TXN-
9	9	9	3	2419.22	THB	Credit Card	TXN-
10	10	10	3	798.91	THB	Credit Card	TXN-

### 6.17.1 Purpose

General financial transactions for services & appointments.

### 6.17.2 Important Columns

- `transaction_id`: Surrogate identifier.
- `appointment_id`: Related appointment reference.
- `service_id`: Related service reference.
- `amount`: Transaction amount.
- `currency` (default 'THB'): Transaction currency.
- `payment_method`: Payment method used.
- `reference_id`: External payment reference.
- `description`: Transaction description.
- `created_at` (default CURRENT\_TIMESTAMP): Timestamp of transaction creation.

### 6.17.3 Keys & Constraints

- PK: `transaction_id`
- FK: `appointment_id` → `appointment.id` (ON DELETE SET NULL)
- FK: `service_id` → `service.id` (ON DELETE SET NULL)

## 6.18 Request Form table

	request_id	company_id	first_name	middle_name	last_name	address	city
1	3	1	David	Alan	Wilson	12 Silom Soi 4	Bangkok
2	4	1	Somsak	<null>	Rakthai	88 Ladprao Rd	Bangkok
3	5	1	Mary	<null>	Johnson	55 Sathorn Tai	Bangkok
4	6	2	Kenji	<null>	Tanaka	99 Thong Lor	Bangkok
5	7	2	Wichai	<null>	Srisuk	101 Phetchaburi Rd	Bangkok
6	8	2	Sarah	Jane	Connor	23 Asoke Montri	Bangkok
7	9	3	Nopadol	<null>	Kaewmanee	77 Rama 9	Bangkok
8	10	3	James	<null>	Bond	007 Wireless Rd	Bangkok
9	11	3	Ploy	<null>	Pailin	456 Ratchada	Bangkok

### 6.18.1 Purpose

Incoming requests/applications associated with a company.

### 6.18.2 Important Columns

- **request\_id**: Surrogate identifier.
- **company\_id**: Company reference.
- **first\_name**: Requester's first name.
- **middle\_name**: Requester's middle name.
- **last\_name**: Requester's last name.
- **address**: Requester's address.
- **city**: Requester's city.
- **postal\_code**: Requester's postal code.
- **email**: Requester's email address.
- **telephone\_no**: Requester's telephone number.
- **specialize**: Requester's specialization or area of interest.

### 6.18.3 Keys & Constraints

- PK: **request\_id**
- FK: **company\_id** → **company.id** (ON DELETE CASCADE)

## 7 Database Design Quality, Security, and efficiency

### 7.1 Normalization Design

The schema follows proper normalization to reduce redundancy and anomalies while keeping queries practical.

- **1NF:** All columns store atomic values (e.g., single email/telephone per user row; no repeating groups).
- **2NF:** Non-key attributes depend on the whole key in composite PK tables (e.g., `service_record` depends on both `service_id` and `appointment_id`).
- **3NF:** Non-key attributes do not transitively depend on keys (e.g., company city/postal code are attributes of the company, not derived from other non-keys).
- **Reference Integrity:** Foreign keys enforce valid relationships (e.g., vehicle `owner_id` → customer, membership → plan/customer).
- **Controlled Denormalization (where justified):** Read-heavy listing views include joined names and labels at query-time rather than stored duplicates, preserving normalization in base tables.

### 7.2 Security Mindset & Implementation

Security is treated as a first-class design constraint across data, access, and operations.

- **Least Privilege:** Separate roles for application, admin, and CI with minimal permissions; no superuser credentials in app runtime.
- **Password Storage:** Use `crypt` (`bcrypt`) with per-user salt; never store plaintext or reversible encryption.
- **Authentication:** Dedicated functions for login that return minimal user info on success; avoid exposing sensitive fields.
- **Input Validation:** Constrain data via types, CHECKs, and FK references; reject invalid states early.
- **Auditability:** Transaction and membership history retained to support investigations and dispute resolution.

### 7.3 Dummy Database

Improvement proposal: introduce a dedicated dummy (non-production) database with least-privilege roles so day-to-day workers and app processes cannot perform any DDL (DROP/ALTER/CREATE). This avoids accidental destructive changes while enabling safe development/testing using synthetic data. The dummy environment mirrors the schema but contains only synthetic or anonymized records.

- **Purpose:** Enable safe iteration, demos, and automated tests without risking real user or financial data.
- **Data Strategy:** Populate with generated data that respects constraints (FKs, uniques) and realistic distributions; never reuse production credentials or PII.
- **Isolation:** Host on a separate instance/DB name with distinct users and roles; restrict network access to development IPs only.
- **DDL Restrictions:** Revoke CREATE/ALTER/DROP on schemas/tables for worker roles; grant only necessary DML (SELECT/INSERT/UPDATE/DELETE). This ensures

workers cannot drop any tables.

- **Role Design:** Use separate roles, e.g., read-only (`app_ro`), read-write (`app_rw`) without DDL, and admin (`app_admin`) restricted to maintainers.

#### Example (PostgreSQL least-privilege setup)

```

1 -- Lock down default schema DDL for everyone
2 REVOKE CREATE ON SCHEMA public FROM PUBLIC;
3
4 -- Roles without DDL
5 CREATE ROLE app_ro  NOINHERIT NOCREATEDB NOCREATEROLE NOSUPERUSER;
6 CREATE ROLE app_rw  NOINHERIT NOCREATEDB NOCREATEROLE NOSUPERUSER;
7
8 -- Optional admin role for maintainers only
9 CREATE ROLE app_admin NOINHERIT NOCREATEDB NOCREATEROLE NOSUPERUSER;
10
11 -- Grant minimal privileges
12 GRANT USAGE ON SCHEMA public TO app_ro, app_rw;
13 GRANT SELECT ON ALL TABLES IN SCHEMA public TO app_ro;
14 GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO
   app_rw;
15
16 -- Ensure future tables are also protected
17 ALTER DEFAULT PRIVILEGES IN SCHEMA public
   GRANT SELECT ON TABLES TO app_ro;
18 ALTER DEFAULT PRIVILEGES IN SCHEMA public
   GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO app_rw;
19
20 -- Do NOT grant CREATE/ALTER/DROP to app_ro/app_rw (prevents dropping
   tables)

```

## 7.4 Query Efficiency Implementation

Queries are structured for clarity and performance with appropriate indexing and access patterns.

- **Primary/Foreign Keys:** Implicit btree indexes on PKs and recommended indexes on high-selectivity FKS (e.g., `appointment.company_id`, `membership.membership_plan_id`).
- **Lookup Indexes:** Create btree indexes on frequent filters and joins (e.g., `vehicle.owner_id`, `employee.company_id`, `service.company_id`).
- **Search Fields:** Index unique identifiers used for authentication and CRUD (`username`, `email`, `vin`, `license_plate`).
- **Time-Ordered Access:** For reporting, order by date/time with supporting indexes (e.g., `financial_record.record_date`, `membership_transaction.purchased_date`).
- **Selective Projections:** Views/functions project only needed columns to reduce I/O and transfer.

## 8 Financial Considerations

The total operational cost for deploying this comprehensive Garage Management Platform is projected at approximately 4,999 THB per month.

This all-inclusive subscription fee grants full access to the platform's suite of enterprise modules, including Customer Relationship Management (CRM), Inventory and Stock Tracking, and Financial Reporting.

Beyond the software licensing, this cost covers all essential infrastructure requirements, such as secure cloud database hosting, daily data backups, and server maintenance.

## 9 Database access

- Hostname: dpg-d4hvij0gjchc73di0oqg-a

- Port: 5432

- Database: sc\_db\_project

- Username: sc\_db\_project\_user

- Password: jOtTexUwkUPGFP3E3rvevTmixbtYvliZ

- Internal Database URL:

postgresql://sc\_db\_project\_user:jOtTexUwkUPGFP3E3rvevTmixbtYvliZ@dpg-d4hvij0gjchc73di0oqg-a/sc\_db\_project

- External Database URL:

postgresql://sc\_db\_project\_user:jOtTexUwkUPGFP3E3rvevTmixbtYvliZ@dpg-d4hvij0gjchc73di0oqg-a.singapore-postgres.render.com/sc\_db\_project

- PSQL Command:

```
PGPASSWORD=jOtTexUwkUPGFP3E3rvevTmixbtYvliZ psql -h dpg-d4hvij0gjchc73di0oqg-a.singapore-postgres.render.com -U sc_db_project_user sc_db_project
```

## 10 Github Repository

You can access the repository at the following URL: <https://github.com/Tawancs/cs-db-project>