Food Truck Booking App - Setup and Deployment Guide

This document walks you through creating the Supabase backend, running the migration, configuring environment variables, and deploying the Next.js app to Vercel. It also explains the built-in change control (release switching) rollback mechanism.

1) Prerequisites

Accounts: Supabase and Vercel.

Local tools: Node.js 18+ and Git (optional).

2) Create the Supabase project

1. In Supabase, create a new project.

2. In Project Settings -> API, copy:

Item	Where used
Project URL	NEXT_PUBLIC_SUPABASE_URL
anon public key	NEXT_PUBLIC_SUPABASE_ANON_KEY
service_role key	SUPABASE_SERVICE_ROLE_KEY (server-only)

3) Apply the database migration

Open the Supabase SQL Editor and run the SQL in:

```
supabase/migrations/001_init.sql
```

This creates tables, views, RPC functions (accept/ignore, activate_release), and Row Level Security policies.

4) Create your first admin

After you sign up in the app, your profile role starts as empty. To bootstrap the first admin:

1. In Supabase Table Editor -> **profiles**, find your user row.

2. Set **role** to admin.

3. (Optional) Create your first baseline release is already seeded as 1.0.0.

5) Local development

1. Copy environment file:

```
cp .env.local.example .env.local
```

2. Fill in values in .env.local from Supabase.

3. Install + run:

```
npm i
npm run dev
```

App runs at http://localhost:3000

6) Create a truck record for a truck owner

For the MVP, each truck owner has one truck row (unique(owner_id)).

After you assign a user role to truck_owner, insert a row into **trucks**:

```
insert into public.trucks (owner_id, display_name)
values ('<truck_owner_user_uuid>', 'My Truck Name');
```

Once created, that owner can see blanket requests and requests to their truck.

7) Business requests and public map

Business owners create requests from Business Dashboard -> New request.

Requests require a date/time and location name. Latitude/longitude are optional but recommended so the public map can place a marker.

The public map reads the view public.public_bookings which only contains **accepted** requests.

8) Deploy to Vercel

1. Push the project to GitHub (recommended) or use Vercel import from folder.

2. In Vercel, create a new project and import the repo.

3. Add environment variables in Vercel Project Settings -> Environment Variables:

```
NEXT_PUBLIC_SUPABASE_URL=...
NEXT_PUBLIC_SUPABASE_ANON_KEY=...
SUPABASE_SERVICE_ROLE_KEY=...
```

4. Deploy. Vercel will run npm run build and host the app.

9) Notifications

This MVP uses an in-app inbox concept via the truck_requests_inbox view and the truck dashboard. For email/SMS notifications, add a Supabase Edge Function or external provider (SendGrid/Twilio) and trigger it on insert of truck_requests.

10) Change control / rollback

The change control mechanism is database-driven via the releases table:

- Admin creates releases (versions) in Admin -> Change control.

- Exactly one release should be active.

- To roll back, activate a previous release. The DB function activate_release flips the active flag and logs the change in release_changes.

How it helps: use the release features JSON to gate enhancements (feature flags). If a bug is found, roll back by re-activating the prior release without redeploying code.

11) Production rollback options

You have two rollback levers:

1) **Vercel deployment rollback**: revert to a previous deployment in Vercel if code is broken.

2) **In-app rollback**: switch the active release if the issue is caused by a feature/config change gated by releases.

12) Using the provided PowerShell scaffolder

From a folder where you want the project created, run:

```
powershell -ExecutionPolicy Bypass -File .\create-project.ps1
```

It will create a new folder food-truck-booking with all files.