# Air-Protech Cooling Services: Comprehensive Service Booking & Inventory Management System

## 🔹 Pages & Functionalities

### 👤 User (Customer) Pages

**1️⃣ Home Page**

- Overview of services & products.
- Contact details and business location.
- **(LOGIN functionality available here).**

**2️⃣ Service Booking Page**

- Request repairs, installations, or maintenance.
- Booking priority order
- Select service type, preferred schedule, and provide details.
- View real-time status of the service request.

**3️⃣ Air Conditioning Products Page**

- Display aircon units, spare parts, and accessories.
- Track product order status (Processing → Shipped → Delivered).

**4️⃣ My Orders & Service Requests**

- View past orders and service history.

- Track the status of ongoing service requests.

## 🔧 Technician Pages

### 5️⃣ Assigned Jobs Page

- View work orders assigned by the admin.
- Update job status (Pending → In Progress → Completed).
- Upload service report after completing the job.

### 6️⃣ Job Location & Navigation

- View assigned job location on a map. – TEXT FORMAED
- Mark job as “On the Way” or “Delayed” for admin tracking. – ON THE WAY PA OR MA DELAY

## 🛠️ Admin Pages

### 7️⃣ Dashboard

- Overview of service requests, sales, and technician assignments.
- Daily, weekly, and monthly reports.

### 8️⃣ Customer Management

- View and manage customer profiles. // ga libog ko ari nga part
- Track service history and orders. //diri pod

### 9️⃣ Work Order Management

- View all service requests.
- Assign jobs to available technicians.
- Track job progress and completion.

### 🔟 Product & Inventory Management

- Add, update, or remove aircon products & accessories.
- Track stock levels and set restock alerts.

### 1️⃣2️⃣ Reports & Analytics

- Generate revenue reports based on completed service bookings and product sales.
- Track service request trends and customer demand.
- Monitor technician performance and job completion efficiency.
- Analyze inventory movement and restocking needs.
    - 

---

## 🔹 Tech Stack Recommendation

**A Python Web Framework** (e.g., **Django**, **Flask**, or **FastAPI**)

**Vue.js** for the front end

**PostgreSQL** for the database

---

Would you like **mockups** or a **basic database schema** to get started? 🚀

YES Kent — 100% you **can and should establish that relationship in an RDBMS**. What you're describing is exactly what relational databases are meant to do — managing **logical, controlled workflows** through well-defined relationships.

Let me walk you through the **real-world flow** you just described and how it maps into **relationships** and **foreign keys** in an RDBMS (like MySQL, PostgreSQL, etc.):

---

# 💡 WORKFLOW: Step by Step + Relationships

## 🔹 Step 1: Customer Sends a Service Request

- Table: `SERVICE_REQUEST`
- Foreign Key: `user_id` → `USER`

✅ **Relationship:**
**1 USER → many SERVICE_REQUESTS** (1:M)

---

## 🔹 Step 2: Admin Receives & Takes Charge

- Table: `SERVICE_REQUEST_ASSIGNMENT`
- Foreign Keys:
  - `sr_id` → `SERVICE_REQUEST`
  - `admin_id` → `ADMIN`

✅ **Relationship:**
**M:N between ADMIN and SERVICE_REQUEST** (via `SERVICE_REQUEST_ASSIGNMENT`)

---

## 🔹 Step 3: Admin Checks for Available Technician

- Table: `TECHNICIAN` + a helper like `TECHNICIAN_AVAILABILITY` or status column like `is_available`
- Filter logic in app: only show `TECHNICIAN` where `is_available = true`

✅ **No relationship needed**, just selection logic/query

---

### 🔹 Step 4: Admin Assigns Work Order to Technician

- Table: `WORK_ORDER`
- Foreign Keys:
  - `sr_id` → `SERVICE_REQUEST`
  - `tech_id` → `TECHNICIAN`
  - `admin_id` → `ADMIN` *(optional for audit)*

✅ **Relationships:**

- **1 SERVICE_REQUEST → 1 WORK_ORDER** (1:1)
- **1 TECHNICIAN → many WORK_ORDERS** (1:M)
- **1 ADMIN → many WORK_ORDERS** (1:M, optional)

---

## ✅ Summary of Relationships in RDBMS Terms

| From | To | Type |
|---|---|---|
| `USER` | `SERVICE_REQUEST` | 1:M |

| | | |
|---|---|---|
| `ADMIN` | `SERVICE_REQUEST_ASSIGNMENT` | 1:M |
| `SERVICE_REQUEST` | `SERVICE_REQUEST_ASSIGNMENT` | 1:M |
| `SERVICE_REQUEST` | `WORK_ORDER` | 1:1 |
| `TECHNICIAN` | `WORK_ORDER` | 1:M |
| `ADMIN` | `WORK_ORDER` | 1:M (optional) |

---

## ✅ Yes, You Can Enforce All This in RDBMS:

- Using **foreign keys**
- Using **composite keys or unique constraints** if needed
- Using **Junction tables (like `SERVICE_REQUEST_ASSIGNMENT`)** for M:N
- And application-side logic for technician availability

---

Let me know if you want the **SQL table relationships** (FK constraints), or if you want to **add triggers or status flags** to prevent assigning unavailable techs. You're basically building a clean and logical workflow here — great thinking 💪

## Air-Protech Cooling Services: Entity-Relationship Diagram (ERD)

# Core Entities

- **User**: Customers who can book services and purchase products
- **Admin**: Staff members with system administration privileges (includes regular staff and main admins)
- **Technician**: Service personnel who perform field work and installations
- **ServiceRequest**: Customer requests for repairs, maintenance, or installations
- **Product**: Air conditioning units, parts, and accessories available for sale
- **Order**: Customer purchases of products
- **Inventory**: Current stock levels of all products

# Entity Relationships

## User Relationships

1. **User** → **ServiceRequest** (1) d
    - One user can submit many service requests
    - Each service request belongs to exactly one user
2. **User** → **Order** (1)d

    - One user can place many product orders
    - Each order belongs to exactly one user
3. **User → UserProfile** (1:1) D
    - Each user has exactly one profile
    - Each profile belongs to exactly one user

## Admin Relationships

4. **Admin → ServiceRequest** (M via ServiceRequestAssignment) D
    - One admin can manage many service requests
    - One service request can be handled by different admins over time
5. **Admin → CustomerAccessLog** (1) D
    - One admin can generate many customer access log entries
    - Each log entry is created by exactly one admin
6. **Admin → OrderAccessLog** (1)D
    - One admin can generate many order access log entries
    - Each log entry is created by exactly one admin
7. **Admin → ServiceRequestAccessLog** (1)D
    - One admin can generate many service request access logs
    - Each log entry is created by exactly one admin
8. **Admin → InventoryTransaction** (1) D
    - One admin can perform many inventory transactions
    - Each inventory transaction is performed by exactly one admin

## Technician Relationships

9. **Technician → ServiceRequest** (M via JobAssignment) D
    - One technician can be assigned to many service requests
    - One service request can be assigned to multiple technicians (e.g., complex jobs)
10. **Technician → ServiceReport** (1)D
    - One technician can create many service reports
    - Each service report is created by exactly one technician

## Service Request Relationships

11. **ServiceRequest → ServiceReport** (1:1)D
    - Each service request has exactly one final service report
    - Each service report belongs to exactly one service request
12. **ServiceRequest → ServiceRequestStatusUpdate** (1) D
    - One service request can have many status updates
    - Each status update belongs to exactly one service request
13. **ServiceRequest → ServiceRequestAccessLog** (1)
    - One service request can appear in many access logs
    - Each log entry refers to exactly one service request

### Product and Inventory Relationships

14. **Product** → **Inventory** (1:1) D
    - Each product has exactly one inventory record
    - Each inventory record tracks exactly one product
15. **Product** → **Order** (M via OrderItem) D
    - One product can appear in many orders
    - One order can contain many products
16. **Product** → **InventoryTransaction** (1)
    - One product can have many inventory transactions
    - Each inventory transaction affects exactly one product

### Order Relationships

17. **Order** → **OrderStatusUpdate** (1)
    - One order can have many status updates
    - Each status update belongs to exactly one order
18. **Order** → **OrderAccessLog** (1)
    - One order can appear in many access logs
    - Each log entry refers to exactly one order

## Tracking and Logging Relationships

19. **User** → **CustomerAccessLog** (1) D
    - One user can appear in many customer access log entries
    - Each log entry refers to exactly one user
20. **Admin** → **ActivityLog** (1) D
    - One admin can generate many activity log entries
    - Each activity log entry is created by exactly one admin
21. **Technician** → **LocationUpdate** (1)
    - One technician can have many location updates
    - Each location update belongs to exactly one technician

## Additional Important Relationships

22. **ServiceRequest** → **Product** (M [via ServiceRequestProduct]) D
    - One service request can involve many products (repairs, installations)
    - One product can be involved in many service requests
23. **Technician** → **SkillSet** (M [via TechnicianSkill])
    - One technician can have many skills
    - One skill can be possessed by many technicians