Database
This document has important information related to the database content

Refit
Project Description:
The proposed web application will be developed using the Python based Flask micro-framework, which is known for its simplicity and flexibility. The app's codebase will be stored in a Git repository to facilitate version control and collaboration among developers. The app will be deployed to DigitalOcean using a continuous integration and continuous deployment (CI/CD) process, which automates the testing, building, and deployment of the app to ensure that the app is always up-to-date and functional.
The main function of the app is to provide a form on the frontend where users or administrators can input report information, which will then be processed using Python to calculate scores and generate a web-based and downloadable PDF version of the report card. This will enable users to easily access and share their report card with others.
Additionally, the app will have different functionalities for individual users, the Med-I-Well administrator, and clients, who will have access to a dashboard. This will allow each user to have a personalized experience and access the features that are relevant to them.
User Account:

Admin Blueprint
The provided code creates an 'admin' blueprint in a Flask web application.
1. The necessary Flask modules and a custom `models` module are imported.
2. The 'admin' blueprint is created using Flask's `Blueprint` function.
3. A route is defined at "/admin" that responds to HTTP GET requests.
4. When the "/admin" route is accessed, the `user_page` function fetches all `User` records from the database.
5. The `user_page` function then renders the 'admin/admin.html' template, passing the fetched User data to it.
Auth Blueprint
The provided code creates an 'auth' blueprint in a Flask web application, handling user authentication and registration.
1. The required Flask modules and a custom `models` module are imported.
2. An 'auth' blueprint is created using Flask's `Blueprint` function.
3. Four routes are defined:
- The "/" and "/login" routes: These handle both GET and POST requests. When a POST request is sent with a username and password, the application checks if these credentials are correct. If they are, the user is redirected to the home page. Otherwise, an error message is printed.
- The "/forgot_password" route: This handles both GET and POST requests. When a POST request is sent with an email and a new password, the application searches for a user with the given email. If a user is found, the password is updated.
- The "/home" route: This handles GET requests and simply renders the home page.
- The "/register" route: This handles both GET and POST requests. When a POST request is sent with registration details, the application creates a new user with these details and redirects the user to a page based on their user type.
All the routes, except "/home", render a template when a GET request is sent. These templates contain forms where users can enter their details.
Contractor Blueprint
The provided code creates a 'contractor' blueprint in a Flask web application, handling contractor-specific functionalities.
1. The required Flask modules and a custom `models` module are imported.
2. A 'contractor' blueprint is created using Flask's `Blueprint` function.
3. Four routes are defined:
- The "/formcontractor/" route: This handles both GET and POST requests. When a POST request is sent with contractor details, the application creates a new contractor and contractor detail record in the database with these details. Then, it redirects the user to the contractor dashboard. When a GET request is sent, it renders the "contractor/formcontractor.html" template.
- The "/dashboardcontractor" route: This handles GET requests and renders the contractor dashboard via the "contractor/dashboardcontractor.html" template.
- The "/handle_qr_code" route: This handles POST requests. When a POST request is sent with a JSON object containing a QR code, the application prints the QR code and returns a success message.
This blueprint essentially allows the application to handle contractor registration, display the contractor dashboard, and process QR codes.
Technician Blueprint
The provided code creates a 'technician' blueprint in a Flask web application, handling technician-specific functionalities.
1. The required Flask modules and a custom `models` module are imported.
2. A 'technician' blueprint is created using Flask's `Blueprint` function.
3. The following routes are defined:
- The "/formtechnician/" route: This handles both GET and POST requests. When a POST request is sent with technician details, the application creates a new user detail record in the database with these details and then redirects the user to the technician dashboard. When a GET request is sent, it renders the "technician/formtechnician.html" template.

- The "/dashboardtechnician" route: This handles GET requests and renders the technician dashboard via the "technician/dashboardtechnician.html" template.
- The "/equipment/equipment_create" route: This handles GET requests and renders the equipment creation page via the "equipment/equipment_create.html" template.
- The "/equipment/repair" route: This handles GET requests and renders the repair page via the "equipment/repair.html" template.
- The "/equipment/recovery" route: This handles GET requests and renders the recovery page via the "equipment/recovery.html" template.

This blueprint essentially allows the application to handle technician registration, display the technician dashboard, and manage equipment-related activities such as creation, repair, and recovery.

Wholesaler Blueprint

The provided code creates a 'wholesaler' blueprint in a Flask web application, handling wholesaler-specific functionalities.

1. The required Flask modules are imported.
2. A 'wholesaler' blueprint is created using Flask's `Blueprint` function.
3. The following routes are defined:

- The "/formwholesaler" route: This handles both GET and POST requests. When a POST request is sent with wholesaler details, these details are retrieved from the form data and then flashed as a success message. The user is then redirected to the wholesaler dashboard. When a GET request is sent, it renders the "wholesaler/formwholesaler.html" template.
- The "/dashboardwholesaler" route: This handles GET requests and renders the wholesaler dashboard via the "wholesaler/dashboardwholesaler.html" template.

This blueprint essentially allows the application to handle wholesaler registration and display the wholesaler dashboard.

Contractor Login and Registration Page
The "Contractor Login and Registration Page" is a web page designed for contractors to access their accounts through login or register for a new account. When a contractor selects the "Contractor" role, they will be directed to this page, where they have two options: "Log In" or "Register."
Log In
If the contractor selects "Log In," they will be prompted to enter their username and password to access their existing account.
Register
If the contractor selects "Register," they will be directed to the "Registration Form." Here, they must provide their name, email ID, password, and confirm the password. Additionally, they need to enter their company name, branch ID, address (including street address, city, province, postal code), and phone number to create a new account.
Contractor Login and Registration Page
Welcome, Contractor! Please choose an option below:
- [ ] Log In
- [ ] Register
[Select](#)

Contractor Login Form
Log In
Please enter your credentials to access your account:
- Username: [Enter Username]
- Password: [Enter Password]

Contractor Registration Form
Register
Please provide the following details to create a new account:
- Name: [Enter Full Name]
- Email ID: [Enter Email Address]
- Password: [Enter Password]
- Confirm Password: [Confirm Password]
- Company Name: [Enter Company Name]
- Branch ID: [Enter Branch ID]
- Address:
- Street: [Enter Street Address]
- City: [Enter City]
- Province: [Enter Province]
- Postal Code: [Enter Postal Code]
- Phone Number: [Enter Phone Number]

Cylinder Type Selection Form
The "Cylinder Type Selection Form" is a web form that allows users to choose between two options: "New Cylinder" or "Recovery Cylinder." Upon selection, the form dynamically redirects the user to the corresponding page for the chosen option.
New Cylinder Option
If the user selects "New Cylinder," they will be directed to the "New Cylinder Form." This form displays auto-populated information such as cylinder tag ID, refrigerant ID, technician ID, and the select date (which defaults to the current date but can be edited). The user can then enter details such as refrigerant type, cylinder size in lbs, refrigerant weight in lbs, purchase date (if different from the manufacturer date), and wholesaler.
Recovery Cylinder Option
If the user selects "Recovery Cylinder," they will be directed to the "Recovery Cylinder Form." This form displays non-editable details such as cylinder tag ID, company name, refrigerant ID, technician ID, and the recovery date (which defaults to the current date but can be edited). The user must then enter details such as refrigerant type, cylinder type (with options: clean/reuse, non-usable, and burnout), cylinder size in lbs, and refrigerant type in lbs.
Cylinder Type Selection Form
Please select a cylinder type:
- [ ] New Cylinder
- [ ] Recovery Cylinder
[Select](#)

New Cylinder Form
Auto Populated Information
- Cylinder Tag ID: [Cylinder Tag ID]
- Refrigerant ID: [Refrigerant ID]
- Technician ID: [Technician ID]
- Select Date: [Current Date]
New Cylinder Details
- Refrigerant Type: [Enter Refrigerant Type]
- Cylinder Size (lbs): [Enter Cylinder Size]
- Refrigerant Weight (lbs): [Enter Refrigerant Weight]
- Purchase Date (if different): [Enter Purchase Date]
- Wholesaler: [Enter Wholesaler]

Recovery Cylinder Form
Auto Populated Information
- Cylinder Tag ID: [Cylinder Tag ID]
- Company Name: [Company Name]
- Refrigerant ID: [Refrigerant ID]
- Technician ID: [Technician ID]
- Recovery Date: [Current Date]
Recovery Cylinder Details
- Refrigerant Type: [Enter Refrigerant Type]
- Cylinder Type:
- [ ] Clean/Reuse
- [ ] Non-Usable
- [ ] Burnout
- Cylinder Size (lbs): [Enter Cylinder Size]
- Refrigerant Type (lbs): [Enter Refrigerant Type]

cylinder recovery form
The "Cylinder Recovery Form" is a web form designed for technicians to document the recovery process of a refrigerant cylinder. The form allows technicians to view auto-populated information, such as the cylinder's tag ID, refrigerant ID, technician ID, cylinder type, size, and current date (which can be edited if necessary). Additionally, the form displays the current refrigerant weight in pounds, and it automatically calculates and displays the "Refrigerant Weight After Service" by adding the weight reclaimed by the technician.
The technician is required to enter the "Refrigerant Weight Reclaimed" in pounds. Upon submitting the form, it captures all the relevant data for record-keeping.
Cylinder Recovery Form
Auto Populated Information
- Cylinder Tag ID: CYL001
- Refrigerant ID: REF001
- Technician ID: TECH001
- Cylinder Type: TypeX
- Cylinder Size (lbs): 50
- Create Date: [Current Date]
- Current Refrigerant Weight (lbs): 25
- Refrigerant Weight After Service (lbs): [Calculated]
Information to be Entered
- Refrigerant Weight Reclaimed (lbs): [Enter reclaimed weight]
cylinder repair form
The "Cylinder Repair Form" is a web form designed for technicians to document the repair process of a refrigerant cylinder. The form provides auto-populated information, such as the cylinder's tag ID, refrigerant ID, technician ID, cylinder type, size, and current date (editable). The technician can also view and edit the current refrigerant weight in pounds. Additionally, the form calculates and displays the "Refrigerant Weight After Service" by subtracting the refrigerant weight added to the equipment from the current weight.
The technician is required to enter the "Refrigerant Weight Added to Equipment" in pounds. The form also includes an option to select whether to "Add Cylinder" or not, with choices of "Yes" and "No."
The form ensures the technician can easily enter the necessary repair details, such as refrigerant weight added and whether a cylinder needs to be added after service, making the documentation process efficient and accurate.
Cylinder Repair Form
Auto Populated Information
- Cylinder Tag ID: CYL001
- Refrigerant ID: REF001
- Technician ID: TECH001
- Cylinder Type: TypeX
- Cylinder Size (lbs): 50
- Create Date: [Current Date]
- Current Refrigerant Weight (lbs): [Editable]
Information to be Entered
- Refrigerant Weight Added to Equipment (lbs): [Enter weight]
- Add Cylinder:
- [ ] Yes
- [ ] No

Equipment Create Form
The "Equipment Create Form" is a web form designed for technicians to create new equipment entries. The form displays auto-populated information, including profile info, equipment tag ID, technician ID, and the create date (editable, defaulted to the current date). The technician can also edit the address, and if the organization location exists in the system, the organization's ID will be displayed.
The form dynamically populates the refrigerant ID based on the selected refrigeration type. Additionally, it calculates and displays the "Total Refrigerant Charge," which is the sum of the factory charge and additional charge.
The technician can enter the manufacturer's name, model number, serial number, equipment type, and refrigerant type. They can also input the factory charge amount (lbs) and additional refrigerant charge. If the technician selects "Yes," they can add the amount of refrigerant, which automatically updates the "Total Charge." Additionally, the form includes fields for the location description and additional notes.
Equipment Create Form
Auto Populated Information
- Profile Info: [Profile Info]
- Equipment Tag ID: EQUIP001
- Technician ID: TECH001
- Create Date: [Current Date]
- Address: [Editable Address]
- Organization's ID: [Organization ID (if exists in the system)]
Refrigerant Information
- Refrigeration Type: [Refrigeration Type]
- Refrigerant ID: [Automatically populated based on Refrigeration Type]
- Total Refrigerant Charge (lbs): [Calculated as Factory Charge + Additional Charge]
Equipment Details
- Manufacturer's Name: [Enter Manufacturer's Name]
- Model Number: [Enter Model Number]
- Serial Number: [Enter Serial Number]
- Equipment Type: [Enter Equipment Type]
- Refrigerant Type: [Enter Refrigerant Type]
- Factory Charge Amount (lbs): [Enter Factory Charge Amount]
- Additional Refrigerant Charge (lbs): [Enter Additional Refrigerant Charge]
Add Refrigerant
- [ ] Yes
- [ ] No
- Amount of Refrigerant to Add (lbs): [Enter Amount]
- Location Description: [Enter Location Description]
Additional Notes
When the submit button is pressed, the form also sends gps co-ordinates to the database (for now it just prints into the consol as database needs to be updated), default it asks for user permission for location access, which needs to be added to the previous screen that was disucssed to allow for location access.

Equipment repair form
The "Repair Form" is a web form designed for technicians to document equipment repairs. The form displays auto-populated information, including the equipment tag ID, technician ID, and the repair date (editable, defaulted to the current date). It also provides details such as refrigerant ID, refrigeration type, factory charge amount of refrigerant (lbs), additional amount added (lbs), and the total refrigerant amount in the equipment (automatically calculated as the sum of factory charge and additional amount).
The form further includes fields for the equipment's model number, serial number, and ODS Sheet number. Technicians can enter test results using checkboxes with options such as repaired discharge, line on compressor, leak detected, leak repair, vacuum test performed, leak not repaired, no leak detected, no longer contains refrigerant, and compressor oil removed. Additionally, there's a checkbox labeled "Other," allowing technicians to provide specific details if none of the predefined options fit.
Upon filling out the form, the technician can click the "Submit" button. However, before submitting, the form prompts the technician to enter a unique job number, which helps identify and track the repair.
Repair Form
Auto Populated Information
- Equipment Tag ID: EQUIP001
- Technician ID: TECH001
- Repair Date: [Current Date]
- Refrigerant ID: REF001
- Refrigeration Type: [Refrigeration Type]
- Factory Charge Amount Refrigerant (lbs): 50
- Additional Amount Added (lbs): [Enter additional amount]
- Total Refrigerant Amount in Equipment (lbs): [Calculated]
- Model Number: [Enter model number]
- Serial Number: [Enter serial number]
- ODS Sheet Number: [Enter ODS sheet number]
Test Results
- [ ] Repaired Discharge
- [ ] Line on Compressor
- [ ] Leak Detected
- [ ] Leak Repair
- [ ] Vacuum Test Performed
- [ ] Leak Not Repaired
- [ ] No Leak Detected
- [ ] No Longer Contains Refrigerant
- [ ] Compressor Oil Removed
- [ ] Other: [Enter specific problem]
Job Number
- Job Number: [Enter job number]
Equipment Recovery form
The "Recovery Form" is a web form designed for technicians to document refrigerant recovery processes. The form displays auto-populated information, including the equipment tag ID, technician ID, and the repair date (editable, defaulted to the current date). It also provides details such as refrigerant ID, refrigeration type, factory charge amount of refrigerant (lbs), additional charge amount, and the total charge (automatically calculated as the sum of factory charge and additional amount).
The form further includes fields for the equipment's model number, serial number, and ODS Sheet number. Technicians can enter test results using checkboxes with options such as repaired discharge, line on compressor, leak detected, leak repair, vacuum test performed, leak not repaired, no leak detected, no longer contains refrigerant, and compressor oil removed. Additionally, there's a checkbox labeled "Other," allowing technicians to provide specific details if none of the predefined options fit.
After selecting "Reclaim Gas," the form will display the option to choose "Recovery Cylinder" or use a "Scan Button" to reclaim the cylinder. The form then prompts the technician to enter a unique job

number before submission, which helps identify and track the recovery process
Recovery Form
Auto Populated Information
- Equipment Tag ID: EQUIP001
- Technician ID: TECH001
- Repair Date: [Current Date]
- Refrigerant ID: REF001
- Refrigeration Type: [Refrigeration Type]
- Factory Charge Refrigerant (lbs): 50
- Additional Charge (lbs): [Enter additional charge]
- Total Charge (lbs): [Calculated]
- Model Number: [Enter model number]
- Serial Number: [Enter serial number]
- ODS Sheet Number: [Enter ODS sheet number]
Test Results
- [ ] Repaired Discharge
- [ ] Line on Compressor
- [ ] Leak Detected
- [ ] Leak Repair
- [ ] Vacuum Test Performed
- [ ] Leak Not Repaired
- [ ] No Leak Detected
- [ ] No Longer Contains Refrigerant
- [ ] Compressor Oil Removed
- [ ] Other: [Enter specific problem]
Reclaim Gas
- [ ] Reclaim Gas
- [ ] Recovery Cylinder
- [ ] Scan Button
Job Number
- Job Number: [Enter job number]

Organization and Wholesaler Login and Registration Page
The "Organization and Wholesaler Login and Registration Page" is a web page designed for users with roles such as "Organization" or "Wholesaler." Upon selecting the respective role, users will be directed to this page, where they have two options: "Log In" or "Register."
Log In
If the user selects "Log In," they will be prompted to enter their username and password to access their existing account.
Register
If the user selects "Register," they will be directed to the "Registration Form." Here, they must provide their name, email ID, password, and confirm the password. Additionally, they need to enter their company name, branch, mailing address (including street address, city, province, postal code), billing address (including street address, city, province, postal code), and phone number to create a new account.
Organization and Wholesaler Login and Registration Page
Welcome! Please choose an option below:
- [ ] Log In
- [ ] Register
[Select](#)

Organization and Wholesaler Login Form
Log In
Please enter your credentials to access your account:
- Username: [Enter Username]
- Password: [Enter Password]

Organization and Wholesaler Registration Form
Register
Please provide the following details to create a new account:
- Name: [Enter Full Name]
- Email ID: [Enter Email Address]
- Password: [Enter Password]
- Confirm Password: [Confirm Password]
- Company Name: [Enter Company Name]
- Branch: [Enter Branch]
- Mailing Address:
- Street: [Enter Street Address]
- City: [Enter City]
- Province: [Enter Province]
- Postal Code: [Enter Postal Code]
- Billing Address:
- Street: [Enter Street Address]
- City: [Enter City]
- Province: [Enter Province]
- Postal Code: [Enter Postal Code]
- Phone Number: [Enter Phone Number]

Technician Registration Form

The "Technician Registration Form" is a user-friendly web page designed for technicians who wish to sign up for an account. Upon selecting the "Register" option from the main page, technicians will be directed to this form where they can provide their personal details to create a new account.

The form includes the following fields for technicians to enter their information:

Technician Registration Form

Please provide your information below to create a new technician account:

Personal Details

- First Name: [Enter First Name]
- Last Name: [Enter Last Name]

Company Details

- Company Name: [Enter Company Name]

Account Information

- Email: [Enter Email Address]
- Password: [Enter Password]
- Confirm Password: [Confirm Password]

Additional Information

- Date of Birth: [Enter Date of Birth]
- ODS License Number: [Enter ODS License Number]
- Gender: [Select Gender]

Address

- Address Line: [Enter Address Line]
- Province: [Select Province]
- City: [Enter City]
- Postal Code: [Enter Postal Code]

Phone Number

- Phone Number: [Enter Phone Number]

Technicians can fill out the form by entering their relevant information in each field. Once all the required details are provided, they can submit the form to create their technician account.

QR Tag Scanning and Cylinder Selection Page

When a technician scans a QR tag, they are directed to a page that allows them to choose a cylinder type. The page presents two options: "New Cylinder" or "Recovery Cylinder." Here's how the process unfolds:

Select Cylinder Type

When a technician scans a QR tag, they are presented with the option to select a cylinder type.

New Cylinder

Recovery Cylinder

New Cylinder Option

If the technician chooses the "New Cylinder" option, they are directed to a form where they can input information. The form includes non-editable auto-populated fields:

Cylinder Tag ID: [Auto-Populated]

Refrigerant ID: [Auto-Populated]

Technician ID: [Auto-Populated]

Select Date: [Choose Date]

Additionally, the technician can input the following details:

Refrigerant Type: [Enter Refrigerant Type]

Cylinder Size (lbs): [Enter Cylinder Size]

Refrigerant Weight (lbs): [Enter Refrigerant Weight]

Purchase Date: [Enter Purchase Date]

Wholesaler: [Enter Wholesaler Name]

Recovery Cylinder Option

If the technician selects the "Recovery Cylinder" option, they are directed to a form where they can input information. The form includes non-editable auto-populated fields:

Cylinder Tag ID: [Auto-Populated]

Company Name: [Auto-Populated]

Refrigerant ID: [Auto-Populated]

Technician ID: [Auto-Populated]

Recovery Date: [Choose Recovery Date]

The technician is prompted to provide the following details:

Refrigerant Type: [Enter Refrigerant Type]

Cylinder Type: [Select Cylinder Type: Clean/Reuse, Non-Usable, Burnout]

Cylinder Size (lbs): [Enter Cylinder Size]

Refrigerant Type (lbs): [Enter Refrigerant Weight]

The technician is provided with an interactive experience where they can select the appropriate cylinder type and proceed to enter the corresponding details. This process aids in efficient data collection and tracking for cylinder management.

#Refit Database
##Introduction
This database facilitates the storage and organization of critical data. It acts as a repository for user accounts, content, transactions, and various other essential components of the web application. The database empowers your application to handle high volumes of data and concurrent user interactions. This documentation provides an overview of the SQLite database used for the web application. It explains the purpose and structure of each table created in the database and outlines the relationships between them.
##Dropped TABLES
By dropping the tables before creating them again, it ensures a clean slate for the database. This can be useful when there is a need to reset the database structure. Dropping tables allows you to start fresh by recreating the tables with the desired structure and relationships.
##Tables
###User:
Stores user information for authentication and access control.
Columns:
user_id (Primary Key): Unique identifier for the user.
email: User's email address.
password: User's password.
role: User's role in the application.
added_date: Date the user was added to the system.
user_detail: Additional details about the user.
status: User's account status (active, inactive, etc.).
###Technician Table
Purpose: Stores information about technicians associated with contractors.
Columns:
technician_id (Primary Key): Unique identifier for the technician.
user_detail: Additional details about the technician.
user_id: Foreign key referencing the User table to link with the corresponding user.
contractor_id: Foreign key referencing the Contractor table to link with the associated contractor.
date_begin: Start date of the technician's association with the contractor.
date_end: End date of the technician's association with the contractor.
user_status: Status of the technician's user account.
contractor_status: Status of the technician's association with the contractor.
###Contractor Table
Purpose: Stores information about contractors associated with the application.
Columns:
contractor_id (Primary Key): Unique identifier for the contractor.
name: Contractor's name.
user_id: Foreign key referencing the User table to link with the corresponding user.
logo: Path or URL to the contractor's logo.
status: Contractor's status.
subscription_id: Foreign key referencing the Subscription table to link with the associated subscription.
code_2fa_code: Two-factor authentication code for the contractor's account.
###Contractor_Detail Table
Purpose: Stores additional details about contractors.
Columns:
contractor_id (Primary Key): Unique identifier for the contractor.
name: Contractor's name.
phone: Contractor's phone number.
address: Contractor's address.
employees: Number of employees working for the contractor.
are_they_tracking_refrigerant: Indicates if the contractor tracks refrigerant.

time_basis: Basis of time calculation for the contractor (hourly, daily, etc.).
###Refit_admin Table
Purpose: Stores information about administrators associated with the application.
Columns:
admin_id (Primary Key): Unique identifier for the admin.
name: Admin's name.
user_id: Foreign key referencing the User table to link with the corresponding user.
status: Admin's status.
code_2fa_code: Two-factor authentication code for the admin's account.
admin_level: Level of administrative access.
###Wholesaler Table
Purpose: Stores information about the associated wholesalers.
Columns:
wholesaler_id (Primary Key): Unique identifier for the wholesaler.
name: Wholesaler's name.
user_id: Foreign key referencing the User table to link with the corresponding user.
status: Wholesaler's status.
tag_id: Foreign key referencing the Tags table to link with associated tags.
###Invoices Table
Purpose: Stores invoice information related to subscriptions and tags.
Columns:
invoice_id (Primary Key): Unique identifier for the invoice.
subscription_id: Foreign key referencing the Subscription table to link with the associated subscription.
tag_id: Foreign key referencing the Tags table to link with the associated tag.
amount: Invoice amount.
payment_method: Payment method used.
tax: Tax amount.
date: Date of the invoice.
###Subscription Table
Purpose: Stores information about subscriptions.
Columns:
subscription_id (Primary Key): Unique identifier for the subscription.
Start_date: Start date of the subscription.
End_Date: End date of the subscription.
Package_size: Size of the subscription package.
compliant: Indicates if the subscription is compliant.
###Tags Table
Purpose: Stores information about tags associated with cylinders, and units.
Columns:
tag_id (Primary Key): Unique identifier for the tag.
tag_number: Tag number or identifier.
tag_url: URL or path associated with the tag.
type: Type of the tag.
cylinder_id: Foreign key referencing the Cylinder table to link with associated cylinders.
###USER LOGGING Table
Purpose: Stores logging information for user activities.
Columns:
log_id (Primary Key): Unique identifier for the log entry.
user_id: Foreign key referencing the User table to link with the corresponding user.
entry_date: Date and time of the user activity.
ip_address: IP address of the user.
address_gps: GPS coordinates of the user's location.
###User_detail Table

Purpose: Stores additional details about users.
Columns:
user_id (Primary Key): Unique identifier for the user.
ODS_license_number: License number for Ozone Depleting Substances handling.
first_name: User's first name.
middle_name: User's middle name.
last_name: User's last name.
birthdate: User's date of birth.
gender: User's gender.
address: User's address.
province: User's province.
city: User's city.
postal_code: User's postal code.
telephone: User's telephone number.
###Unit Table
Purpose: Stores information about units that are on record.
Columns:
unit_id (Primary Key): Unique identifier for the unit.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
unit_name: Name or identifier of the unit.
address: Address of the unit.
province: Province of the unit.
city: City of the unit.
postal_code: Postal code of the unit.
telephone: Telephone number of the unit.
tag_id: Foreign key referencing the Tags table to link with associated tags.
other_attribute: Other attribute related to the unit.
installation_date: Date of unit installation.
last_maintenance_date: Date of the last maintenance performed on the unit.
manufacturer: Manufacturer of the unit.
model: Model of the unit.
type_of_refrigerant: Type of refrigerant used in the unit.
factory_charge_amount: Amount of refrigerant initially charged in the unit.
unit_type: Type of the unit.
store_id: Foreign key referencing the Store table to link with the associated store.
Organizations Table
Purpose: Stores information about organizations/clients(of the contractor).
Columns:
organization_id (Primary Key): Unique identifier for the organization.
name: Name of the organization.
user_id: Foreign key referencing the User table to link with the corresponding user.
logo: Path or URL to the organization's logo.
status: Organization's status (active, inactive, etc.).
subscription_id: Foreign key referencing the Subscription table to link with the associated subscription.
code_2fa_code: Two-factor authentication code for the organization's account.
Store Table
Purpose: Stores information about stores associated which are under the organizations(clients[of the contractor]).
Columns:
store_id (Primary Key): Unique identifier for the store.
organization_id: Foreign key referencing the Organizations table to link with the associated organization.
branch: Store's branch information.

name: Store's name.
user_id: Foreign key referencing the User table to link with the corresponding user.
address: Store's address.
Store_locations Table
Purpose: Stores GPS locations for stores.
Columns:
store_id (Primary Key): Unique identifier for the store.
gps_location: GPS coordinates of the store.
ODS_Sheets Table
Purpose: Stores information about (ODS) sheets associated with various entries.
Columns:
ods_id (Primary Key): Unique identifier for the ODS sheet.
contractor_id: Foreign key referencing the Contractor table to link with the associated contractor.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
unit_id: Foreign key referencing the Unit table to link with the associated unit.
tag_id: Foreign key referencing the Tags table to link with associated tags.
repair_id: Foreign key referencing the Repairs table to link with the associated repair.
rec_id: Foreign key referencing the Reclaim_Recovery table to link with the associated reclaim/recovery.
technician_offer Table
Purpose: Stores information about offers made to technicians by contractors.
Columns:
contractor_id: Foreign key referencing the Contractor table to link with the associated contractor.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
offer_status: Status of the offer made.
email_time_sent: Date and time when the offer email was sent.
Cylinder Table
Purpose: Stores information about cylinders associated that are tagged.
Columns:
cylinder_id (Primary Key): Unique identifier for the cylinder.
cylinder_size: Size of the cylinder.
cylinder_type: Type of the cylinder.
cylinder_weight: Weight of the cylinder.
added_date: Date the cylinder was added.
refrigerant_id: Foreign key referencing the Refrigerant table to link with the associated refrigerant.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
purchase_date: Date of cylinder purchase.
supplier: Supplier of the cylinder.
last_refill_date: Date of the last refill performed on the cylinder.
condition: Condition of the cylinder.
tag_id: Foreign key referencing the Tags table to link with associated tags.
Repairs Table
Purpose: Stores information about repairs performed on units.
Columns:
repair_id (Primary Key): Unique identifier for the repair.
unit_id: Foreign key referencing the Unit table to link with the associated unit.
purchase_id: Foreign key referencing the Purchase table to link with the associated purchase.
repair_date: Date of the repair.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
causes: Causes of the repair.
status: Status of the repair.
Reclaim_Recovery Table
Purpose: Stores information about reclaim and recovery actions performed on refrigerants.

Columns:
rec_id (Primary Key): Unique identifier for the reclaim/recovery.
purchase_id: Foreign key referencing the Purchase table to link with the associated purchase.
tank_id: Foreign key referencing the Tank table to link with the associated tank.
unit_id: Foreign key referencing the Unit table to link with the associated unit.
gas_type: Type of gas reclaimed or recovered.
quantity_before_in_lbs: Quantity of gas before the reclaim/recovery process in pounds.
quantity_after_in_lbs: Quantity of gas after the reclaim/recovery process in pounds.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
notes: Additional notes about the reclaim/recovery.
date: Date of the reclaim/recovery.
status: Status of the reclaim/recovery.
refrigerant_id: Foreign key referencing the Refrigerant table to link with the associated refrigerant.
cylinder_id: Foreign key referencing the Cylinder table to link with the associated cylinder.
Refrigerant Table
Purpose: Stores information about refrigerants.
Columns:
refrigerant_id (Primary Key): Unique identifier for the refrigerant.
refrigerant_name: Name of the refrigerant.
list: Additional information about the refrigerant.
Maintenance Table
Purpose: Stores information about maintenance activities performed on units.
Columns:
maintenance_id (Primary Key): Unique identifier for the maintenance activity.
technician_id: Foreign key referencing the Technician table to link with the associated technician.
unit_id: Foreign key referencing the Unit table to link with the associated unit.
log: Log of maintenance activity.
last_updated: Date and time of the last update to the maintenance record.
service_history: History of maintenance services performed.
maintenance_date: Date of the maintenance activity.
maintenance_type: Type of the maintenance activity.
parts_used: Parts used during the maintenance activity.
notes: Additional notes about the maintenance activity.
Maintenance_detail Table
Purpose: Stores detailed information about individual maintenance activities.
Columns:
maintenance_detail_id (Primary Key): Unique identifier for the maintenance detail.
maintenance_id: Foreign key referencing the Maintenance table to link with the associated maintenance activity.
description: Description of the maintenance detail.
status: Status of the maintenance detail.
##ERD Diagram
After the program is run, and the database is created, the UML diagram is created using DBeaver database tool.
Connect to the database (database.db)
Right-click on the connected database on left-panel.
Click on view diagram.

GPS location form with address suggestions

This outlines the process for a technician using an application to fill out a form while utilizing GPS location and address suggestions. The technician begins by accessing the application's main dashboard and selecting the "Scan QR Code" button to initiate the process. Upon selecting this button, the application requests permission to access the technician's GPS location.

If permission is granted, the technician proceeds to the form page. This page includes fields for personal information and an address field. When the technician interacts with the address field, two branches are possible. If the technician selects a suggested address, the address field auto-fills with related information. On the other hand, if the technician manually enters an address, an elastic search query is triggered to provide address suggestions. Once an address is selected, related fields auto-fill accordingly.

Following the completion of the form, a validation process occurs. If the form is valid, the submitted data is stored, and a confirmation message is displayed. The technician is then redirected back to the main dashboard. In case of invalid form data, a validation error message appears, and the technician remains on the form page.

Overview

The Contractor Dashboard offers users multiple functionalities, one of which is a QR scanner. This feature allows for the quick and easy scanning of QR codes, with the decoded data subsequently sent to the server for further processing.

QR Scanner Library: ZXing

The dashboard uses the ZXing (Zebra Crossing) library for barcode image processing. This well-regarded library enables the reading of various barcode formats, including QR codes, directly from the browser using the device's camera.

How the QR Scanner Works

Initialization:

Upon accessing the dashboard, the QR code reader is initialized and readied for use.

Starting the Scanner:

Clicking the "Reclaim/Recover" button activates the QR code scanning process. Once activated, the camera feed becomes visible, allowing users to position a QR code for scanning.

Decoding the QR Code:

The library automatically detects and decodes any QR code presented in the camera's view. Once decoded, the result is presented on the dashboard.

Sending Decoded Data to the Server:

After the QR code is successfully decoded, the dashboard sends the decoded information to the server via a POST request. This data can then be processed further based on the application's requirements.

Error Handling:

In case of any issues during the scanning process, such as the inability to access the camera or an unrecognized QR code, a corresponding error message is displayed on the dashboard.

Conclusion

The QR scanner integrated into the Contractor Dashboard is both efficient and user-friendly. By leveraging the capabilities of the ZXing library, the dashboard ensures seamless compatibility across a wide range of devices and browsers, offering users a hassle-free scanning experience.

1. The scanner needs to be added to other pages where required, right now it is only on contractor page
2. Need to test login, logout, register, and other pages
3.

Update Note
Summary
During the conversation, the following tasks were accomplished:
1. A new unit testing file was created in the `tests` folder, named `test_user.py`.
2. The `models.py` file was examined, which contains SQLAlchemy models for various database tables.
3. Unit tests were generated for the `User` and `User_Detail` models.
4. Several separate unit testing files were created for the `Technician`, `Admin`, and `Contractor` models.
5. The complete `test_technician.py`, `test_admin.py`, and `test_contractor.py` files were provided for testing the respective models.
Files
1. [`test_user.py`](test_user.py) - Unit tests for the `User` model.
2. [`test_user_detail.py`](test_user_detail.py) - Unit tests for the `User_Detail` model.
3. [`test_technician.py`](test_technician.py) - Unit tests for the `Technician` model.
4. [`test_admin.py`](test_admin.py) - Unit tests for the `Admin` model.
5. [`test_contractor.py`](test_contractor.py) - Unit tests for the `Contractor` model.

Models.py
Introduction
SQLAlchemy is a powerful SQL toolkit and Object-Relational Mapping (ORM) system that enables Python developers to interact with their database like they would with SQL, and also in a more Pythonic and object-oriented way. This document will provide an overview of the SQLAlchemy models defined in our codebase, which will aid any developer who takes over this project in understanding how we are interacting with our database.
Model Overview
Each SQLAlchemy model in our codebase corresponds to a table in our database. For instance, the User model corresponds to the 'User' table, Technician model to the 'Technician' table, and so forth. The attributes of each model represent the columns in their respective tables. These models are an abstraction that allows us to interact with our database in a more intuitive and Pythonic way.
A brief description of what each model represents:
2. Database Schema
The database schema consists of the following tables:
2.1 User
Represents user information with columns like user_id, email, password, role, etc.
Has relationships with several other classes representing different user roles (e.g., Technician, Contractor, Refit_admin, Wholesaler, etc.).
2.2 User_detail
Contains additional details about users with columns like user_detail_id, ODS_license_number, first_name, last_name, etc.
Has a relationship with the User class.
2.3 Technician
Represents information about technicians with columns like technician_id, user_detail, user_id, etc.
Has relationships with the User class and several other classes representing different technician-related data (e.g., Unit, ODS_Sheets, Cylinder, etc.).
2.4 Contractor
Represents contractor information with columns like contractor_id, name, user_id, etc.
Has relationships with the User class and other classes related to contractors (e.g., Technician_offer, Contractor_Detail, etc.).
2.5 Refit_admin
Represents refit admin information with columns like admin_id, name, user_id, etc.
Has a relationship with the User class.
2.6 Wholesaler
Represents wholesaler information with columns like wholesaler_id, name, user_id, etc.
Has relationships with the User class and Tags class.
2.7 Tags
Represents tags with columns like tag_id, invoice_id, tag_number, etc.
Has relationships with several other classes related to different entities using tags (e.g., Wholesaler, Cylinder, Unit, etc.).
2.8 Invoices
Represents invoice information with columns like invoice_id, subscription_id, tag_id, etc.
Has relationships with the Tags class and the Subscription class.
2.9 Subscription
Represents subscription information with columns like subscription_id, Start_date, End_Date, etc.
Has relationships with the Invoices class and the Organizations class.
2.10 Organizations
Represents organization information with columns like organization_id, name, logo, etc.
Has relationships with the Store class, the Subscription class, and the User class.
2.11 Store
Represents store information with columns like store_id, organization_id, branch, etc.
Has relationships with the Organizations class and the User class.

2.12 Store_locations
Represents store locations with columns like store_id and gps_location.
2.13 Cylinder
Represents cylinder information with columns like cylinder_id, cylinder_size, cylinder_type, etc.
Has relationships with the Reclaim_Recovery class, the Refrigerant class, the Technician class, and the Tags class.
2.14 Repairs
Represents repair information with columns like repair_id, unit_id, purchase_id, etc.
Has relationships with the Unit class, the Technician class, and the ODS_Sheets class.
2.15 Reclaim_Recovery
Represents information related to the reclaim and recovery process with columns like rec_id, purchase_id, unit_id, etc.
Has relationships with the ODS_Sheets class, the Unit class, the Technician class, the Cylinder class, and the Refrigerant class.
2.16 Refrigerant
Represents refrigerant information with columns like refrigerant_id, refrigerant_name, etc.
Has relationships with the Cylinder class and the Reclaim_Recovery class.
2.17 Maintenance
Represents maintenance information with columns like maintenance_id, technician_id, unit_id, etc.
Has relationships with the Maintenance_detail class, the Unit class, and the Technician class.
2.18 Maintenance_detail
Represents details related to maintenance with columns like maintenance_detail_id, maintenance_id, description, status, etc.
Has a relationship with the Maintenance class.
2.19 Technician_offer
Represents an offer made by a technician to a contractor with columns like contractor_id, technician_id, offer_status, etc.
Has relationships with Contractor and Technician.
2.20 Contractor_Detail
Represents detailed information about a contractor with columns like contractor_id, name, phone, address, etc.
Has a relationship with the Contractor class.
2.21 Unit
Represents a unit in the database with columns like unit_id, unit_name, address, province, etc.
Has relationships with Technician, Tags, and other classes for different types of relationships.
2.22 ODS_Sheets
Represents sheets related to ODS (Ozone-Depleting Substances) with columns like ods_id, contractor_id, technician_id, etc.
Has relationships with Contractor, Technician, Unit, Tags, Repairs, and Reclaim_Recovery.
3. CRUD Operations
The code provides CRUD (Create, Read, Update, Delete) operations using the CRUD class. These operations can be performed on any of the defined model classes (e.g., User, Technician, Contractor, etc.). The CRUD operations are as follows:
3.1 Create
The create method allows creating a new instance of a model class and adding it to the database.
Example: User.create(User, email='example@example.com', password='password', role='technician', ...)
3.2 Read
The read method allows querying the database for instances of a model class based on filters.
Example: User.read(User, User.role == 'technician', User.status == 'active')
3.3 Update
The update method allows updating an existing instance of a model class.
Example: User.update(User, user_id=1, email='newemail@example.com')

3.4 Delete
The delete method allows deleting an instance of a model class based on its primary key.
Example: User.delete(User, user_id=1)
Relationships in SQLAlchemy
SQLAlchemy allows us to define relationships between tables. These relationships are represented in the models by the relationship() function. For example, in the User model, we have a relationship to the Technician model. This represents a one-to-one relationship: one user can have one user detail. The backref argument creates a back reference from Technician to User, allowing us to access a technician's user by using technician.user.

Refit Admin User Flow

Refit Admins play a crucial role in maintaining the Refit service, ensuring accounts are created and used properly. They possess the ability to oversee accounts with finer detail, create custom reports, and provide assistance with user issues. Refit Admins exclusively use the desktop website for their tasks.

Login and Logout

1. Refit Admin accesses the Refit login page.
2. Refit Admin enters their credentials (username and password).
3. The system verifies the credentials.
4. If the credentials are valid, the Refit Admin gains access to the system.
5. After completing their tasks, the Refit Admin logs out to end the session.

Search and View Account Information and History

1. Refit Admin navigates to the account management section.
2. Refit Admin searches and views information and history of various accounts, including regulators, companies, and service people.
3. Refit Admin can access detailed account information to ensure proper functioning.

Make Changes to Accounts

1. Refit Admin selects the account to modify in the account management section.
2. Refit Admin makes necessary changes to accounts (e.g., updating contact information or account settings).
3. The system saves the changes and updates the account accordingly.

Search and View Complete Histories of Units and Refrigerant Tanks

1. Refit Admin accesses the unit and tank history section.
2. Refit Admin can search and view the complete histories of all units and refrigerant tanks in the system.
3. Detailed historical data is available for analysis and decision-making.

Make Changes to Units and Tanks

1. Refit Admin selects the unit or tank to modify in the unit and tank management section.
2. Refit Admin makes necessary changes to units or tanks (e.g., updating specifications or statuses).
3. The system saves the changes and updates the respective records.

Generate Reports

1. Refit Admin accesses the report generation section.
2. Refit Admin can generate custom reports based on various accounts or histories (e.g., regulator reports, company reports, or service people reports).
3. The system generates and presents the reports for review and analysis.

View Trends

1. Refit Admin navigates to the trends section.
2. Refit Admin can view trends based on various accounts or histories (e.g., trends in refrigerant usage, unit repairs, or tank refills).
3. Graphical representations aid in understanding the data and identifying patterns.

Create, Modify, and Delete Accounts

1. Refit Admin has the authority to create new accounts for service people, companies, wholesalers, and equipment owners.
2. Refit Admin can modify account information as required (e.g., updating user details or company information).
3. If necessary, Refit Admin can delete accounts while adhering to appropriate procedures.

Investigate Bugs and Issues with the Platform

1. Refit Admin receives bug reports via email from users or system alerts.
2. Refit Admin investigates the reported bugs or issues.
3. Refit Admin works with the development team to identify and address the root cause of the problems.
4. Upon resolution, Refit Admin communicates the updates and solutions to the users.

User Flow for Contractor

Create a New Account

1. User accesses the account creation page.
2. User provides necessary details to create a new account, such as name, user ID, logo, subscription ID, status, and 2FA code.
3. System validates the information provided.
4. If validation is successful, the contractor account is created.
5. User can now log in to the system.

Login and Logout

1. User enters their credentials (user ID and password) on the login page.
2. System verifies the credentials.
3. If the credentials are valid, the user is granted access to the system.
4. User performs desired actions within the system.
5. When finished, the user logs out of the system to end the session.

Offer a Technician to Join the Team

1. User (contractor) accesses the technician management section.
2. User selects the option to invite a new technician.
3. User provides the necessary details for the technician invitation, such as name and email.
4. System sends an invitation email to the technician.
5. Technician receives the invitation and can choose to accept or decline it.
6. If the technician accepts, they are added to the contractor's team.

Purchase the Tags

1. User (wholesaler) accesses the tag purchase section.
2. User selects the desired quantity and type of tags.
3. User proceeds with the purchase and provides necessary payment details.
4. System processes the payment and generates the tags.
5. User receives the purchased tags.

Delete the Connection of Technician

1. User (contractor) accesses the technician management section.
2. User selects the technician account to be removed.
3. User confirms the deletion request.
4. System removes the technician's connection from the contractor's team.
5. The technician's historical records are retained, and they can still access their records independently.

Replace the Invalid QR Tag with a New One

1. User (service person) encounters an invalid QR tag on a refrigeration unit.
2. User accesses the QR tag replacement section.
3. User scans the invalid tag and selects the replacement option.
4. User attaches a new QR tag to the unit.
5. User logs the replacement in the system, associating the new tag with the unit.

Organization User Flow
Request an Account for the Company
1. Organization accesses the account creation page.
2. Organization requests to have an account made by providing company details.
3. System assigns a unique ID to the organization's account.
4. Organization can use the assigned ID to log in later.
Login and Logout
1. Organization enters their credentials (unique ID and password) on the login page.
2. System verifies the credentials.
3. If the credentials are valid, the organization is granted access to the system.
4. Organization performs desired actions within the system.
5. When finished, the organization logs out of the system to end the session.
Manage Administrator Accounts
1. Organization accesses the administrator management section.
2. Organization can create, modify, or delete additional administrator accounts.
3. Administrator accounts can log in and view the company information.
Make Changes to Account Information
1. Organization accesses the account settings page.
2. Organization views and updates their account information, such as company details and contact information.
3. Organization can modify their account details as needed.
Manage Employees
1. Organization accesses the employee management section.
2. Organization can add, modify, or remove an employee from the company account.
3. Organization sends an invitation to a new employee to join the company.
View History of Refrigerant Usage
1. Organization accesses the refrigerant usage history section.
2. Organization can view the history of refrigerant usage, which may be company-wide or specific to service persons, units, or tanks of refrigerant.
View Technicians' ODS History
1. Organization accesses the technician ODS history section.
2. Organization can view the history of ODS transactions performed by technicians while employed at their business.
View History of Employment
1. Organization accesses the employment history section.
2. Organization can view the history of employment and associated service persons' accounts.
Generate Reports
1. Organization accesses the report generation section.
2. Organization generates reports based on the previously-mentioned histories, such as refrigerant usage, ODS history, and employment records.
View Issues and Rulings
1. Organization accesses the issues and rulings section.
2. Organization views any current or past issues/inconsistencies with refrigerant usage.
3. Organization also views any current or past rulings from governing bodies.
Submit Reports
1. Organization accesses the report submission section.
2. Organization can submit reports regarding service person, tank, or unit issues.

Store (Branch) User Flow

Create an Account

1. Store (Branch) accesses the account creation page.
2. Store (Branch) provides necessary details to create a new account, such as organization ID, branch name, user ID, store name, and address.
3. System validates the information provided.
4. If validation is successful, the Store (Branch) account is created.
5. Store (Branch) can now log in to the system.

Check ODS History

1. Store (Branch) logs in to the system using their credentials (user ID and password).
2. Store (Branch) accesses the ODS history section.
3. Store (Branch) can view the ODS history for their specific branch, including ODS transactions, technician information, unit details, and repair records.

View GPS Locations

1. Store (Branch) logs in to the system using their credentials (user ID and password).
2. Store (Branch) accesses the GPS locations section.
3. Store (Branch) can view the GPS locations associated with their branch, enabling them to track the geographic positions of refrigerant units or equipment.

Introduction
The Refit System is a digital platform designed to streamline and simplify the tracking of refrigerant gases in Canada. It aims to replace the current paper-based tracking system with a cloud-based solution, making it more efficient and automated for end users, including service people, companies, equipment owners, and wholesalers. The system allows for easy management of refrigerant transactions, equipment repairs, and maintenance records, providing a comprehensive solution for compliance with rules and regulations.
The Refit System caters to different user roles, including Technician, Contractor ,Organization, store, wholesaler, and Refit Administrators. Each user has specific functionalities and information access based on their role and requirements.
Technician user flow
Step 1: Create a New Account
1. User accesses the account creation page.
2. User provides necessary details to create a new account, such as name, email, and password.
3. System validates the information provided.
4. If validation is successful, the user account is created.
5. User can now log in to the system.
Step 2: Log in and Log out
1. User enters their credentials (email and password) on the login page.
2. System verifies the credentials.
3. If the credentials are valid, the user is granted access to the system.
4. User performs desired actions within the system.
5. When finished, the user logs out of the system to end the session.
Step 3: Make Changes to Account
1. User accesses the account settings page.
2. User views and updates their account information, such as name, email, and password.
3. User can modify their account details as needed.
Step 4: Scan Existing QR Codes on Equipment
1. User (Service Person) uses the Refit app to scan QR codes on refrigeration equipment.
2. The system retrieves and displays relevant equipment information associated with the QR code.
Step 5: View History of Equipment Repairs and Refills
1. User (Service Person) accesses the equipment history section.
2. User views the repair and refill history of a specific equipment unit.
3. The system displays details such as repair dates, refill quantities, and maintenance records.
Step 6: Record Repairs and Refills
1. User (Service Person) uses the Refit app to record any repairs or refills performed on equipment.
2. User provides necessary details, such as repair type, quantity of refrigerant refilled, and maintenance notes.
3. The system logs the repair and refill information for the equipment unit.
Step 7: Create New QR Code Listings on Equipment
1. User (Service Person) accesses the QR code creation section.
2. User creates new QR code listings for equipment units that do not have QR codes.
3. The system generates QR codes and associates them with the respective equipment units.
Step 8: Scan QR Codes on Tanks of Refrigerant
1. User (Service Person) uses the Refit app to scan QR codes on tanks of refrigerant.
2. The system retrieves and displays information about the refrigerant in the tank.
Step 9: Replace Invalid QR Tag with a New One
1. User (Service Person) encounters an invalid QR tag on a refrigeration unit.
2. User accesses the QR tag replacement section.
3. User scans the invalid tag and selects the replacement option.
4. User attaches a new QR tag to the unit.
5. User logs the replacement in the system, associating the new tag with the unit.
Step 10: View History of Tank of Refrigerant

1. User (Service Person) accesses the tank history section.
2. User views the usage history of a specific tank of refrigerant.
3. The system displays details such as tank refill dates, quantities used, and any related issues.
Step 11: View Current or Past Issues with Refrigerant
1. User (Service Person) accesses the refrigerant issues section.
2. User views any current or past issues or inconsistencies with refrigerant usage.
3. The system provides details about the issues and their resolution status.
Step 12: View History of Refrigerant Use
1. User (Service Person) accesses the refrigerant use history section.
2. User views their history of refrigerant use, including locations, quantities used, and types of refrigerant.
3. The system also displays maintenance history associated with the refrigerant use.
Step 13: Ability to Save to Device and Push to Cloud
1. In case of no internet connection, the user can save data to their device locally.
2. When the internet connection is restored, the user can push the saved data to the cloud for synchronization.
Step 14: Leave Current Employer or Accept Employment Invitation
1. User accesses the employment status section.
2. User can choose to leave their current employer or accept an employment invitation from a new employer.
3. The system updates the user's employment status accordingly.