

Freight Cargo Inventory and Shipment Management System

Final Report

Muhammad Ahmad 30239993

Syed Essam Uddin Khawaja 30242943

Muhammad Zohaib Talat 30257908

CPSC 471 W26 - Group G-4

Abstract

This report presents the design, implementation, and evaluation of a Freight Cargo Inventory and Shipment Management System built for CPSC 471. The system models a global freight logistics network using a normalized PostgreSQL relational database hosted on Supabase. The schema consists of 15 relations (tables), 2 views, 2 stored functions, and 1 trigger, all derived from an Enhanced Entity-Relationship Diagram through the standard EERD-to-relational-model conversion algorithm. The application frontend is built with Next.js 15 (React 19, TypeScript, Tailwind CSS) and communicates with the database via parameterized SQL queries using the pg library. Authentication is handled by Supabase Auth with role resolution performed through SQL JOINS against the application's own user/role tables. The system supports two roles (Admin and Warehouse Staff), full CRUD operations on all entities, personalized dashboards scoped to assigned warehouses, inventory management via a stored function with negative-quantity guards, automatic audit logging via triggers, user registration approval workflows, and staff-to-admin promotion requests. The seed dataset models a realistic global freight network spanning 63 locations across 6 continents, 12 warehouses, 22 carriers, 55 shipments, and 77 containers. A live platform demo is deployed on Vercel at <https://cpsc-471-inventory-manager.vercel.app>. Its connected PostgreSQL relational database can be viewed on Supabase, where it is hosted, by logging into the following account; email ‘’ and password ‘’.

Report Overview

This report is organized as follows:

1. **Section 1 - Introduction:** outlines the freight logistics problem, project goals, and overall system structure.
2. **Section 2 - Project Design:** explains the target users, key design decisions, and implementation trade-offs.
3. **Section 3 - Enhanced Entity-Relationship Diagram (EERD):** presents the conceptual database design and the changes made from the earlier course deliverables.
4. **Section 4 - Relational Model Diagram (RM):** presents the relational schema derived from the EERD, compares it with the earlier Progress Report 2 version, and explains the key conversion decisions.
5. **Section 5 - Supporting UML and Architecture Diagrams:** documents the PlantUML class, sequence, component, and state diagrams that describe the complete database schema, authentication flow, system architecture, user lifecycle, inventory adjustment and audit trail, application page map, and database setup sequence.
6. **Section 6 - Implementation:** describes the DBMS choice and representative SQL transactions used in the running system.
7. **Section 7 - Visual Interface Description:** summarizes the major application pages and how users interact with the frontend.
8. **Section 8 - Supabase Dashboard and SQL Editor:** shares login information to our Supabase and documents the hosted database environment and direct SQL verification workflow.
9. **Section 9 - User Guide:** provides a practical walkthrough for accessing and using the application as an admin, staff member, or newly registered user.
10. **Section 10 - Summary of Changes from Initial Design:** compares the original proposal and progress-report plans with the final implemented system.
11. **Section 11 - Conclusion:** reflects on the completed system and how it meets the course objectives.

1. Introduction

1.1 Problem Description

Freight logistics companies manage enormous volumes of cargo moving between warehouses, ports, and distribution centers worldwide. Tracking which cargo items are stored at which warehouse, which containers hold which cargo, which shipments are in transit, and which carriers handle which routes requires a robust relational database at the center of the operation. Without a proper database system, this information is often tracked in spreadsheets, leading to data inconsistencies, lost inventory, and missed shipments.

1.2 Project Goal

Our goal was to design and implement a database-centric system that:

1. Models the real-world entities and relationships of a freight cargo operation using a normalized relational schema
2. Enforces data integrity through primary keys, foreign keys, NOT NULL constraints, and CHECK-like guards in stored functions
3. Demonstrates advanced SQL features: views for reporting, stored functions for business logic, and triggers for automatic audit logging
4. Provides a usable web interface for two distinct user roles (Admin and Staff) to interact with the data
5. Includes realistic seed data for demonstration and evaluation

1.3 System Overview

The system we built is a full-stack web application with the following components:

- **Database:** PostgreSQL hosted on Supabase, with schema defined in ordered SQL files (sql/001 through sql/005)
- **Backend:** Next.js 15 App Router server components and API routes that execute direct parameterized SQL against the database via the pg library
- **Frontend:** React 19 with Tailwind CSS, featuring a role-filtered sidebar, dark/light mode toggle, and mobile-responsive layout
- **Authentication:** Supabase Auth for session management (JWT cookies), with role resolution via SQL JOINS against our own users, admins, and warehouse_staff tables

2. Project Design

2.1 Users of the System

The system serves three categories of users:

Administrators (Admin role)

- Have full access to all entity management pages (warehouses, carriers, routes, locations, shipments, containers, cargo items, inventory)
- Can create, read, update, and delete records in all tables
- Manage other users: create new accounts, approve pending registrations, approve or reject staff-to-admin promotion requests, assign users to warehouses, demote admins back to staff
- Access the Reports page with aggregated views
- See a personalized dashboard showing KPIs, recent shipments, and warehouse capacity for their assigned warehouses only
- Each admin can manage multiple warehouses (many-to-many through the manages table)

Warehouse Staff (Staff role)

- Have access to a subset of entity pages: Dashboard, Inventory, Shipments, Containers, and Cargo Items
- Can view and manage inventory records for their assigned warehouse
- Can create new cargo items and adjust inventory quantities
- Can submit a request for admin promotion (with a reason, max 1000 characters)
- See a personalized dashboard scoped to their single assigned warehouse
- Do not have access to master data management (carriers, routes, locations), user management, or reporting

Pending Users

- Users who self-register through the /register page start with `account_status = 'pending_registration'`
- They can log in, but are immediately redirected to the /pending page
- They cannot access any entity pages until an admin approves their registration

2.2 Design Decisions and Trade-offs

Email-based linking (no auth_user_id column): We chose to link Supabase Auth users to our public.users table by email rather than storing a Supabase auth_user_id column. This avoids coupling our schema to Supabase internals and means all user management can happen through SQL without needing the Supabase dashboard.

Disjoint specialization tables retained: The EERD specifies User ISA Admin and User ISA WarehouseStaff as a disjoint total specialization. We kept separate admins and warehouse_staff tables rather than collapsing them into a single table with a role column. This was required by the earlier EERD design and also makes role determination a simple LEFT JOIN operation.

Direct SQL (no ORM): All application data access uses parameterized SQL queries via the pg library. We chose this deliberately to maximize SQL visibility and demonstrate database concepts directly. Supabase's ORM connection snippet is used only during setup to obtain DATABASE_URL; the running application itself does not use an ORM.

Gapless user IDs: We use manual MAX(user_id) + 1 inserts for new users instead of relying solely on BIGSERIAL auto-increment. This avoids gaps in user IDs that would result from failed or rolled-back inserts, which is important for a demonstration system where IDs are visible in the UI.

Server actions with redirect for error handling: Next.js server actions cannot return values to server components directly. We use redirect('/path?error=message') and redirect('/path?success=message') patterns to communicate results back to the user.

3. Enhanced Entity-Relationship Diagram (EERD)

3.1 Original EERD (Progress Report 1)

The original EERD was designed during Progress Report 1. It contained the following entities and relationships:

Entities: User, Admin (specialization), Warehouse Staff (specialization), Warehouse, Location, CargoItem, InventoryRecord (weak entity), Carrier, Route, Shipment, Container

Relationships: ISA (disjoint), MANAGES, WORKS_AT, CREATES, UPDATES, LOCATED_AT, STORES, TRACKS, HANDLED_BY, USES, DELIVERED TO/FROM, ASSIGNED, CONTAINS

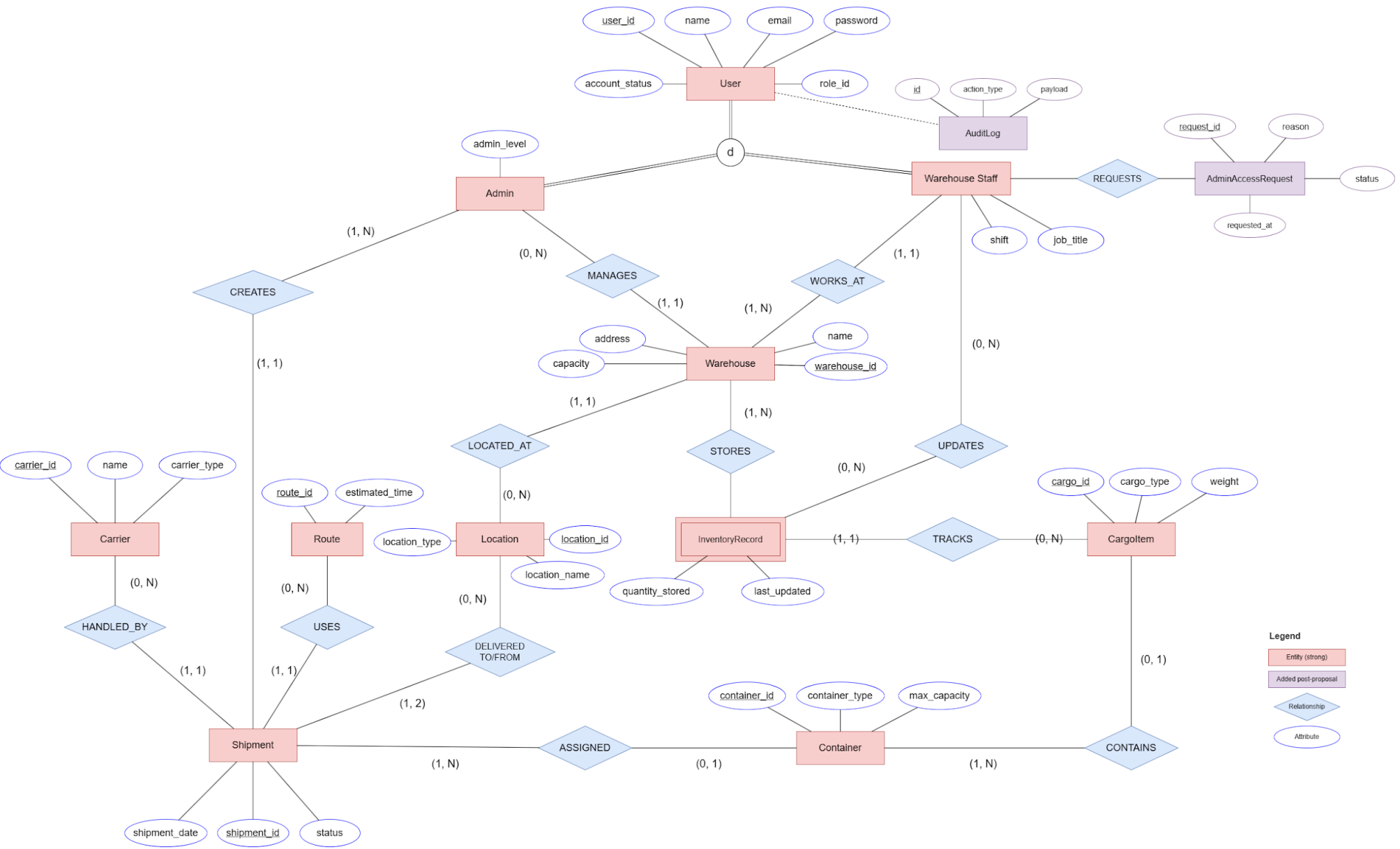
3.2 Final EERD (Changes from Original)

The final EERD is in docs/RM&EERD/EERD.drawio. The following changes were made during implementation:

12. **Added `account\_status` attribute to User** - needed for the registration approval workflow. Values: active, pending_registration, rejected, disabled.
13. **Added `role\_id` attribute to User** - explicit role indicator (1=Admin, 2=Staff) used alongside the specialization tables for quick lookups.
14. **Added AdminAccessRequest entity** - a new entity for the staff-to-admin promotion workflow, with attributes: request_id (PK), user_id (FK), reason, status, requested_at, resolved_at, resolved_by_user_id (FK). Connected to User via a REQUESTS relationship.
15. **Added AuditLog entity** - for tracking data changes, with attributes: id (PK), user_id (FK, nullable), action_type, entity_type, entity_id, payload (JSONB), created_at. Connected to User via a dashed optional link (ON DELETE SET NULL).
16. **Added `warehouse\_id` attribute to WarehouseStaff** - a direct foreign key to Warehouse for simple staff-warehouse assignment, in addition to the WORKS_AT relationship.
17. **Corrected InventoryRecord notation** - shown with a double border to properly indicate it is a weak entity, uniquely identified by the composite key (cargo_id, warehouse_id).

All original entities and relationships from the Prog1 EERD were preserved. No entities or relationships were removed.

Freight Cargo Inventory and Shipment Management System
Enhanced Entity-Relationship Diagram (EERD) - Final



3.3 Entity Summary Table

Entity	Type	PK	Key Attributes
User	Strong	user_id	name, email, password, role_id, account_status
Admin	Specialization of User	user_id (FK)	admin_level
WarehouseStaff	Specialization of User	user_id (FK)	job_title, shift, warehouse_id (FK)
Warehouse	Strong	warehouse_id	name, address, capacity, location_id (FK)
Location	Strong	location_id	location_name, location_type
CargoItem	Strong	cargo_id	cargo_type, weight
InventoryRecord	Weak (on Warehouse + CargoItem)	(cargo_id, warehouse_id)	quantity_stored, last_updated
Carrier	Strong	carrier_id	name, carrier_type
Route	Strong	route_id	estimated_time
Shipment	Strong	shipment_id	shipment_date, status, carrier_id, route_id, origin_loc_id, destination_loc_id
Container	Strong	container_id	container_type, max_capacity, shipment_id (FK, nullable)
ContainerCargo	Associative	(cargo_id, container_id)	(none; pure junction table)
AuditLog	Strong (added)	id	user_id, action_type, entity_type, entity_id, payload, created_at
AdminAccessRequest	Strong (added)	request_id	user_id, reason, status, requested_at, resolved_at, resolved_by_user_id

3.4 Relationship Summary Table

Relationship	Entity A	Entity B	Cardinality A	Cardinality B
ISA (disjoint)	User	Admin, WarehouseStaff	1	1
MANAGES	Admin	Warehouse	(0, N)	(1, 1)
WORKS_AT	WarehouseStaff	Warehouse	(1, 1)	(1, N)
CREATES	Admin	Shipment	(1, N)	(1, 1)
UPDATES	WarehouseStaff	InventoryRecord	(0, N)	(0, N)
LOCATED_AT	Warehouse	Location	(1, 1)	(0, N)
STORES	Warehouse	InventoryRecord	(1, N)	--
TRACKS	InventoryRecord	CargoItem	(1, 1)	(0, N)
HANDLED_BY	Carrier	Shipment	(0, N)	(1, 1)
USES	Route	Shipment	(0, N)	(1, 1)
DELIVERED TO/FROM	Location	Shipment	(0, N)	(1, 2)
ASSIGNED	Shipment	Container	(1, N)	(0, 1)
CONTAINS	Container	CargoItem	(1, N)	(0, 1)
REQUESTS	WarehouseStaff	AdminAccessRequest	(0, N)	(1, 1)

4. Relational Model Diagram (RM)

4.1 Original Relational Model (Progress Report 2)

The original relational model was designed during Progress Report 2. It contained the following 13 relations derived from the original EERD:

Relations: User, Admin, WarehouseStaff, Manages, Warehouse, Location, CargoItem, InventoryRecord, Carrier, Route, Shipment, Container, ContainerCargo

This version already included `role_id` in User and `warehouse_id` in WarehouseStaff, but it did not yet include the account lifecycle or approval/audit workflow relations that were added later during implementation.

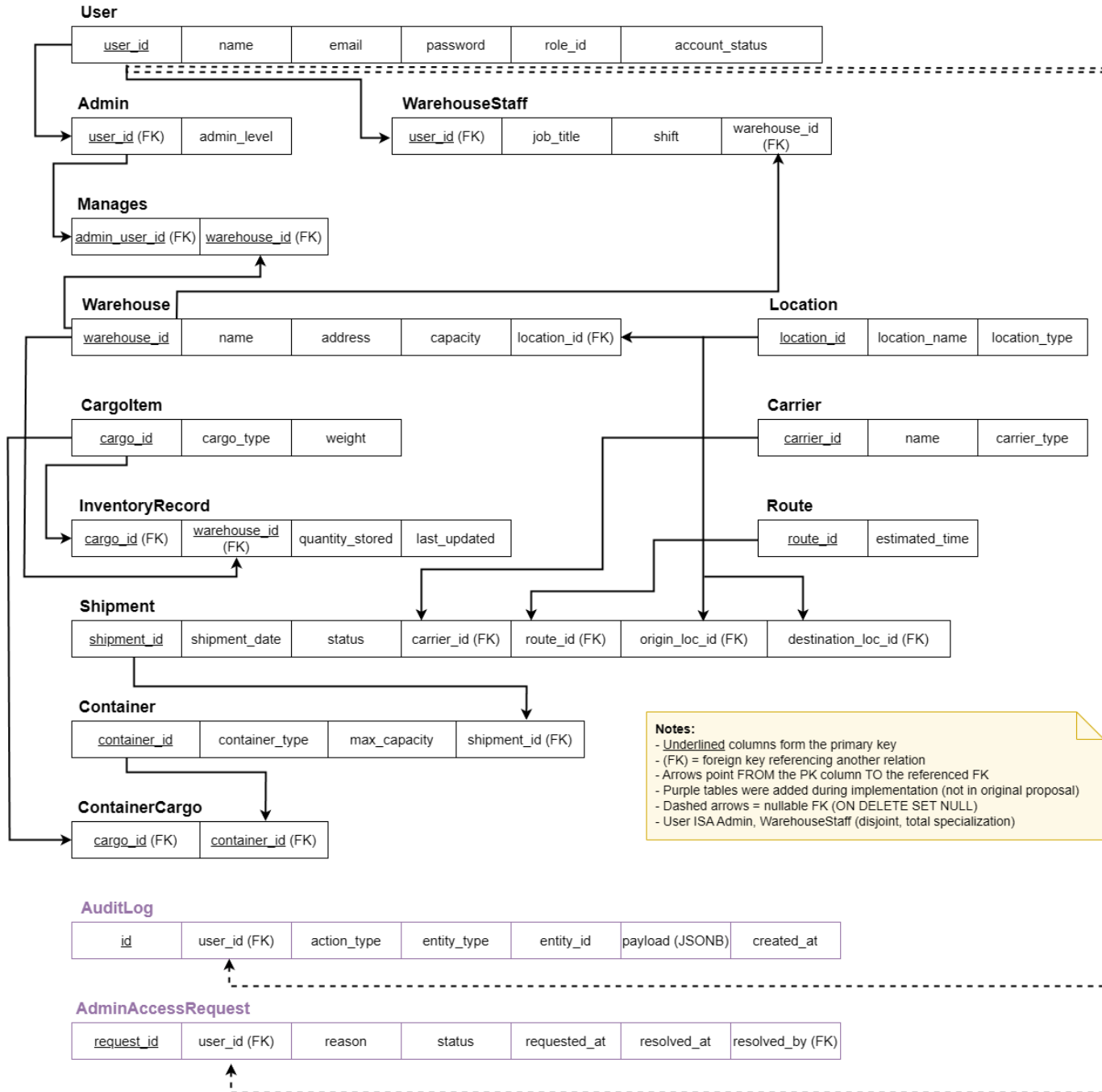
4.2 Final Relational Model (Changes from Original)

The final relational model is in `docs/RM&EERD/RelationalModel.drawio`. The following changes were made during implementation:

18. **Added `account\_status` to User** - needed to represent registration and account lifecycle states such as `pending_registration`, `active`, `rejected`, and `disabled`.
19. **Added AuditLog relation** - a new relation for persistent change tracking with columns: `id` (PK), `user_id` (FK nullable), `action_type`, `entity_type`, `entity_id`, `payload`, `created_at`.
20. **Added AdminAccessRequest relation** - a new relation supporting the staff-to-admin promotion workflow, with columns: `request_id` (PK), `user_id` (FK), `reason`, `status`, `requested_at`, `resolved_at`, `resolved_by_user_id` (FK nullable).
21. **Retained all 13 original relations from Progress Report 2** - the existing core freight, warehouse, shipment, and inventory relations were preserved and extended rather than replaced.

The final RM therefore contains 15 relations in total.

Freight Cargo Inventory and Shipment Management System Relational Model (Final Implementation)



4.3 Relation Summary Table

Relation	Type	Primary Key	Key Columns
User	Strong	user_id	name, email, password, role_id, account_status
Admin	Specialization of User	user_id (FK)	admin_level
WarehouseStaff	Specialization of User	user_id (FK)	job_title, shift, warehouse_id (FK)
Manages	Associative	(admin_user_id, warehouse_id)	admin_user_id (FK), warehouse_id (FK)
Warehouse	Strong	warehouse_id	name, address, capacity, location_id (FK)
Location	Strong	location_id	location_name, location_type
CargoItem	Strong	cargo_id	cargo_type, weight
InventoryRecord	Weak (on Warehouse + CargoItem)	(cargo_id, warehouse_id)	quantity_stored, last_updated
Carrier	Strong	carrier_id	name, carrier_type
Route	Strong	route_id	estimated_time
Shipment	Strong	shipment_id	shipment_date, status, carrier_id, route_id, origin_loc_id, destination_loc_id
Container	Strong	container_id	container_type, max_capacity, shipment_id (FK, nullable)
ContainerCargo	Associative	(cargo_id, container_id)	cargo_id (FK), container_id (FK)
AuditLog	Strong (added)	id	user_id, action_type, entity_type, entity_id, payload, created_at
AdminAccessRequest	Strong (added)	request_id	user_id, reason, status, requested_at, resolved_at, resolved_by_user_id

4.4 Conversion Decisions

The following decisions were made during the EERD-to-RM conversion:

Disjoint specialization (ISA): We used the "separate tables" approach for the User ISA hierarchy. The admins table has user_id as both its PK and an FK referencing users.user_id. Same for warehouse_staff. This preserves the disjoint constraint at the application level (a user is inserted into exactly one specialization table) and allows each specialization to have its own attributes without NULL columns.

Weak entity (InventoryRecord): InventoryRecord is identified by the composite key (cargo_id, warehouse_id). Both columns are foreign keys referencing their respective parent tables with ON DELETE RESTRICT, preventing deletion of a cargo item or warehouse that still has inventory records.

M:N relationship (CONTAINS): The many-to-many relationship between Container and CargoItem was converted to the container_cargo junction table with a composite PK (cargo_id, container_id).

M:N relationship (MANAGES): The many-to-many relationship between Admin and Warehouse was converted to the manages table with a composite PK (admin_user_id, warehouse_id).

1:N relationships: All 1:N relationships were implemented by placing the FK on the "many" side. For example, warehouses.location_id references locations.location_id, and shipments.carrier_id references carriers.carrier_id.

Dual FK to same table (DELIVERED TO/FROM): The Shipment entity has two FK columns referencing Location: origin_loc_id and destination_loc_id. Both use ON DELETE RESTRICT.

Optional participation (ASSIGNED): The Container entity has a nullable shipment_id FK. Containers can exist without being assigned to a shipment (NULL means "in yard, unassigned").

5. Supporting UML and Architecture Diagrams

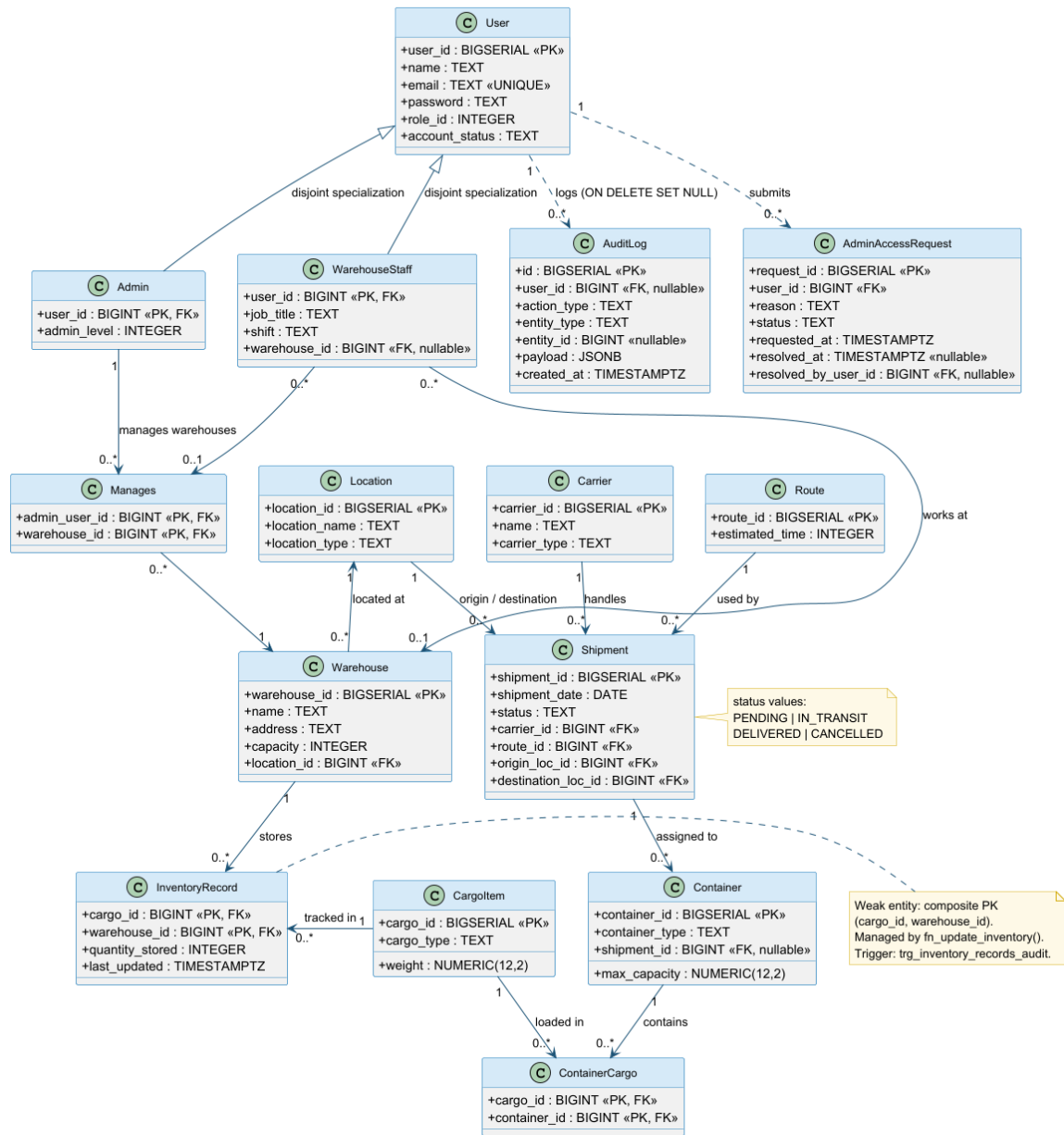
In addition to the EERD and Relational Model Diagram in the previous two sections, the project includes seven supporting PlantUML diagrams that describe the implemented system from complementary perspectives. They are stored under docs/plantuml/ alongside the rendered PNGs that are reproduced below.

5.1 Complete Database Schema

The complete database schema diagram is a PlantUML class diagram that mirrors the implemented database. It includes every table, its columns with data types, primary keys, and foreign key relationships with multiplicities annotated on each connector.

In addition to the table definitions, the diagram highlights the weak entity InventoryRecord with its composite primary key (cargo_id, warehouse_id), documents the stored function fn_update_inventory() that manages safe inventory adjustments, and shows the trigger trg_inventory_records_audit that fires on inventory changes. The permitted shipment status values (PENDING, IN_TRANSIT, DELIVERED, CANCELLED) are also listed as notes on the Shipment entity.

Freight Cargo Inventory and Shipment Management System
Complete Database Schema Diagram (Final)

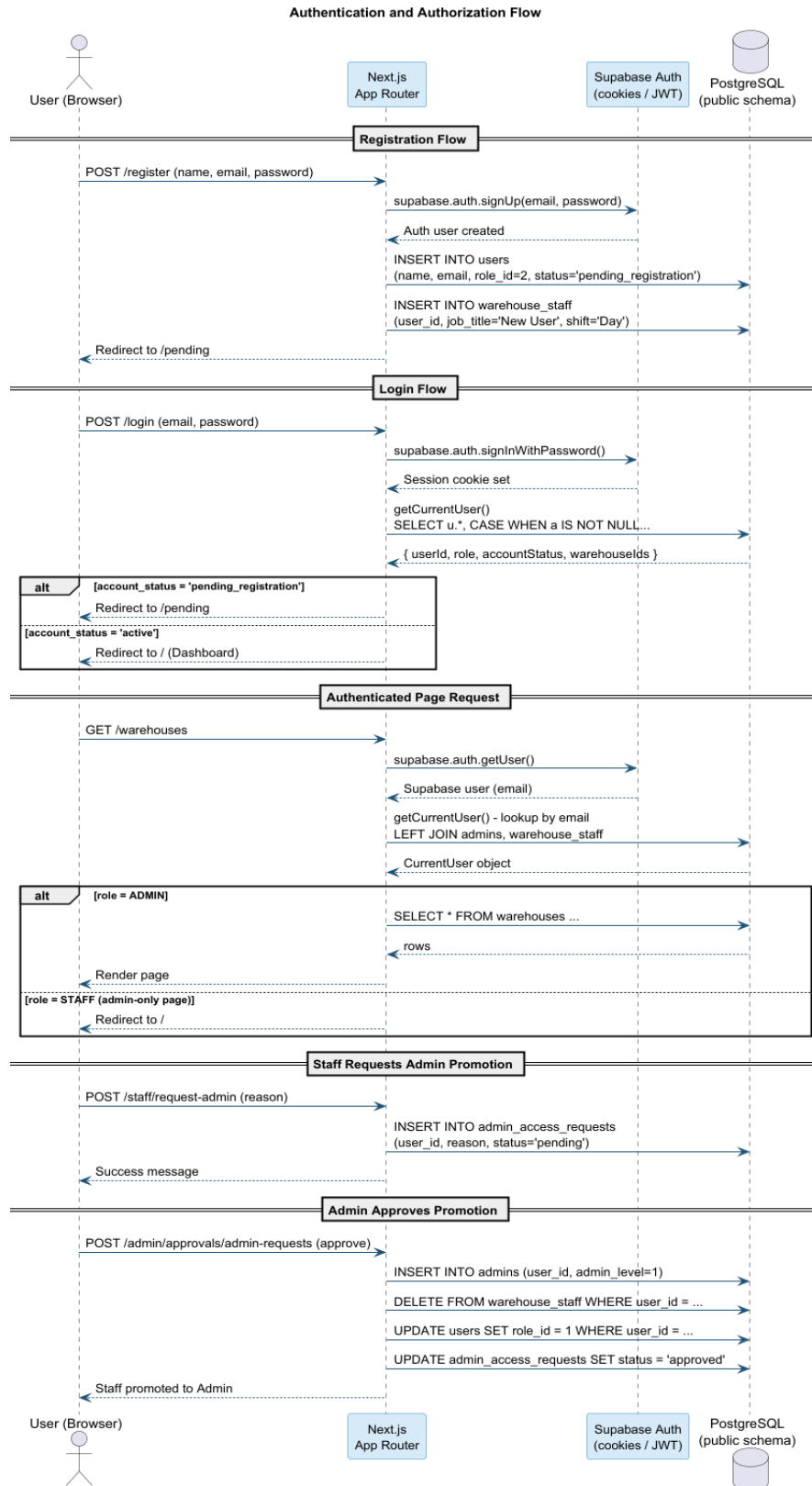


(can be better viewed in the source code folder)

5.2 Authentication and Authorization Flow

This sequence diagram documents the complete authentication and authorization workflow that is enforced on every request.

It covers registration (user self-registers, a Supabase Auth account is created, and an application user row is inserted with a pending registration status), login (Supabase verifies credentials, `getCurrentUser()` resolves the user's role through SQL JOINS, and the user is routed based on their account status), authenticated page requests (how each page load checks authentication and enforces role-based restrictions), the staff-to-admin promotion request, and the admin approval workflow with the SQL operations that convert a staff user into an admin.

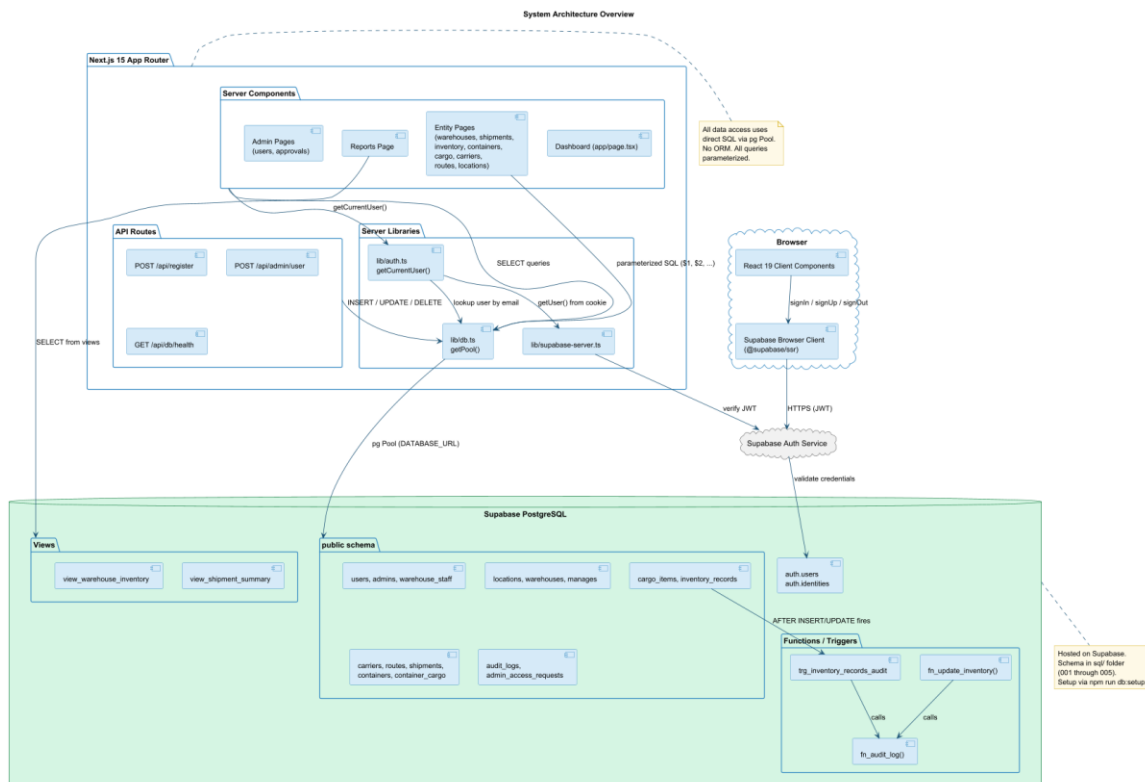


(can be better viewed in the source code folder)

5.3 System Architecture

The architecture overview is a component diagram showing the three-tier structure of the running system.

The browser layer contains the React client components and the Supabase browser client used for authentication. The server layer is a Next.js App Router process running server components, API routes, and the shared library modules (auth.ts for role resolution, db.ts for the pg connection pool, and supabase-server.ts for server-side auth cookies). The database layer is a Supabase-hosted PostgreSQL instance containing the public schema tables, the auth schema used by Supabase Auth, and the stored functions, triggers, and views defined in our SQL migrations. The diagram also highlights three key architectural properties: all data access uses direct SQL via a pg Pool (no ORM), every query is parameterized to prevent SQL injection, and role resolution is performed with SQL JOINs against our own tables rather than relying on Supabase metadata.

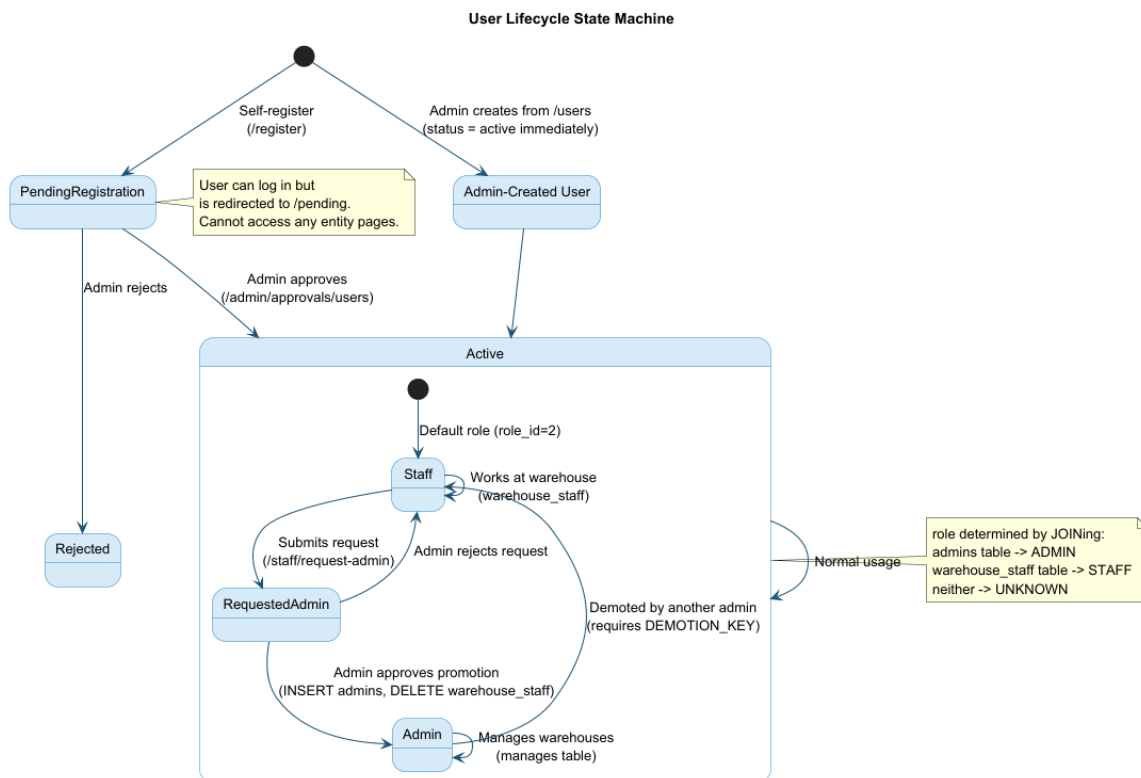


(can be better viewed in the source code folder)

5.4 User Lifecycle State Machine

This state diagram captures every state a user account can be in and the transitions between them.

The states are Pending Registration (the initial state after self-registration), Active Staff (the default active role after approval), Active Admin (either promoted from staff or created directly by an admin), Requested Admin (a staff user who has submitted a promotion request), and Rejected (a registration that was rejected by an admin). The transitions include registration approval and rejection, submission of an admin promotion request, approval and rejection of that request, and admin demotion back to staff (which requires the DEMOTION_KEY environment variable).

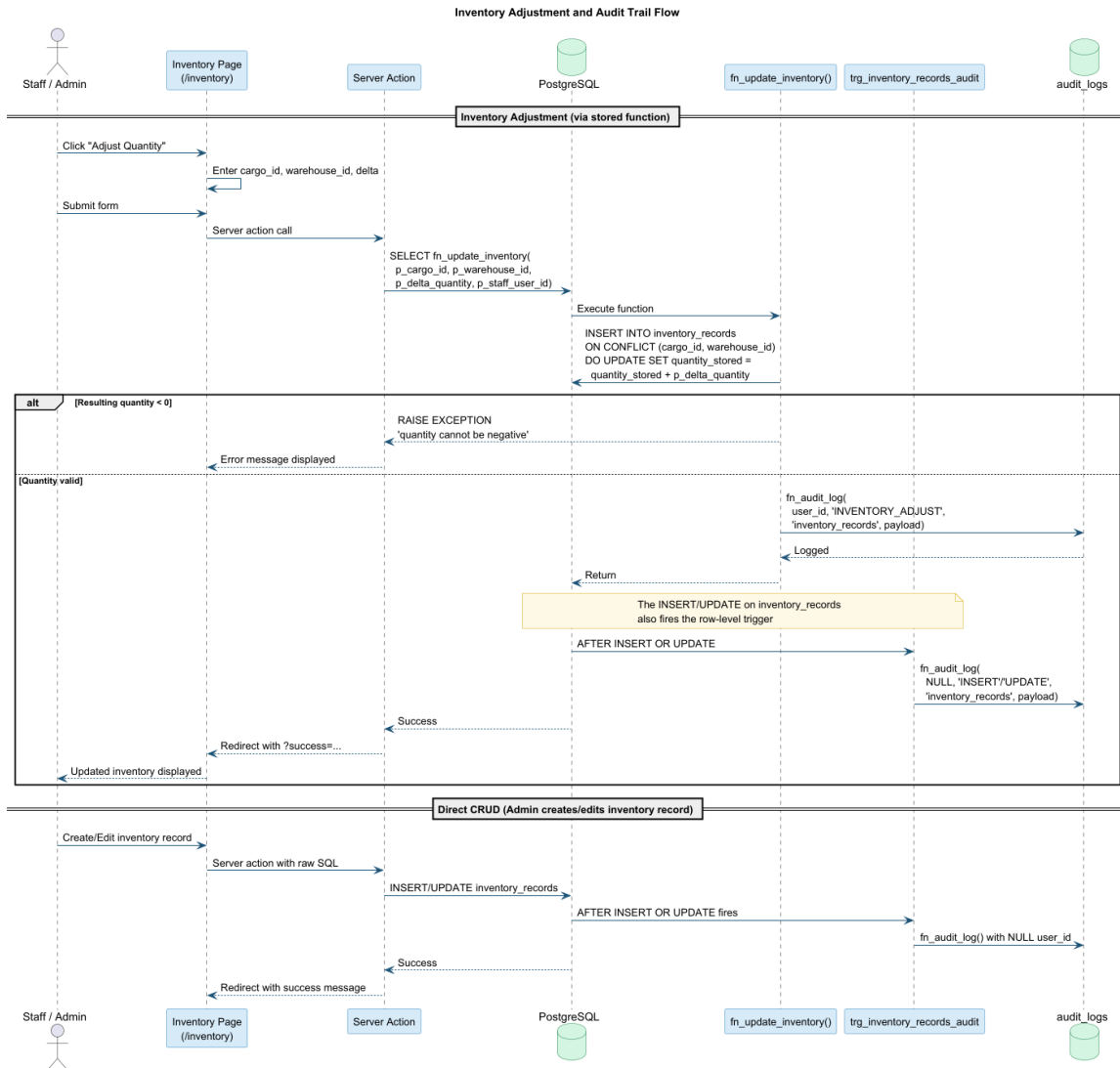


(can be better viewed in the source code folder)

5.5 Inventory Adjustment and Audit Trail

This sequence diagram details what happens end to end when a user adjusts an inventory quantity on the inventory page.

The user action triggers a server action that calls the stored function `fn_update_inventory()`. That function performs an UPSERT on `inventory_records` using `ON CONFLICT`, raises an exception if the adjustment would make the quantity negative, and on success calls `fn_audit_log()` to record the change. In parallel, the trigger `trg_inventory_records_audit` also fires on the INSERT or UPDATE and writes its own entry, which is why two audit log rows are produced per adjustment.

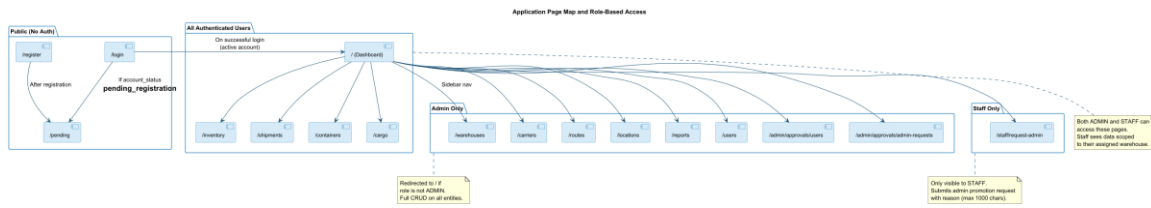


(can be better viewed in the source code folder)

5.6 Application Page Map

The page navigation diagram is a component diagram that lays out every page in the application grouped by access level.

Public pages include /login, /register, and /pending. Pages accessible to any authenticated user include the dashboard at /, and the Inventory, Shipments, Containers, and Cargo pages. Admin-only pages are Warehouses, Carriers, Routes, Locations, Reports, Users, and the two approval queues under /admin/approvals. The single staff-only page is /staff/request-admin, where a staff user submits an admin promotion request.

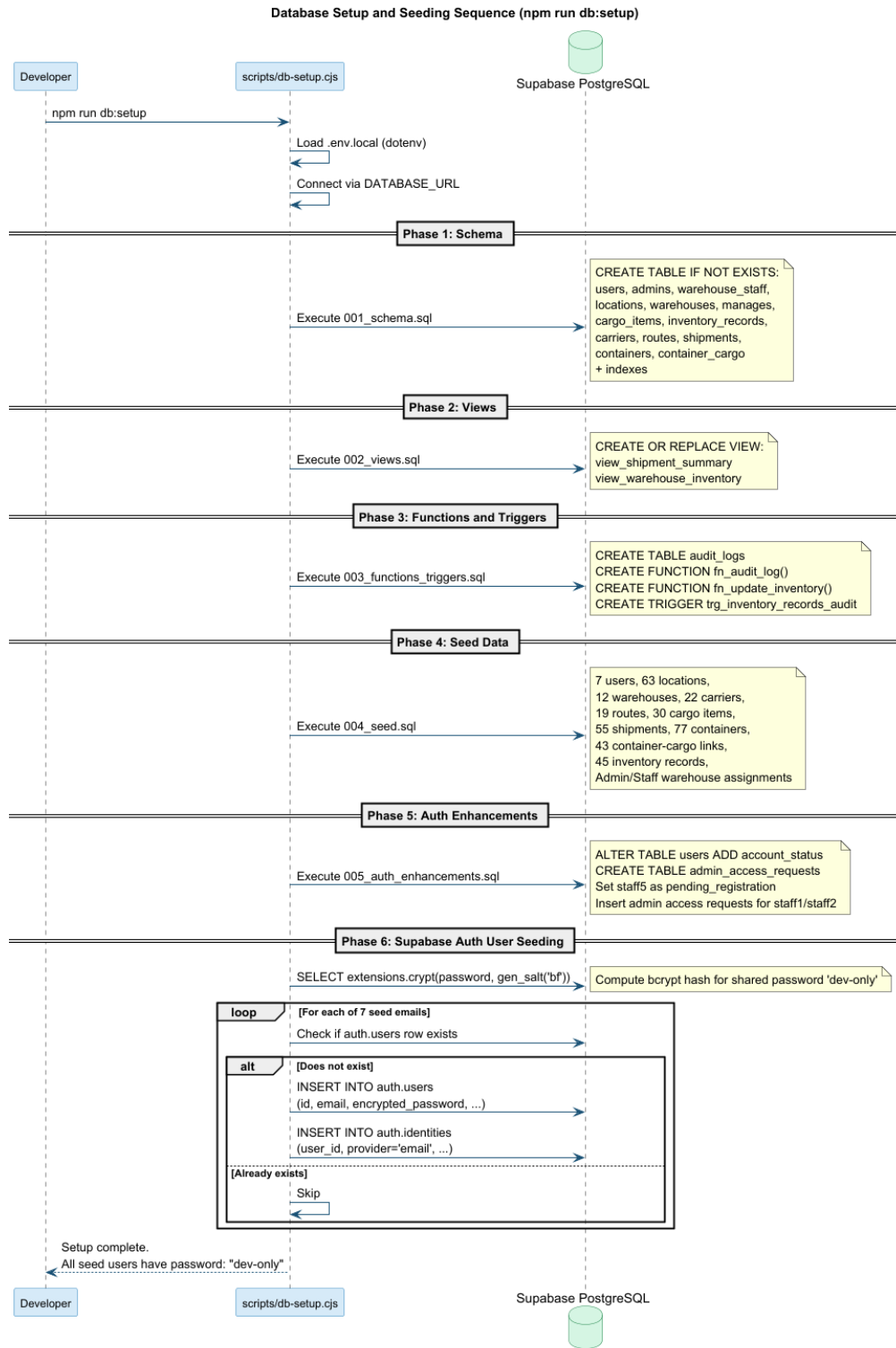


(can be better viewed in the source code folder)

5.7 Database Setup Sequence

This sequence diagram shows what happens when a developer runs `npm run db:setup` to provision a fresh database.

The setup script loads environment variables from `.env.local`, executes the SQL migration files in order (001 through 005), and then creates the Supabase Auth accounts by issuing direct SQL INSERT statements against `auth.users` and `auth.identities`. Passwords are hashed with `bcrypt` via the `pgcrypto` extension (`extensions.crypt`). A key detail captured by the diagram is that every token column on `auth.users` must be set to an empty string rather than `NULL`, otherwise Supabase GoTrue rejects the login with a schema error.



(can be better viewed in the source code folder)

6. Implementation

6.1 DBMS Selection

We selected **PostgreSQL** as our DBMS, hosted on **Supabase**. The reasons for this choice:

- 22. PostgreSQL supports all required features: stored functions (PL/pgSQL), triggers, views, JSONB columns, and CHECK-like constraints via RAISE EXCEPTION in functions
- 23. Supabase provides a free-tier hosted PostgreSQL instance with a web-based SQL editor, making it easy for people to inspect the database directly
- 24. Supabase also provides built-in authentication (Supabase Auth) that integrates well with Next.js via the @supabase/ssr package
- 25. The pg Node.js library provides direct SQL access with parameterized queries, which is ideal for a course focused on SQL rather than ORMs

6.2 SQL Statements for Key Transactions

Below are the SQL statements for the primary transactions implemented in the system.

Transaction 1: User Registration

```
-- Step 1: Create Supabase Auth account (via supabase.auth.signUp in application code)
-- Step 2: Insert application user
INSERT INTO users (user_id, name, email, password, role_id, account_status)
VALUES ((SELECT COALESCE(MAX(user_id), 0) + 1 FROM users), $1, $2, 'dev-only',
2, 'pending_registration');

-- Step 3: Insert warehouse_staff specialization
INSERT INTO warehouse_staff (user_id, job_title, shift)
VALUES ($1, 'New User', 'Day');
```

Transaction 2: Admin Approves User Registration

```
UPDATE users SET account_status = 'active' WHERE user_id = $1;
```

Transaction 3: Create Shipment

```
INSERT INTO shipments (shipment_id, shipment_date, status, carrier_id, route_id,
origin_loc_id, destination_loc_id)
VALUES ((SELECT COALESCE(MAX(shipment_id), 0) + 1 FROM shipments), $1, $2, $3,
$4, $5, $6);
```

Transaction 4: Assign Container to Shipment

```
UPDATE containers SET shipment_id = $1 WHERE container_id = $2;
```

Transaction 5: Adjust Inventory (via stored function)

```
SELECT fn_update_inventory($1, $2, $3, $4);
-- Parameters: cargo_id, warehouse_id, delta_quantity, staff_user_id
-- Function performs UPSERT with ON CONFLICT, checks for negative quantity,
-- and logs to audit_logs via fn_audit_log()
```

Transaction 6: Create Warehouse

```
INSERT INTO warehouses (warehouse_id, name, address, capacity, location_id)
VALUES ((SELECT COALESCE(MAX(warehouse_id), 0) + 1 FROM warehouses), $1, $2, $3, $4);
```

Transaction 7: Delete Entity (with FK check)

```
-- Example: Delete carrier (checks for referencing shipments first)
-- Application code first checks:
SELECT COUNT(*) FROM shipments WHERE carrier_id = $1;
-- If count > 0, deletion is blocked with an error message
-- If count = 0:
DELETE FROM carriers WHERE carrier_id = $1;
```

Transaction 8: Promote Staff to Admin

```
-- Step 1: Insert into admins specialization
INSERT INTO admins (user_id, admin_level) VALUES ($1, 1);
-- Step 2: Remove from warehouse_staff specialization
DELETE FROM warehouse_staff WHERE user_id = $1;
-- Step 3: Update role_id
UPDATE users SET role_id = 1 WHERE user_id = $1;
-- Step 4: Mark request as approved
UPDATE admin_access_requests SET status = 'approved', resolved_at = NOW(),
resolved_by_user_id = $2 WHERE request_id = $3;
```

Transaction 9: Demote Admin to Staff

```
-- Step 1: Remove from admins
DELETE FROM admins WHERE user_id = $1;
-- Step 2: Remove all manages entries
DELETE FROM manages WHERE admin_user_id = $1;
-- Step 3: Insert into warehouse_staff
INSERT INTO warehouse_staff (user_id, job_title, shift) VALUES ($1, 'Staff', 'Day');
-- Step 4: Update role_id
UPDATE users SET role_id = 2 WHERE user_id = $1;
```

Transaction 10: Get Current User Role (getCurrentUser)

```
SELECT u.user_id, u.name, u.account_status,
CASE
  WHEN a.user_id IS NOT NULL THEN 'ADMIN'
  WHEN ws.user_id IS NOT NULL THEN 'STAFF'
  ELSE 'UNKNOWN'
END AS role
FROM users u
LEFT JOIN admins a ON a.user_id = u.user_id
LEFT JOIN warehouse_staff ws ON ws.user_id = u.user_id
WHERE u.email = $1
LIMIT 1;
```

Transaction 11: Dashboard KPIs (personalized)

```
-- Shipment counts by status for user's warehouses
SELECT
  COUNT(*) FILTER (WHERE s.status = 'IN_TRANSIT') AS in_transit,
  COUNT(*) FILTER (WHERE s.status = 'PENDING') AS pending,
  COUNT(*) FILTER (WHERE s.status = 'DELIVERED') AS delivered
FROM shipments s
WHERE s.origin_loc_id IN (SELECT location_id FROM warehouses WHERE warehouse_id
= ANY($1))
   OR s.destination_loc_id IN (SELECT location_id FROM warehouses WHERE
warehouse_id = ANY($1));
```

Transaction 12: Reports - View Queries

```
-- Warehouse inventory summary
SELECT * FROM view_warehouse_inventory ORDER BY warehouse_name;

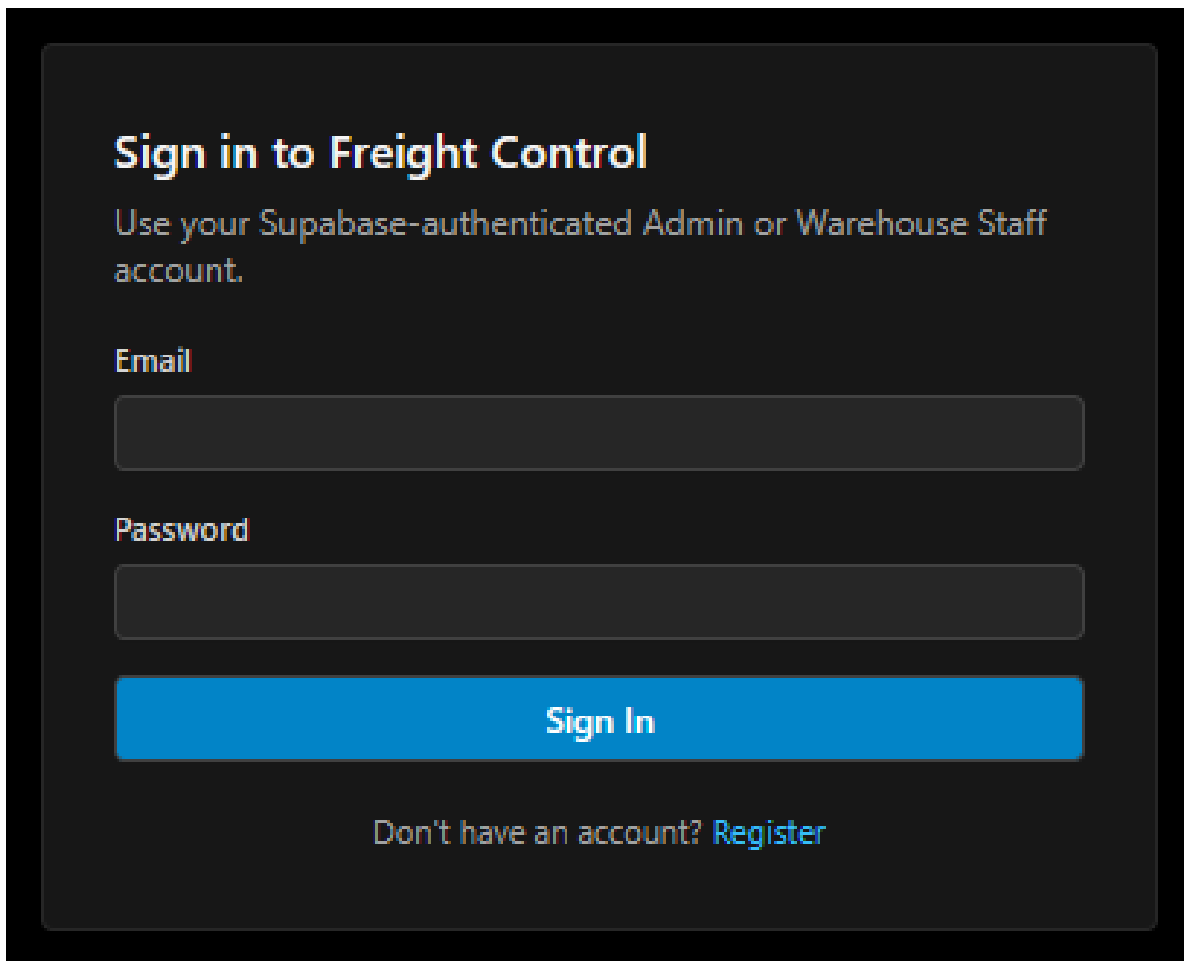
-- Shipment summary
SELECT * FROM view_shipment_summary ORDER BY shipment_date DESC LIMIT 500;
```

7. Visual Interface Description

The application features a modern, dark-mode-first design built with Tailwind CSS. The interface is organized around a persistent sidebar navigation and a top bar showing the current user's email and a sign-out button.

7.1 Sign In Page

The login page presents a clean form with email and password fields. Authentication is handled by Supabase Auth. On successful login, the user is redirected to the dashboard (if active) or the pending page (if registration is pending).

A dark-themed sign-in form for 'Freight Control'. The form is centered on a black background. It features a title 'Sign in to Freight Control' in white, followed by a subtitle 'Use your Supabase-authenticated Admin or Warehouse Staff account.' in a lighter gray. Below the subtitle are two input fields: 'Email' and 'Password', both with light gray borders. A prominent blue 'Sign In' button is positioned below the password field. At the bottom, there is a link 'Don't have an account? Register' in a light blue color.

Sign in to Freight Control

Use your Supabase-authenticated Admin or Warehouse Staff account.

Email

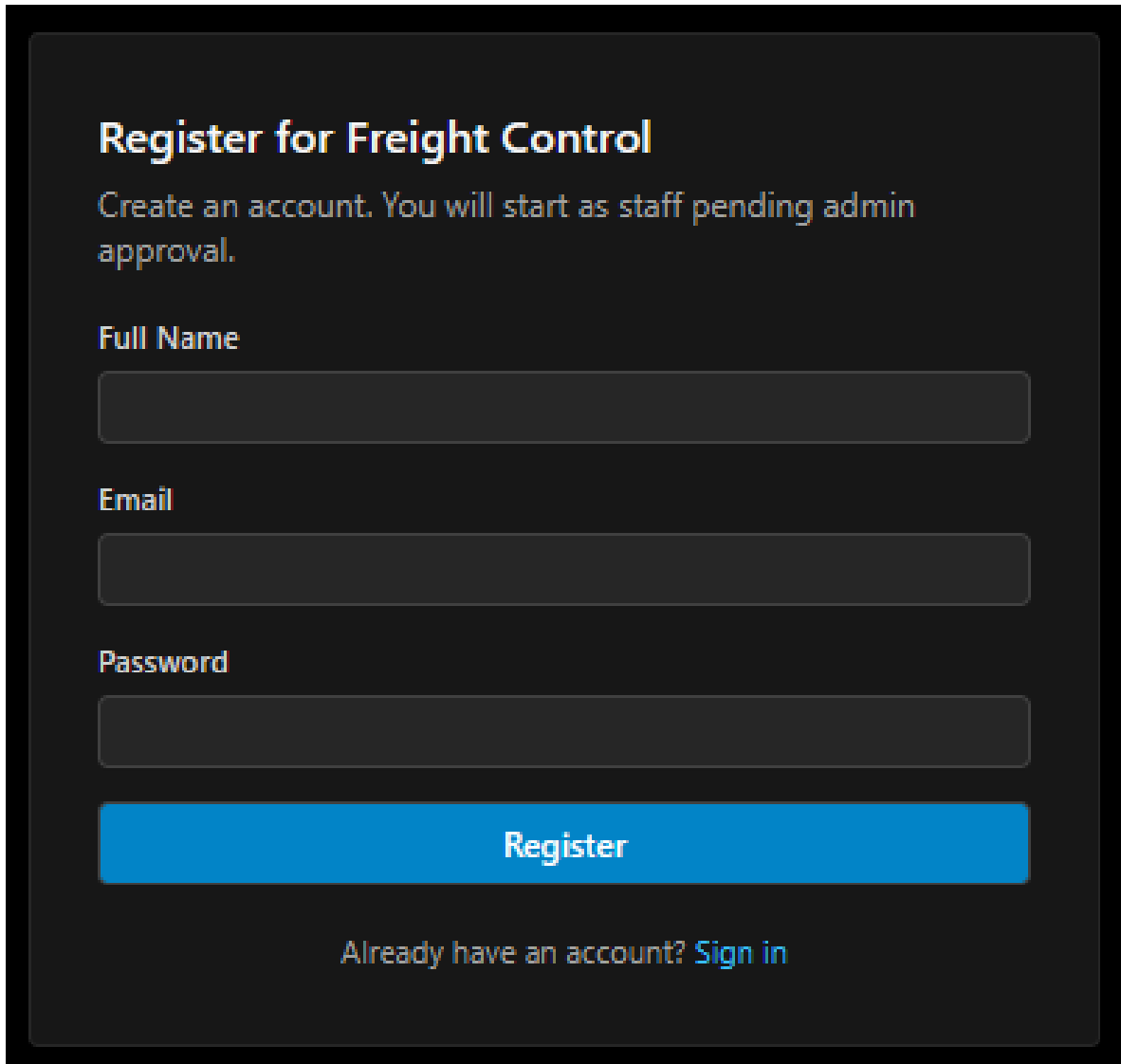
Password

Sign In

Don't have an account? [Register](#)

7.2 Registration Page

New users can self-register by providing their name, email, and password. The system creates both a Supabase Auth account and an application user with `pending_registration` status. The user is then shown a confirmation and redirected to the pending page.

The image shows a registration form titled "Register for Freight Control". The form is set against a dark background with light-colored text and input fields. It includes fields for "Full Name", "Email", and "Password", each with a corresponding input box. Below these fields is a prominent blue "Register" button. At the bottom, there is a link that says "Already have an account? Sign in".

Register for Freight Control

Create an account. You will start as staff pending admin approval.

Full Name

Email

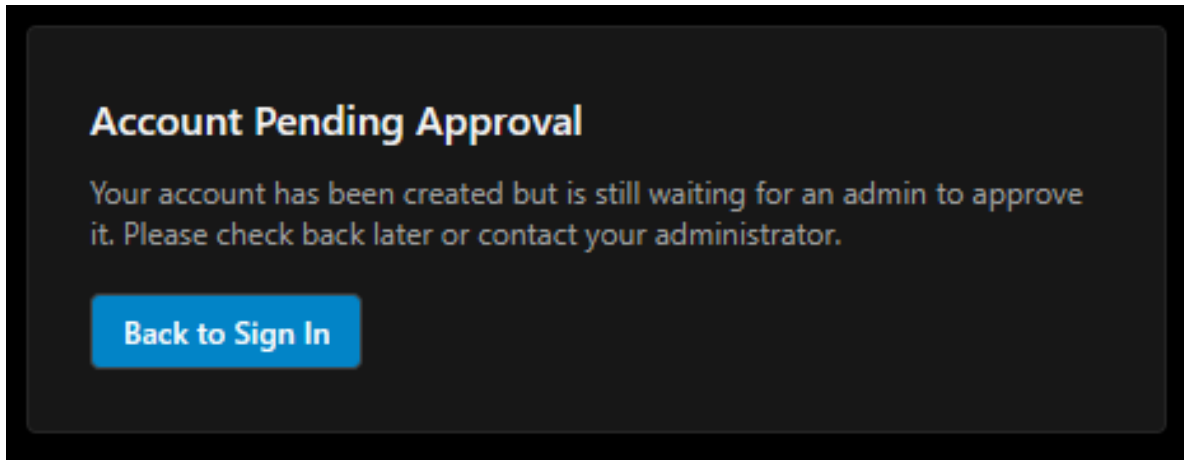
Password

Register

Already have an account? [Sign in](#)

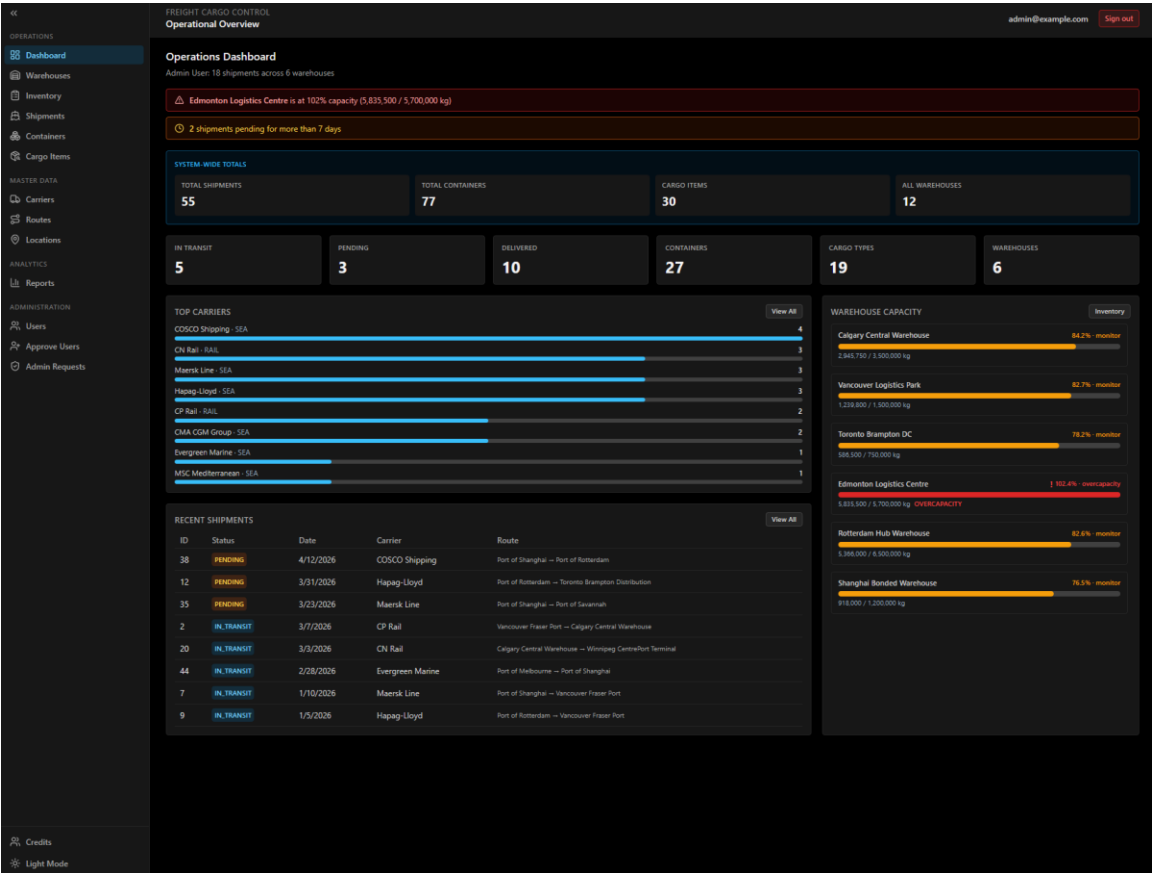
7.3 Pending Page

After registration, users who have not yet been approved by an admin are redirected to the pending page. This screen clearly informs them that their account is awaiting approval and that they cannot access the main system until their `account_status` is changed to `active`.



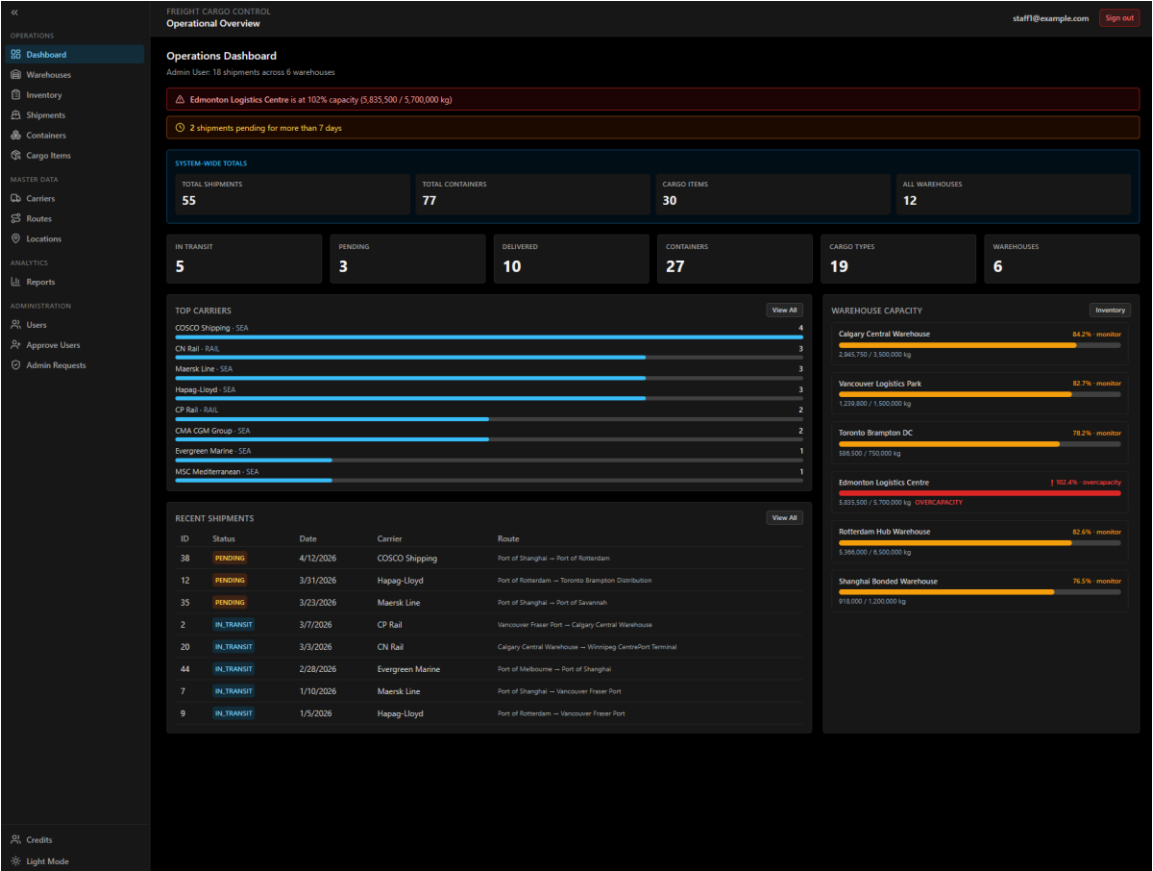
7.4 Admin Dashboard and Sidebar View

The admin dashboard shows personalized KPIs scoped to the admin's assigned warehouses: in-transit, pending, and delivered shipment counts; top carriers by volume; recent shipments; and warehouse capacity utilization bars. The sidebar shows all navigation sections: Operations, Master Data, Analytics, and Administration.



7.5 Staff Dashboard and Sidebar View

The staff dashboard shows the same KPI layout but scoped to the staff member's single assigned warehouse. The sidebar shows a reduced set of navigation items: Operations (minus Warehouses) and a Staff section with "Request Admin."



7.6 Warehouses Page

Displays all warehouses in a table with columns for ID, name, address, capacity, and location. Admins can create new warehouses, edit existing ones, and delete warehouses (with FK constraint checking). The create/edit forms allow selecting a location from a dropdown populated via SQL.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

Night Mode

FREIGHT CARGO CONTROL

Operational Overview

admin@example.com

Sign out

Warehouses

Maintain warehouse master data, locations, and capacity.

CREATE NEW WAREHOUSE

Name

Address

Capacity

Location

Select location...

Save Warehouse

Filter by location: All

Apply

12 records

ID	Name	Address	Capacity	Location ID	Actions
1	Calgary Central Warehouse	2850 Sunridge Blvd NE, Calgary, AB	3500000	1	Delete
2	Vancouver Logistics Park	1255 Commissioner St, Vancouver, BC	1500000	4	Delete
3	Toronto Brampton DC	8400 Lawson Rd, Milton, ON	750000	6	Delete
4	Montreal Dorval Warehouse	6500 Ch. St-François, Dorval, QC	200000	8	Delete
5	Edmonton Logistics Centre	2151 Winterburn Rd NW, Edmonton, AB	5700000	9	Delete
6	Winnipeg Distribution Centre	2000 Berry St, Winnipeg, MB	8500000	12	Delete
7	Ottawa South Warehouse	1150 Leblain Rd, Ottawa, ON	50000	13	Delete
8	Los Angeles Gateway Warehouse	2500 Navy Way, San Pedro, CA 90731, USA	350000	16	Delete
9	Rotterdam Hub Warehouse	Europaweg 300, 3199 LC Rotterdam, NL	6500000	28	Delete
10	Shanghai Bonded Warehouse	189 Yangshupu Rd, Hongkou, Shanghai, CN	1200000	40	Delete
11	Singapore FTZ Warehouse	1 Harbour Front Walk, Singapore 098585	900000	47	Delete
12	Dubai Jebel Ali Warehouse	P.O. Box 17000, Jebel Ali, Dubai, UAE	500000	54	Delete

7.7 Inventory Page

Shows inventory records with cargo ID, warehouse ID, quantity stored, and last updated timestamp. Users can adjust quantities (which calls `fn_update_inventory()` on the server), create new inventory records, and delete existing ones.

FREIGHT CONTROL

OPERATIONS

WAREHOUSES

MASTER DATA

ANALYTICS

ADMINISTRATION

Dashboard

Inventory

Shipments

Containers

Cargo Items

Carriers

Routes

Locations

Reports

Users

Approve Users

Admin Requests

Night Mode

FREIGHT CONTROL

Operational Overview

admin@example.com

Sign out

Inventory Records

View and maintain inventory records per warehouse and cargo item.

ADJUST INVENTORY QUANTITY

Provide a positive or negative quantity change.

Cargo ID

Warehouse ID

Delta Quantity

Apply Adjustment

MOVE CARGO TO CONTAINER

Link a cargo item to a container with capacity checks.

Cargo ID

Container ID

Move Cargo

Cargo ID	Warehouse ID	Quantity Stored	Last Updated
1	1	45	2026-03-15, 4:30:00 a.m.
5	1	120	2026-03-14, 2:00:00 a.m.
6	1	80	2026-03-10, 5:45:00 a.m.
7	1	35	2026-03-12, 8:20:00 a.m.
18	1	60	2026-03-08, 3:15:00 a.m.
1	2	200	2026-03-16, 1:00:00 a.m.
5	2	300	2026-03-15, 10:30:00 a.m.
10	2	90	2026-03-16, 12:00:00 a.m.
22	2	40	2026-03-13, 4:45:00 a.m.
28	2	85	2026-03-14, 7:00:00 a.m.
1	3	150	2026-03-14, 8:30:00 a.m.
2	3	65	2026-03-15, 3:00:00 a.m.
3	3	70	2026-03-12, 5:00:00 a.m.
15	3	8	2026-03-10, 10:00:00 a.m.
17	3	55	2026-03-11, 2:30:00 a.m.
9	4	80	2026-03-14, 11:30:00 p.m.
14	4	25	2026-03-12, 4:00:00 a.m.
23	4	30	2026-03-13, 3:15:00 a.m.
30	4	45	2026-03-14, 12:00:00 a.m.
6	5	150	2026-03-13, 6:00:00 a.m.
7	5	200	2026-03-14, 1:00:00 a.m.
19	5	90	2026-03-11, 8:00:00 a.m.
5	6	75	2026-03-14, 4:30:00 a.m.
6	6	250	2026-03-15, 2:00:00 a.m.

7.8 Shipments Page

Lists all shipments with ID, date, status, carrier, route, origin, and destination. Admins can create new shipments by selecting carriers, routes, and locations from dropdowns. Status updates (PENDING, IN_TRANSIT, DELIVERED, CANCELLED) are done through edit forms.

Freight Control

Freight Cargo Control

Operational Overview

admin@example.com

Sign out

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

Shipments

Manage freight shipments, assigned routes, and carriers.

CREATE NEW SHIPMENT

Shipment Date

yyyy-mm-dd

Status

Select status...

Carrier

Select carrier...

Route (est. days)

Select route...

Origin

Select origin...

Destination

Select destination...

Save Shipment

Status

All

Apply

33 records

ID	Status	Date	Carrier	Est. Days	Origin	Destination
1	DELIVERED	2025-12-05	J.B. Hunt Transport	1	Edmonton Logistics Centre	Calgary Central Warehouse
2	IN_TRANSIT	2026-03-07	CP Rail	2	Vancouver Fraser Port	Calgary Central Warehouse
3	PENDING	2026-03-27	Evergreen Marine	10	Port of Busan	Vancouver Fraser Port
4	DELIVERED	2025-10-15	ONE Ocean Network Express	10	Port of Busan	Vancouver Fraser Port
5	IN_TRANSIT	2026-02-08	Yang Ming Marine	10	Port of Tokyo	Vancouver Fraser Port
6	DELIVERED	2025-10-25	Yang Ming Marine	10	Port of Tokyo	Vancouver Fraser Port
7	IN_TRANSIT	2026-01-10	Maersk Line	14	Port of Shanghai	Vancouver Fraser Port
8	DELIVERED	2025-10-05	Maersk Line	14	Port of Shanghai	Vancouver Fraser Port
9	IN_TRANSIT	2026-01-05	Hapag-Lloyd	24	Port of Rotterdam	Vancouver Fraser Port
10	DELIVERED	2025-11-30	CN Rail	2	Calgary Central Warehouse	Vancouver Fraser Port
11	DELIVERED	2025-12-14	Air Canada Cargo	1	Calgary Intermodal Terminal	Toronto Pearson Cargo Hub
12	PENDING	2026-03-31	Hapag-Lloyd	16	Port of Rotterdam	Toronto Brampton Distribution
13	DELIVERED	2025-12-08	CN Rail	3	Winnipeg CentrePort Terminal	Toronto Brampton Distribution
14	DELIVERED	2025-12-20	CMA CGM Group	14	Port of Barcelona	Port of Montreal
15	DELIVERED	2025-10-18	Hapag-Lloyd	12	Port of Rotterdam	Port of Montreal
16	DELIVERED	2025-12-11	CP Rail	2	Port of Halifax	Port of Montreal
17	DELIVERED	2025-12-02	CP Rail	1	Toronto Brampton Distribution	Port of Montreal
18	IN_TRANSIT	2026-01-25	MSC Mediterranean	10	Port of Hamburg	Port of Halifax
19	DELIVERED	2025-10-22	MSC Mediterranean	10	Port of Hamburg	Port of Halifax
20	IN_TRANSIT	2026-03-03	CN Rail	2	Calgary Central Warehouse	Winnipeg CentrePort Terminal
21	DELIVERED	2025-11-22	Yang Ming Marine	14	Kaohsiung Port, Taiwan	Port of Los Angeles
22	IN_TRANSIT	2026-01-15	Evergreen Marine	12	Port of Busan	Port of Los Angeles
23	CANCELLED	2025-11-08	COSCO Shipping	14	Port of Shanghai	Port of Los Angeles
24	DELIVERED	2025-10-08	COSCO Shipping	12	Port of Shanghai	Port of Los Angeles

7.9 Containers Page

Displays all containers with their type, max capacity, and assigned shipment. Admins can create new containers, assign/unassign them to shipments, and manage container-cargo links.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

Night Mode

FREIGHT CARGO CONTROL

Operational Overview

admin@example.com

Sign out

Containers

Track container assignments and shipment allocation.

ASSIGN CONTAINER TO SHIPMENT

Container ID

Shipment ID

Assign

Container ID	Type	Max Capacity (kg)	Shipment ID
1	40ft Standard Dry	26780.00	8
2	40ft High Cube Dry	26460.00	8
3	40ft Standard Dry	26780.00	24
4	20ft Standard Dry	21770.00	24
5	40ft High Cube Dry	26460.00	26
6	40ft Standard Dry	26780.00	26
7	40ft High Cube Dry	26460.00	4
8	40ft Standard Dry	26780.00	15
9	20ft Refrigerated	21250.00	15
10	40ft Standard Dry	26780.00	19
11	40ft High Cube Dry	26460.00	6
12	20ft Standard Dry	21770.00	6
13	40ft Standard Dry	26780.00	34
14	40ft High Cube Dry	26460.00	34
15	20ft Standard Dry	21770.00	52
16	40ft Standard Dry	26780.00	37
17	40ft High Cube Dry	26460.00	37
18	20ft Standard Dry	21770.00	37
19	40ft Flat Rack	39200.00	29
20	40ft Standard Dry	26780.00	36
21	20ft Open Top	21600.00	36
22	40ft Standard Dry	26780.00	46
23	40ft High Cube Dry	26460.00	21
24	40ft Standard Dry	26780.00	21
25	20ft Standard Dry	21770.00	45
26	40ft Standard Dry	26780.00	10
27	40ft Standard Dry	26780.00	30
28	40ft High Cube Dry	26460.00	30
29	40ft Flat Rack	39200.00	14
30	40ft Standard Dry	26780.00	14

7.10 Cargo Items Page

Lists all cargo item types with their ID, type name, and unit weight. Admins and staff can create new cargo items and manage existing ones.

Freight Control

Operations

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

Master Data

Carriers

Routes

Locations

Analytics

Reports

Administration

Users

Approve Users

Admin Requests

Night Mode

Freight Cargo Control

Operational Overview

admin@example.com

Sign out

Cargo Items

Master data for cargo items, types, and per-unit weights.

CREATE NEW CARGO ITEM

Optional: add an initial inventory record for a warehouse.

Cargo Type

Weight (kg)

Initial Warehouse ID (optional)

Initial Quantity (optional)

Save Cargo Item

Cargo ID	Cargo Type	Weight (kg)
1	Consumer Electronics Pallets	850.00
2	Automotive Engine Blocks	2400.00
3	Pharmaceutical Crates	320.00
4	Raw Cotton Bales	1800.00
5	Softwood Lumber Bundle	3200.00
6	Canadian Hard Red Wheat	27000.00
7	Crude Oil Drum Pallet	8500.00
8	Hot-Rolled Steel Coils	22000.00
9	Frozen Atlantic Salmon	1200.00
10	Fresh BC Blueberries	450.00
11	CNC Milling Machine	4800.00
12	Industrial Solvent Drums	1500.00
13	Flat-Pack Furniture Sets	680.00
14	Okanagan Ice Wine Cases	280.00
15	MRI Scanner Unit	6200.00
16	Copper Wire Spools	3600.00
17	Newspprint Paper Rolls	4200.00
18	All-Season Tire Pallets	1100.00
19	Polyethylene Pellets	950.00
20	Colombian Coffee Sacks	2100.00
21	Indian Basmati Rice	1600.00
22	Japanese Auto Parts Crate	1350.00
23	German Optical Instruments	420.00
24	Chilean Copper Cathodes	18000.00
25	Australian Merino Wool	2800.00
26	Italian Carrara Marble	14000.00
27	Thai Natural Rubber Bales	3400.00
28	Korean Semiconductor Wafers	180.00
29	Brazilian Soybean Bulk	25000.00

7.11 Carriers Page (Admin Only)

Displays all carriers with ID, name, and type (AIR, SEA, RAIL, ROAD). Admins can create, edit, and delete carriers. Deletion is blocked if the carrier has associated shipments.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

FREIGHT CARGO CONTROL

Operational Overview

admin@example.comSign out

Carriers

Maintain master data for external carriers.

CREATE NEW CARRIER

Name

Carrier Type

Select type...

Save Carrier

ID	Name	Type	Actions
1	Maersk Line	SEA	Delete
2	MSC Mediterranean	SEA	Delete
3	CMA CGM Group	SEA	Delete
4	Hapag-Lloyd	SEA	Delete
5	COSCO Shipping	SEA	Delete
6	Evergreen Marine	SEA	Delete
7	ONE Ocean Network Express	SEA	Delete
8	Yang Ming Marine	SEA	Delete
9	FedEx Express	AIR	Delete
10	UPS Airlines	AIR	Delete
11	DHL Aviation	AIR	Delete
12	Air Canada Cargo	AIR	Delete
13	Emirates SkyCargo	AIR	Delete
14	Cathay Pacific Cargo	AIR	Delete
15	CN Rail	RAIL	Delete
16	CP Rail	RAIL	Delete
17	BNSF Railway	RAIL	Delete
18	Union Pacific Railroad	RAIL	Delete
19	J.B. Hunt Transport	ROAD	Delete
20	Schneider National	ROAD	Delete
21	XPO Logistics	ROAD	Delete
22	Transforce International	ROAD	Delete

Night Mode

7.12 Routes Page (Admin Only)

Shows all routes with their ID and estimated transit time (in days). Admins can create new routes and manage existing ones.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

FREIGHT CARGO CONTROL

Operational Overview

admin@example.com

Sign out

Routes

Define standard shipment routes with estimated transit times.

CREATE NEW ROUTE

Estimated Time (days)

Save Route

Route ID	Est. Time (days)	Actions
1	1	Delete
2	2	Delete
3	3	Delete
4	4	Delete
5	5	Delete
6	7	Delete
7	8	Delete
8	10	Delete
9	12	Delete
10	14	Delete
11	16	Delete
12	18	Delete
13	21	Delete
14	24	Delete
15	28	Delete
16	30	Delete
17	35	Delete
18	40	Delete
19	45	Delete

Night Mode

7.13 Locations Page (Admin Only)

Lists all 63 locations with their ID, name, and type (WAREHOUSE, PORT, RAIL_TERMINAL, DISTRIBUTION, AIRPORT). Admins can create new locations and manage existing ones.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

Night Mode

FREIGHT CONTROL

Operational Overview

admin@example.com

Sign out

Locations

Master data for shipment origin and destination locations.

CREATE NEW LOCATION

Location Name

Location Type

Select type...

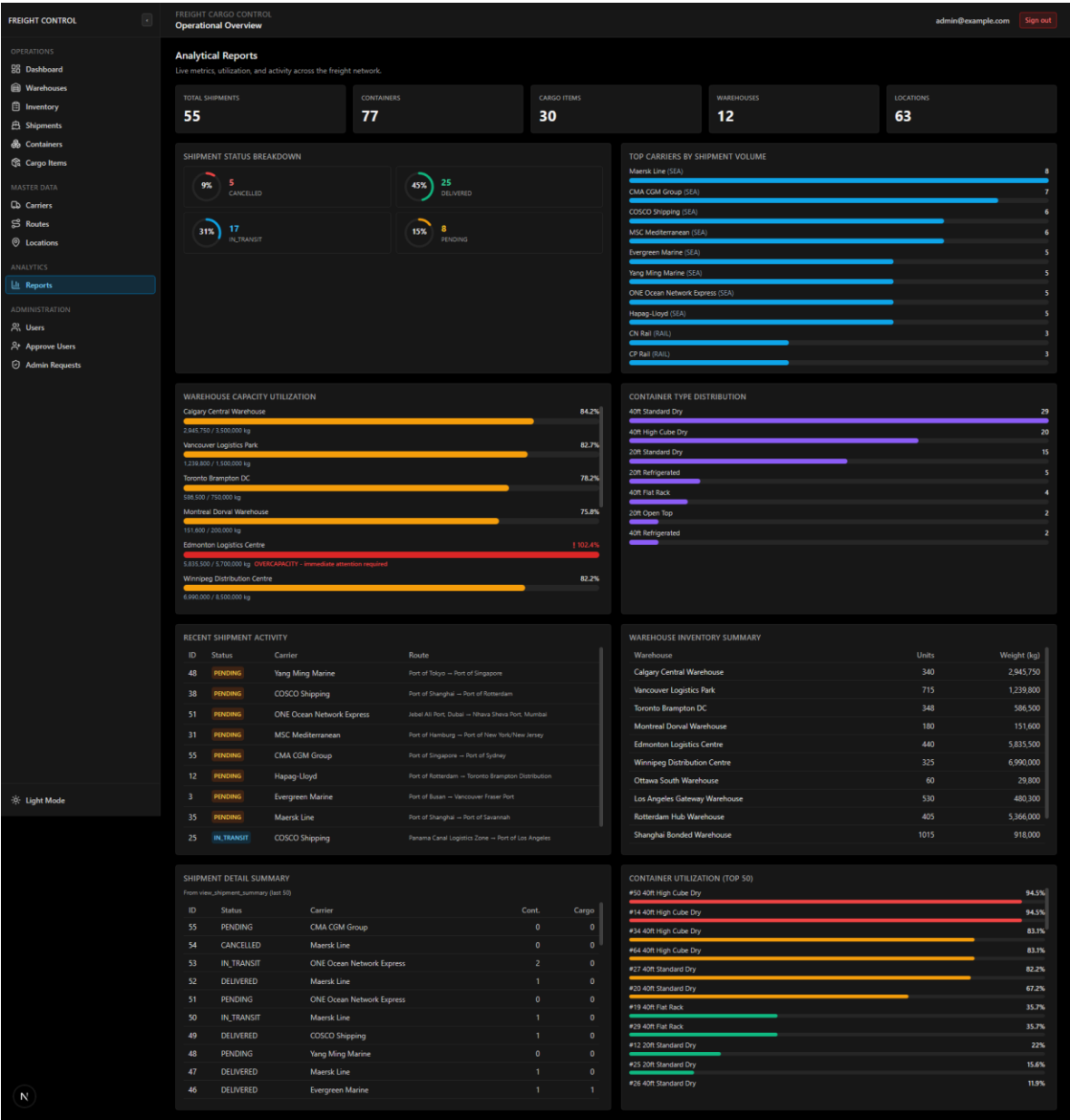
Save Location

ID	Name	Type	Actions
1	Calgary Central Warehouse	WAREHOUSE	Delete
2	Calgary Intermodal Terminal	RAIL_TERMINAL	Delete
3	Vancouver Fraser Port	PORT	Delete
4	Vancouver Logistics Park	WAREHOUSE	Delete
5	Toronto Pearson Cargo Hub	AIRPORT	Delete
6	Toronto Brampton Distribution	DISTRIBUTION	Delete
7	Port of Montreal	PORT	Delete
8	Montreal Dorval Warehouse	WAREHOUSE	Delete
9	Edmonton Logistics Centre	WAREHOUSE	Delete
10	Port of Halifax	PORT	Delete
11	Winnipeg CentrePort Terminal	RAIL_TERMINAL	Delete
12	Winnipeg Distribution Centre	DISTRIBUTION	Delete
13	Ottawa South Distribution	DISTRIBUTION	Delete
14	Port of Prince Rupert	PORT	Delete
15	Hamilton Bayfront Terminal	PORT	Delete
16	Port of Los Angeles	PORT	Delete
17	Port of Long Beach	PORT	Delete
18	Port of New York/New Jersey	PORT	Delete
19	Houston Ship Channel Terminal	PORT	Delete
20	Port of Miami	PORT	Delete
21	Chicago Intermodal Hub	RAIL_TERMINAL	Delete
22	Port of Savannah	PORT	Delete
23	Port of Santos, Brazil	PORT	Delete

7.14 Reports Page (Admin Only)

Displays two summary tables generated from database views:

- **Warehouse Inventory Summary** (from view_warehouse_inventory): warehouse name, total units, total weight
- **Shipment Summary** (from view_shipment_summary): shipment ID, status, date, carrier, container count, cargo item count



7.15 Users Page (Admin Only)

Contains four management panels:

- 26. **Create New User** - form with name, email, and role selection. Creates both Supabase Auth and DB user.
- 27. **Update User** - load by ID, edit name/email, option to promote to admin.
- 28. **Assign User to Warehouse** - works for both admin (inserts into manages) and staff (updates warehouse_staff.warehouse_id).
- 29. **Demote Admin to Staff** - requires the DEMOTION_KEY environment variable. Cannot self-demote.

FREIGHT CONTROL

OPERATIONS

Dashboard

Warehouses

Inventory

Shipments

Containers

Cargo Items

MASTER DATA

Carriers

Routes

Locations

ANALYTICS

Reports

ADMINISTRATION

Users

Approve Users

Admin Requests

FREIGHT CARGO CONTROL

Operational Overview

admin@example.com

Sign out

Users & Roles

Manage application users, their roles, and warehouse assignments.

CREATE NEW USER

Creates a Supabase Auth account and a database user in one step. Default password is "dev-only"; the user must change it after first login.

Name

Email

Role

Select role...

Create User

UPDATE USER

Enter a User ID to load their data, then edit fields below.

User ID

Load

ASSIGN USER TO WAREHOUSE

Admins get added to the manages table. Staff get their warehouse updated.

User ID

Warehouse

Select warehouse...

Assign

DEMOTE ADMIN TO STAFF

Requires demotion key for confirmation.

User ID to Demote

Demotion Key

Demote

ID	Name	Email	Role	Status	Assigned Warehouses
1	Admin User	admin@example.com	ADMIN	active	1 - Calgary Central Warehouse, 2 - Vancouver Logistics Park, 3 - Toronto Brampton DC, 5 - Edmonton Logistics Centre, 9 - Rotterdam Hub Warehouse, 10 - Shanghai Bonded Warehouse
2	Test Admin	test@example.com	ADMIN	active	4 - Montreal Dorval Warehouse, 6 - Winnipeg Distribution Centre, 7 - Ottawa South Warehouse, 8 - Los Angeles Gateway Warehouse, 11 - Singapore FTZ Warehouse, 12 - Dubai Jebel Ali Warehouse
3	Alice Carter	staff1@example.com	STAFF	active	1 - Calgary Central Warehouse
4	Bob Nguyen	staff2@example.com	STAFF	active	2 - Vancouver Logistics Park
5	Clara Johansson	staff3@example.com	STAFF	active	3 - Toronto Brampton DC
6	David Okafor	staff4@example.com	STAFF	active	4 - Montreal Dorval Warehouse
13	Emily Zhang	staff5@example.com	STAFF	pending_registration	-

N

ight Mode

7.16 Approve Users Page (Admin Only)

Lists all users with `account_status = 'pending_registration'`. Admins can approve (sets status to active) or reject (sets status to rejected) each pending user.

The screenshot shows the 'Approve New Users' page in the Freight Control system. The page title is 'Approve New Users' with a subtitle 'Users who registered are listed here for approval. Approved users get staff access by default.' The page displays a table with the following data:

ID	Name	Email	Actions
13	Emily Zhang	staff15@example.com	<button>Approve</button> <button>Reject</button>

The left sidebar contains the following menu items: Dashboard, Warehouses, Inventory, Shipments, Containers, Cargo Items, Carriers, Routes, Locations, Reports, Users, Approve Users (selected), and Admin Requests. The top right corner shows the user 'admin@example.com' and a 'Sign out' button.

7.17 Admin Requests Page (Admin Only)

Lists all pending admin access requests. Shows the requesting user's name, email, and reason. Admins can approve (triggers the promotion SQL transaction) or reject (marks the request as rejected).

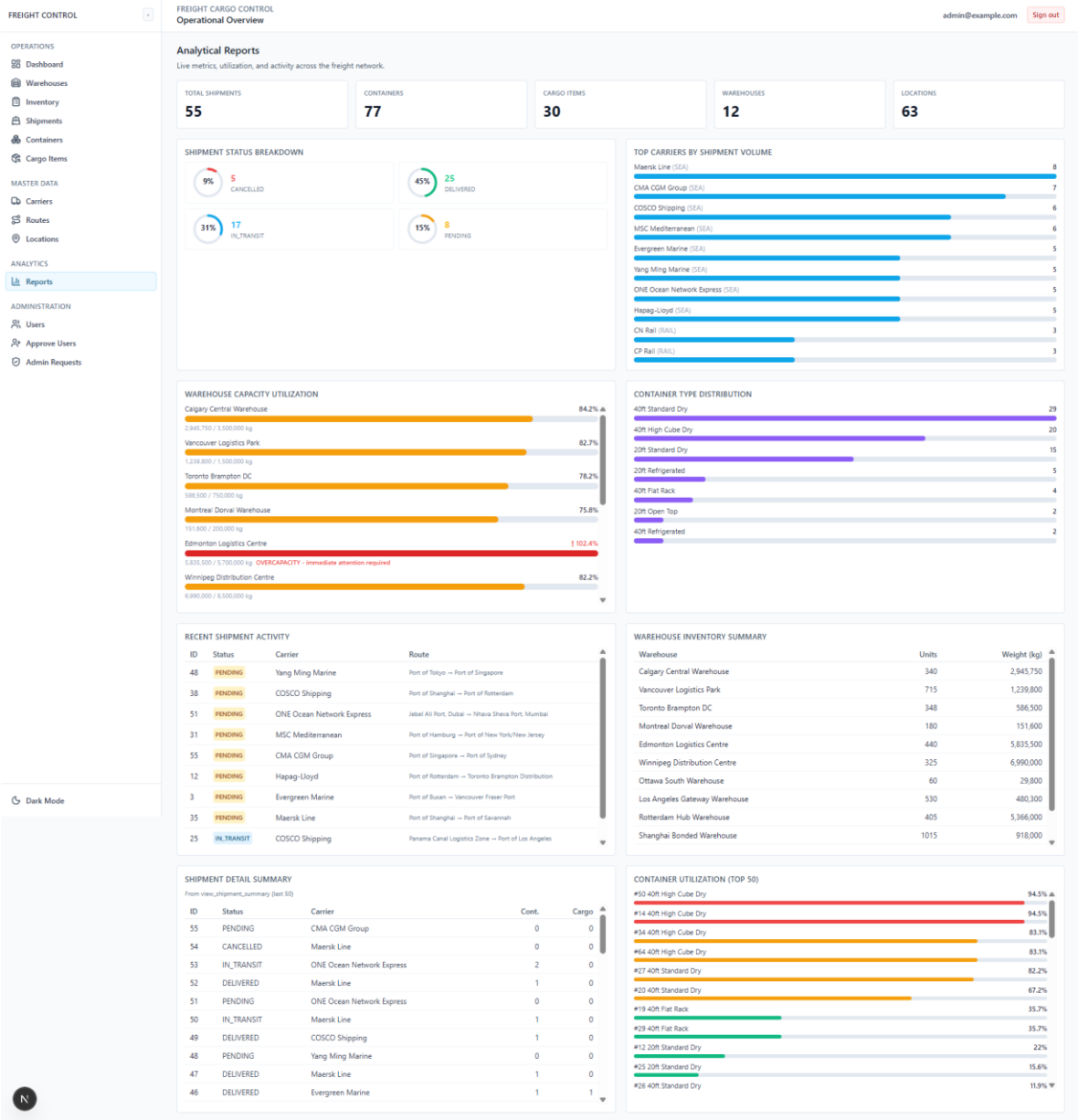
The screenshot shows the 'Admin Access Requests' page in the Freight Control system. The page title is 'Admin Access Requests' with a subtitle 'Staff members requesting promotion to admin. Approving removes their warehouse_staff row and grants admin privileges.' The page displays a table with the following data:

Name	Email	Reason	Date	Actions
Alice Carter	staff1@example.com	I have been managing Calgary Central Warehouse inventory for 6 months and need admin access to approve shipments and manage staff assignments.	2026-04-15	<button>Approve</button> <button>Reject</button>
Bob Nguyen	staff2@example.com	Promoted to shift lead at Vancouver Logistics Park. Need admin privileges to handle carrier onboarding and route management.	2026-04-15	<button>Approve</button> <button>Reject</button>

The left sidebar contains the following menu items: Dashboard, Warehouses, Inventory, Shipments, Containers, Cargo Items, Carriers, Routes, Locations, Reports, Users, Approve Users, and Admin Requests (selected). The top right corner shows the user 'admin@example.com' and a 'Sign out' button.

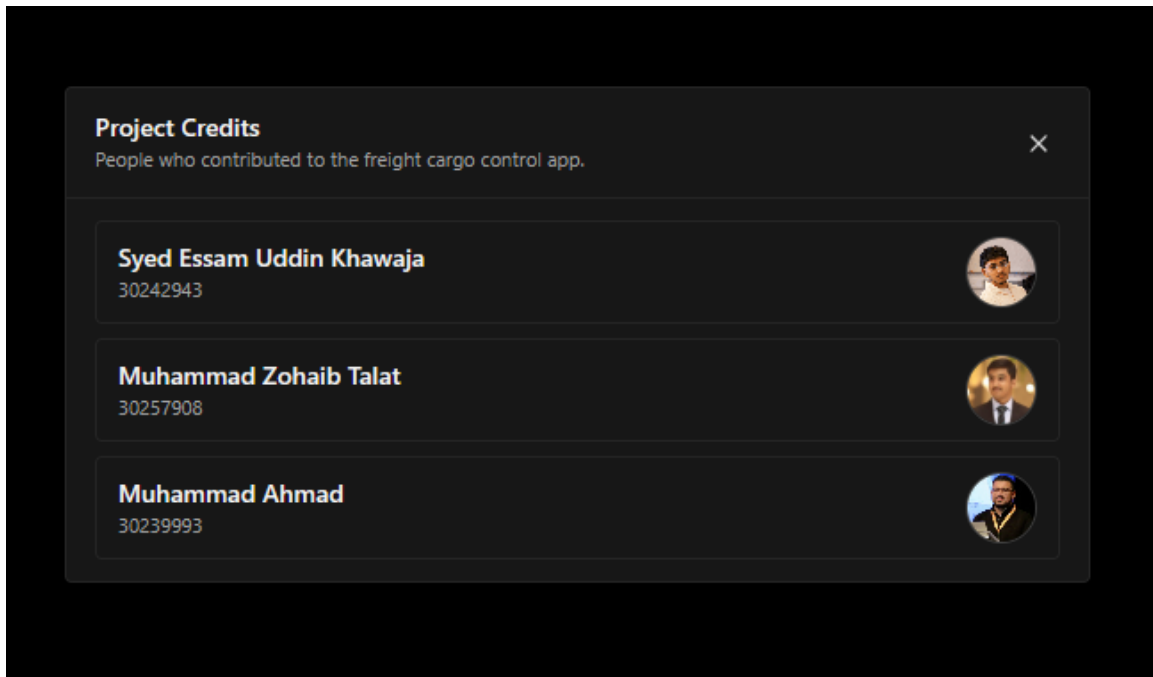
7.18 Light Mode Showcase

The application supports both dark mode (default) and light mode, toggled via a button at the bottom of the sidebar. All pages are fully styled for both themes using Tailwind's `dark:` class variants.



7.19 Credits Pop-up

The left side bar at the bottom left, has a credits button which shares the details of the Project Authors.



8. Supabase Dashboard and SQL Editor

These screenshots are included separately from the application interface because they document the hosted database environment and direct SQL verification workflow rather than the Next.js frontend itself.

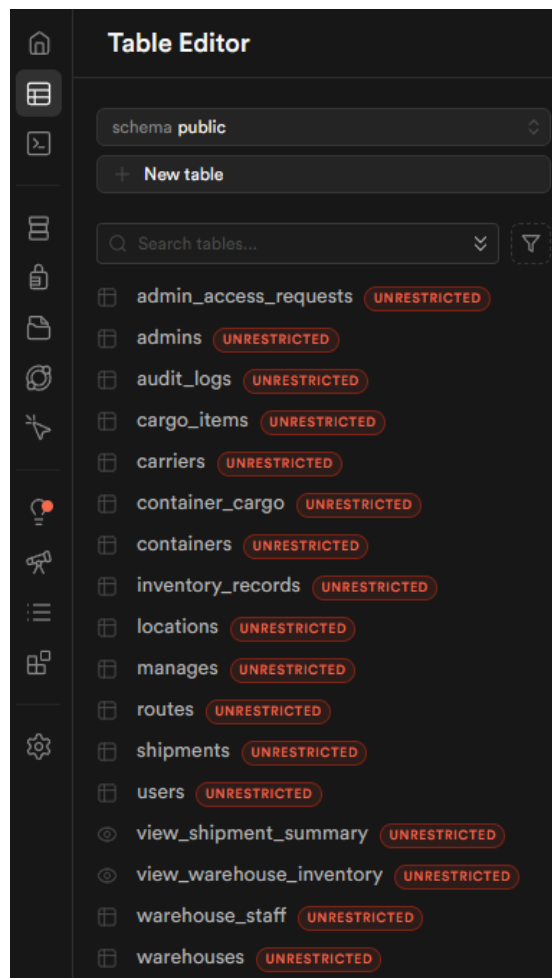
8.1 Supabase Hosting Account

The PostgreSQL relational database which is connected to our platform can be viewed on Supabase, where it is hosted, by logging into the following account:

- Email: ‘ ’
- Password: ‘ ’

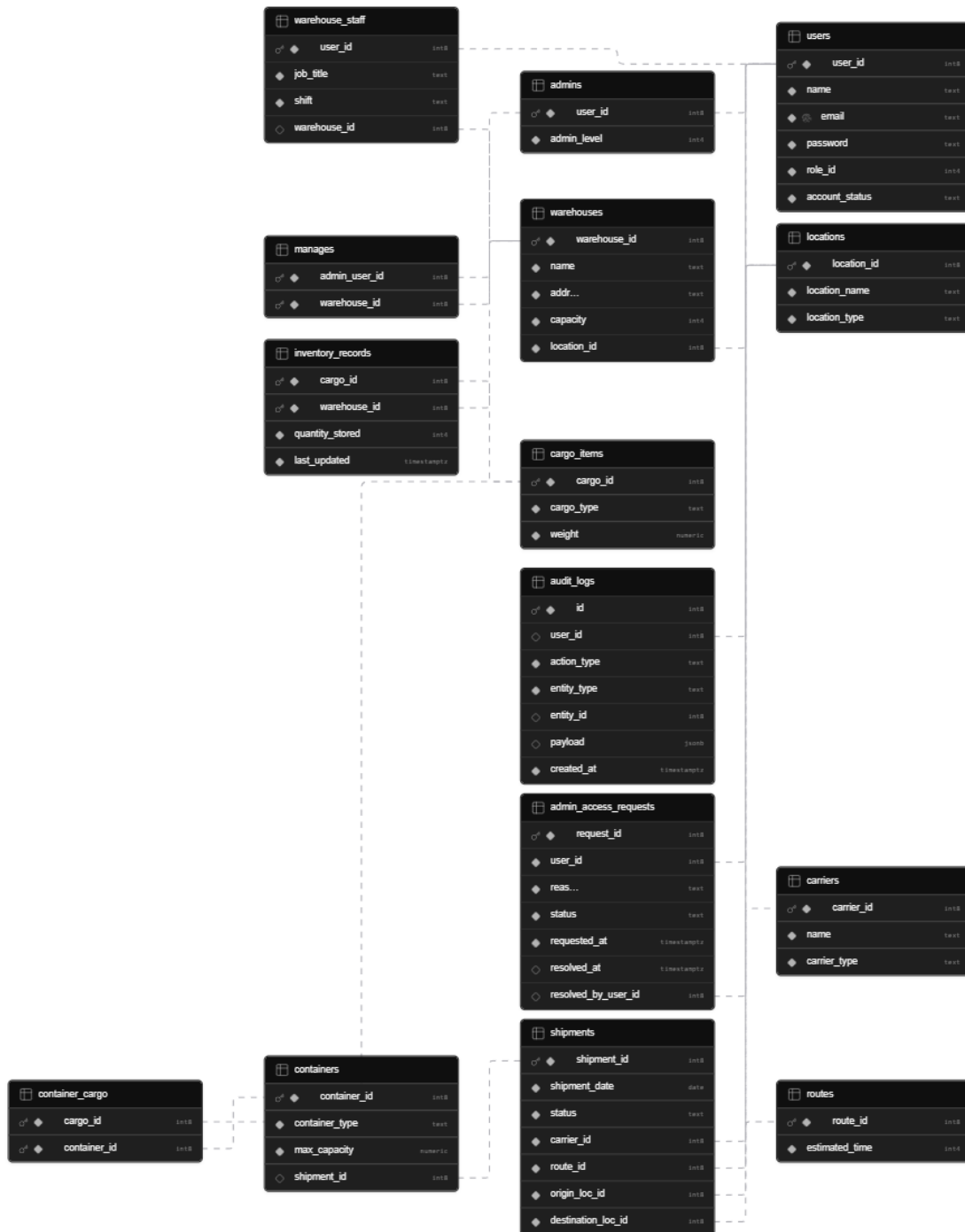
8.2 Supabase SQL Tables Overview

This screenshot shows the Supabase dashboard's Table Editor, displaying all the tables in the public schema. This confirms the database structure matches our relational model.



8.3 Supabase SQL Database Overview Diagram

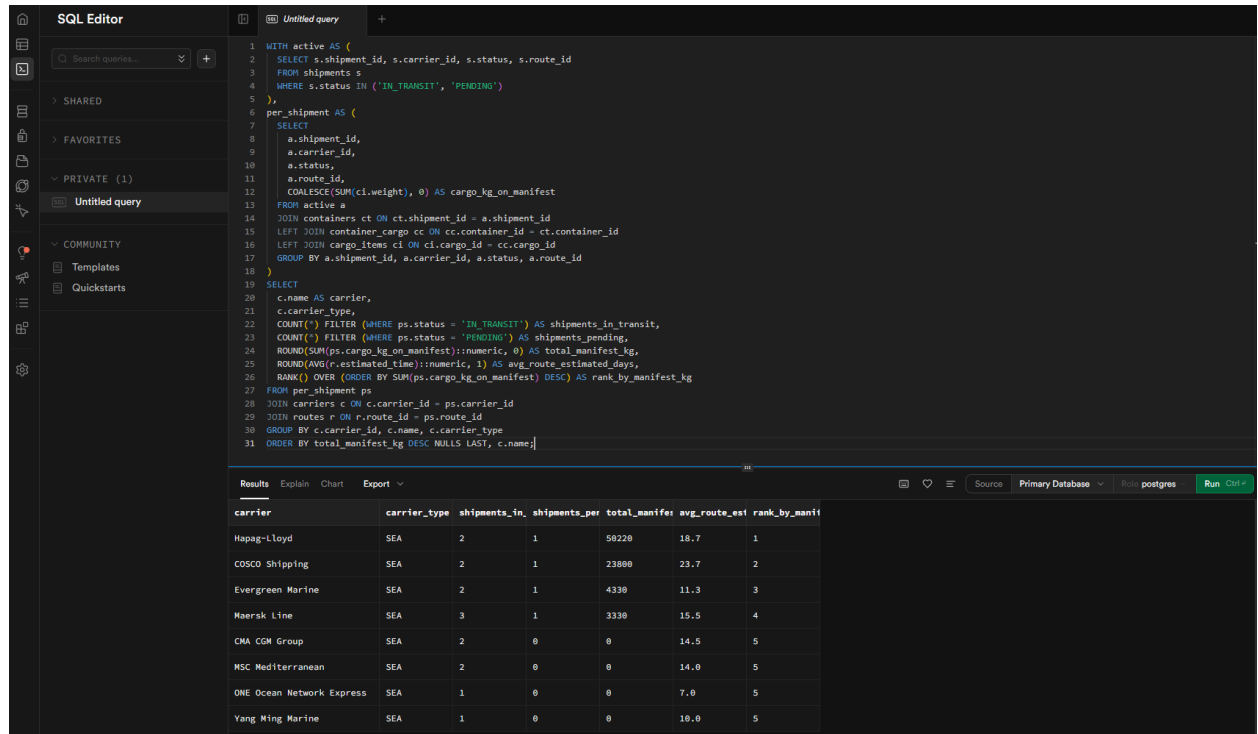
The Supabase dashboard provides an auto-generated database diagram showing all tables and their foreign key relationships. This provides an alternative visualization of the relational model.



8.4 Supabase Example SQL Query Run

This screenshot demonstrates running an ad-hoc SQL query in the Supabase SQL Editor. This is important because it shows that the database supports direct SQL queries, which means that one can verify data, run custom queries, and test the stored functions directly even without setting up the full application locally.

Test Query:



The screenshot displays the Supabase SQL Editor interface. On the left, a sidebar contains navigation options: 'SQL Editor', 'SEARCH queries...', 'SHARED', 'FAVORITES', 'PRIVATE (1)', 'Untitled query', 'COMMUNITY', 'Templates', and 'Quickstarts'. The main editor area shows a SQL query with line numbers 1 through 31. The query is a complex aggregation query involving multiple tables and subqueries. Below the editor, the 'Results' tab is active, showing a table with 7 columns: carrier, carrier_type, shipments_in, shipments_per, total_manifes, avg_route_est, and rank_by_manif. The table contains 8 rows of data, including carriers like Hapag-Lloyd, COSCO Shipping, Evergreen Marine, Maersk Line, CMA CGM Group, MSC Mediterranean, ONE Ocean Network Express, and Yang Ming Marine.

```
1 WITH active AS (  
2   SELECT s.shipment_id, s.carrier_id, s.status, s.route_id  
3   FROM shipments s  
4   WHERE s.status IN ('IN_TRANSIT', 'PENDING')  
5 ),  
6 per_shipment AS (  
7   SELECT  
8     a.shipment_id,  
9     a.carrier_id,  
10    a.status,  
11    a.route_id,  
12    COALESCE(SUM(ci.weight), 0) AS cargo_kg_on_manifest  
13   FROM active a  
14   JOIN containers ct ON ct.shipment_id = a.shipment_id  
15   LEFT JOIN container_cargo cc ON cc.container_id = ct.container_id  
16   LEFT JOIN cargo_items ci ON ci.cargo_id = cc.cargo_id  
17   GROUP BY a.shipment_id, a.carrier_id, a.status, a.route_id  
18 )  
19 SELECT  
20   c.name AS carrier,  
21   c.carrier_type,  
22   COUNT(*) FILTER (WHERE ps.status = 'IN_TRANSIT') AS shipments_in_transit,  
23   COUNT(*) FILTER (WHERE ps.status = 'PENDING') AS shipments_pending,  
24   ROUND(SUM(ps.cargo_kg_on_manifest)::numeric, 0) AS total_manifest_kg,  
25   ROUND(AVG(r.estimated_time)::numeric, 1) AS avg_route_estimated_days,  
26   RANK() OVER (ORDER BY SUM(ps.cargo_kg_on_manifest) DESC) AS rank_by_manifest_kg  
27   FROM per_shipment ps  
28   JOIN carriers c ON c.carrier_id = ps.carrier_id  
29   JOIN routes r ON r.route_id = ps.route_id  
30   GROUP BY c.carrier_id, c.name, c.carrier_type  
31   ORDER BY total_manifest_kg DESC NULLS LAST, c.name;
```

carrier	carrier_type	shipments_in	shipments_per	total_manifes	avg_route_est	rank_by_manif
Hapag-Lloyd	SEA	2	1	59229	18.7	1
COSCO Shipping	SEA	2	1	23888	23.7	2
Evergreen Marine	SEA	2	1	4338	11.3	3
Maersk Line	SEA	3	1	3339	15.5	4
CMA CGM Group	SEA	2	0	0	14.5	5
MSC Mediterranean	SEA	2	0	0	14.0	5
ONE Ocean Network Express	SEA	1	0	0	7.0	5
Yang Ming Marine	SEA	1	0	0	10.0	5

9. User Guide

This section provides a quick guide for using the application. A more detailed guide can be found in `README.md`.

9.1 Accessing the Application

Option A: Live Demo (Recommended)

30. Navigate to <https://cpssc-471-inventory-manager.vercel.app>

31. Log in with any seed account (e.g., `admin@example.com` / `dev-only`)

Option B: Local Setup

32. Clone the repository: `git clone <repo-url>`

33. Install dependencies: `npm install`

34. Create a Supabase project (see `README.md` for detailed instructions)

35. Create `.env.local` with your `DATABASE_URL`, `NEXT_PUBLIC_SUPABASE_URL`,
`NEXT_PUBLIC_SUPABASE_ANON_KEY`, and `DEMOTION_KEY`

36. Run database setup: `npm run db:setup`

37. Start the dev server: `npm run dev`

38. Open <http://localhost:3000>

9.2 Admin Workflow

39. **Log in** as `admin@example.com` (password: `dev-only`)

40. **Dashboard** shows personalized KPIs for your assigned warehouses

41. **Warehouses** - create, edit, or delete warehouses

42. **Inventory** - view all inventory records, adjust quantities (calls `fn_update_inventory`)

43. **Shipments** - create new shipments, update status, assign containers

44. **Containers** - manage containers and assign cargo

45. **Cargo Items** - create new cargo types

46. **Carriers** - manage transport companies

47. **Routes** - manage shipping routes

48. **Locations** - manage ports, terminals, warehouses, and distribution centers

49. **Reports** - view aggregated data from `view_warehouse_inventory` and
`view_shipment_summary`

50. **Users** - create users, assign to warehouses, promote/demote

51. **Approve Users** - approve or reject pending registrations

52. **Admin Requests** - approve or reject staff promotion requests

9.3 Staff Workflow

- 53. **Log in** as staff1@example.com (password: dev-only)
- 54. **Dashboard** shows KPIs for your assigned warehouse only
- 55. **Inventory** - view and adjust inventory for your warehouse
- 56. **Shipments, Containers, Cargo Items** - view and manage
- 57. **Request Admin** - submit a promotion request with a reason (max 1000 characters)

9.4 Registration Workflow

- 58. Navigate to /register
- 59. Fill in name, email, and password
- 60. After registration, you are redirected to /pending
- 61. An admin must approve your registration from /admin/approvals/users
- 62. Once approved, you can log in and access the dashboard

9.5 Running SQL Queries Directly

If the application is not accessible or you want to verify database operations directly:

- 63. Log in to the Supabase dashboard (<https://supabase.com/dashboard>)
- 64. Select the project
- 65. Go to **SQL Editor** in the left sidebar
- 66. Run any SQL query against the database

10. Summary of Changes from Initial Design

Aspect	Original Plan (Proposal / Progress Reports)	Final Implementation
Total tables	12 (from EERD)	15 (added audit_logs, admin_access_requests; plus account_status column)
Authentication	Planned as manual cookie-based	Supabase Auth with JWT cookies and @supabase/ssr
User lifecycle	No status tracking	Full lifecycle: pending_registration -> active/rejected/disabled
Admin promotion	Not planned	Full workflow with admin_access_requests table, staff request form, admin approval UI
Audit logging	Not planned	audit_logs table, fn_audit_log() function, trg_inventory_records_audit trigger
Inventory management	Basic CRUD only	Stored function fn_update_inventory() with UPSERT, negative-quantity guard, and audit logging
Views	2 planned (basic)	2 implemented with JOINS and aggregation
Staff-warehouse link	Only through manages	Direct warehouse_id FK on warehouse_staff for simple assignment
Dashboard	Planned as static KPIs	Personalized, warehouse-scoped KPIs with real SQL aggregations
Seed data	Minimal	63 locations, 12 warehouses, 22 carriers, 55 shipments, 77 containers across 6 continents
Dark mode	Not planned	Full dark/light mode with pure-black background and neutral grey accents
Mobile support	Not planned	Responsive layout with collapsible sidebar drawer
Delete protection	Not planned	All delete operations check for FK references before attempting deletion

Aspect	Original Plan (Proposal / Progress Reports)	Final Implementation
Demotion workflow	Not planned	Admin demotion with DEMOTION_KEY security, self-demotion blocked
Auth user seeding	Not planned	Direct SQL INSERT into auth.users/auth.identities with bcrypt hashing via pgcrypto

11. Conclusion

This project demonstrates a complete, databases-first approach to building a freight cargo management system. Starting from an Enhanced Entity-Relationship Diagram, we applied the standard EERD-to-relational-model conversion algorithm to produce a normalized PostgreSQL schema. We then implemented the schema with full constraint enforcement, added views for reporting, stored functions for business logic, and triggers for automatic audit logging. The web interface provides role-based access for two user types, personalized dashboards, and full CRUD operations on all entities. The system includes realistic seed data modeling a global freight network, making it suitable for demonstration and evaluation. All SQL is visible and parameterized, and the database can be queried directly through the Supabase SQL Editor even without the web application running.