Print to PDF

Team Ride Pro - Complete Supabase Integration Guide

Version: 1.0
Date: December 2024
Purpose: Complete step-by-step guide for integrating Team Ride Pro with Supabase

Table of Contents

1. Quick Start Guide

This is your starting point for integrating Supabase with Team Ride Pro.

> ### ✅ Authentication Requirements - CONFIRMED POSSIBLE
>
> All your authentication requirements are feasible with Supabase:
>
> 1. ✅ **Admin Coaches**: Email/Password → Full site access
> 2. ✅ **Non-Admin Coaches**: Phone + SMS code → "Coach Assignments" tab only
> 3. ✅ **Riders**: Phone + SMS code → "Rider Assignments" tab only

🚀 Recommended Implementation Order

Phase 1: Setup (Day 1) ⏱️ ~2-3 hours

1. ✅ Create Supabase project
2. ✅ Run database schema SQL
3. ✅ Configure email authentication
4. ✅ Get API credentials
5. ✅ Update `scripts/supabase-config.js`

Phase 2: Admin Coaches (Day 2) ⏱️ ~2 hours

1. ✅ Create first admin account
2. ✅ Test email/password login
3. ✅ Verify full access works

4. ✅ Test data loading/saving

Phase 3: Data Migration (Day 3) ⏱️ ~1-2 hours

1. ✅ Export localStorage data
2. ✅ Run migration script
3. ✅ Verify all data migrated
4. ✅ Normalize phone numbers

Phase 4: Phone Auth Infrastructure (Day 4-5) ⏱️ ~4-6 hours

1. ✅ Set up Twilio account
2. ✅ Configure SMS in Supabase
3. ✅ Create phone verification Edge Function
4. ✅ Test phone auth flow

Phase 5: Non-Admin Coaches (Day 6) ⏱️ ~3-4 hours

1. ✅ Implement phone login for coaches
2. ✅ Auto-create user_roles entries
3. ✅ Restrict to "Coach Assignments" tab
4. ✅ Test with real coach account

Phase 6: Riders (Day 7) ⏱️ ~2-3 hours

1. ✅ Adapt phone login for riders
2. ✅ Restrict to "Rider Assignments" tab
3. ✅ Test with real rider account

Phase 7: Deployment (Day 8) ⏱️ ~2-3 hours

1. ✅ Deploy to GitHub Pages
2. ✅ Test on production URL

3. ✅ Connect custom domain
4. ✅ Final testing

Total Estimated Time: 1-2 weeks (depending on experience level)

⚠️ Important Notes

Phone Authentication

- Requires SMS provider (Twilio recommended)
- Costs ~$0.0075 per SMS
- Free trial available ($15.50 credit)
- Phone numbers must be in E.164 format (+14155551234)

Security

- Supabase `anon` key is meant to be public
- Security comes from Row Level Security (RLS) policies
- Already configured in database schema
- Admin keys should NEVER be exposed

2. Complete Setup Guide

This guide will walk you through setting up Supabase for Team Ride Pro, including authentication, database setup, and data migration.

1. Supabase Project Setup

Step 1.1: Create/Login to Supabase Account

1. Go to https://supabase.com
2. Sign up or log in to your account
3. You should see your dashboard

Step 1.2: Create a New Project

1. Click **"New Project"** button
2. Fill in project details:
    - **Name**: `team-ride-pro` (or your preferred name)
    - **Database Password**: Create a strong password (save this securely!)
    - **Region**: Choose closest to your users (e.g., `US West` for California)
3. Click **"Create new project"**
4. Wait 2-3 minutes for project initialization

Step 1.3: Get Your API Credentials

1. In your project dashboard, go to **Settings** (gear icon) → **API**
2. You'll need two keys:
    - **Project URL**: `https://xxxxx.supabase.co`
    - **anon public key**: Long string starting with `eyJ...`

3. Copy both values (you'll need them later)

2. Authentication Configuration

Step 2.1: Enable Phone Authentication

1. In Supabase dashboard, go to **Authentication** → **Providers**
2. Find **"Phone"** in the list
3. Toggle it **ON**
4. **Important**: Phone authentication requires a third-party SMS provider (Twilio, MessageBird, etc.)
    - For testing/development, you can use Supabase's test mode (limited functionality)
    - For production, you'll need to set up Twilio

Step 2.2: Configure Email Authentication (for Admin Coaches)

1. In **Authentication** → **Providers**
2. Ensure **"Email"** is enabled (should be by default)
3. Configure email settings:
    - **Enable email confirmations**: Toggle OFF for easier testing (can enable later)
    - **Secure email change**: Toggle ON (recommended)

Step 2.3: Configure Redirect URLs

1. Go to **Authentication** → **URL Configuration**
2. Add your site URLs:
    - **Site URL**: `http://localhost:8000` (for local testing)
    - **Redirect URLs**: Add:
        - `http://localhost:8000/**`
        - `https://teamridepro.com/**` (for production)

- `https://*.github.io/**` (if using GitHub Pages)

3. Database Setup

Step 3.1: Run Database Schema

1. In Supabase dashboard, go to **SQL Editor**
2. Click **"New query"**
3. Open the file `sql/database-schema.sql` from this project
4. Copy the entire contents
5. Paste into the SQL Editor
6. Click **"Run"** (or press Ctrl+Enter)
7. You should see "Success. No rows returned" message

Step 3.2: Verify Tables Created

1. Go to **Table Editor** in the sidebar
2. You should see these tables:
    - `user_roles`
    - `riders`
    - `coaches`
    - `rides`
    - `rider_feedback`
    - `ride_notes`
    - `rider_availability`
    - `season_settings`
    - `auto_assign_settings`
    - `routes`

Step 3.3: Update Schema for Phone Authentication (Required)

We need to add a function to verify phone numbers. Run this SQL:

```
-- Function to verify if a phone number exists in coaches or riders tab
CREATE OR REPLACE FUNCTION verify_phone_number(phone_to_check TEXT)
RETURNS TABLE(
    exists_in_coaches BOOLEAN,
    exists_in_riders BOOLEAN,
    coach_id BIGINT,
    rider_id BIGINT
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        EXISTS(SELECT 1 FROM coaches WHERE phone = phone_to_check) as e
        EXISTS(SELECT 1 FROM riders WHERE phone = phone_to_check) as ex
        (SELECT id FROM coaches WHERE phone = phone_to_check LIMIT 1)::
        (SELECT id FROM riders WHERE phone = phone_to_check LIMIT 1)::B
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

-- Grant execute permission to anon users (for phone verification)
GRANT EXECUTE ON FUNCTION verify_phone_number(TEXT) TO anon;
GRANT EXECUTE ON FUNCTION verify_phone_number(TEXT) TO authenticated;
```

Step 3.4: Add Admin Role to user_roles Table

We need to distinguish between admin coaches and regular coaches. Update the user_roles table:

```
-- Update user_roles to support admin coaches
ALTER TABLE user_roles
ALTER COLUMN role TYPE TEXT;

-- Remove the constraint and add a new one that includes 'admin'
```

```
DROP POLICY IF EXISTS "Coaches can view all user roles" ON user_roles;
DROP POLICY IF EXISTS "Coaches can insert user roles" ON user_roles;
DROP POLICY IF EXISTS "Coaches can update user roles" ON user_roles;

-- Recreate constraint with admin role
ALTER TABLE user_roles
DROP CONSTRAINT IF EXISTS user_roles_role_check;

ALTER TABLE user_roles
ADD CONSTRAINT user_roles_role_check
CHECK (role IN ('admin', 'coach', 'ride_leader', 'rider'));

-- Update RLS policies to allow admins (admins have coach privileges +
CREATE POLICY "Admins and coaches can view all user roles" ON user_role
    FOR SELECT USING (
        EXISTS (
            SELECT 1 FROM user_roles ur
            WHERE ur.user_id = auth.uid() AND ur.role IN ('admin', 'coa
        )
    );

CREATE POLICY "Admins and coaches can insert user roles" ON user_roles
    FOR INSERT WITH CHECK (
        EXISTS (
            SELECT 1 FROM user_roles ur
            WHERE ur.user_id = auth.uid() AND ur.role IN ('admin', 'coa
        )
    );

CREATE POLICY "Admins and coaches can update user roles" ON user_roles
    FOR UPDATE USING (
        EXISTS (
            SELECT 1 FROM user_roles ur
            WHERE ur.user_id = auth.uid() AND ur.role IN ('admin', 'coa
        )
```

4. Data Migration

Step 4.1: Export Current Local Data

1. Open your application in the browser
2. Open Developer Console (F12)
3. Run this command to export your data:

```
// Export current localStorage data
const data = JSON.parse(localStorage.getItem('teamRideProData') || '{}'
const blob = new Blob([JSON.stringify(data, null, 2)], { type: 'applica
const url = URL.createObjectURL(blob);
const a = document.createElement('a');
a.href = url;
a.download = 'team-ride-pro-export-' + new Date().toISOString().split('
a.click();
```

4. Save the downloaded file - this is your backup!

Step 4.2: Prepare Migration Script

See the Data Migration Guide section for the complete migration script.

Step 4.3: Create First Admin Coach Account

1. In Supabase dashboard, go to **Authentication** → **Users**
2. Click **"Add user"** → **"Create new user"**
3. Fill in:
    - **Email**: Your admin email
    - **Password**: Strong password
    - **Auto Confirm User**: Check this (or confirm via email)
4. Click **"Create user"**

5. Copy the **User UID** (UUID format)
6. Go to **SQL Editor** and run:

```
-- Replace USER_UUID_HERE with the UUID from step 5
-- Replace COACH_ID_HERE with the ID of your coach record
INSERT INTO user_roles (user_id, role)
VALUES ('USER_UUID_HERE', 'admin')
ON CONFLICT (user_id) DO UPDATE SET role = 'admin';

-- Link the auth user to your coach record
UPDATE coaches
SET user_id = 'USER_UUID_HERE'
WHERE id = COACH_ID_HERE;
```

5. Application Configuration

Step 5.1: Update Supabase Config

1. Open `scripts/supabase-config.js`
2. Replace the placeholder values:

```
const SUPABASE_URL = 'https://YOUR_PROJECT_REF.supabase.co';
const SUPABASE_ANON_KEY = 'YOUR_ANON_KEY_HERE';
```

3. Use the values from Step 1.3

6. Deployment

See the Deployment Guide section for detailed deployment instructions.

Authentication Flow Summary

Admin Coaches

1. Login with **email + password**
2. Full access to all tabs and features
3. Role: `admin` in `user_roles` table

Non-Admin Coaches

1. Enter **phone number** on login screen
2. System verifies phone exists in `coaches` table
3. SMS code sent to phone (via Twilio)
4. Enter code to authenticate
5. Access limited to **"Coach Assignments"** tab only
6. Role: `coach` in `user_roles` table

Riders

1. Enter **phone number** on login screen
2. System verifies phone exists in `riders` table
3. SMS code sent to phone (via Twilio)
4. Enter code to authenticate
5. Access limited to **"Rider Assignments"** tab only
6. Role: `rider` in `user_roles` table

3. Authentication Feasibility Analysis

Your Requirements Summary

1. **Admin Coaches**: Email/Password login → Full site access
2. **Non-Admin Coaches**: Phone + SMS code → "Coach Assignments" tab only
3. **Riders**: Phone + SMS code → "Rider Assignments" tab only

> ### ✅ Confirmation: YES, This is Possible with Supabase
>
> All three authentication methods are feasible with Supabase, though phone authentication requires some custom implementation.

Detailed Feasibility

1. Admin Coaches (Email/Password) ✅ FULLY SUPPORTED

Status: ✅ Native Supabase feature - no custom code needed

Effort: ⭐ Easy (1-2 hours)

2. Non-Admin Coaches (Phone + SMS) ⚠️ SUPPORTED WITH CUSTOM FLOW

Status: ✅ Possible, requires custom implementation

Effort: ⭐⭐⭐ Moderate (4-8 hours)

3. Riders (Phone + SMS) ⚠️ SUPPORTED WITH CUSTOM FLOW

Status: ✅ Same as non-admin coaches

Effort: ⭐⭐⭐ Moderate (4-8 hours, or shared with coaches implementation)

Implementation Strategy

Recommended Approach: Phased Implementation

- **Phase 1: Admin Coaches (Email/Password)** ⏱️ ~2 hours
- **Phase 2: Phone Authentication Infrastructure** ⏱️ ~4 hours
- **Phase 3: Non-Admin Coaches** ⏱️ ~3 hours
- **Phase 4: Riders** ⏱️ ~2 hours (reuse coach implementation)

Total Estimated Time: 11-15 hours

Cost Estimate

Supabase

- **Free Tier**: Includes phone auth, 50,000 monthly active users
- **Cost**: $0/month for your use case (unless you exceed free tier limits)

Twilio (SMS)

- **Free Trial**: $15.50 credit (~2,000 SMS)
- **Production**: ~$0.0075 per SMS
- **Monthly Estimate**:
  - 50 coaches × 10 logins/month = 500 SMS = ~$3.75/month
  - 100 riders × 5 logins/month = 500 SMS = ~$3.75/month
  - **Total**: ~$7.50/month for 1,000 SMS

Annual SMS Cost: ~$90/year (very affordable)

Final Recommendation

✅ **YES, proceed with Supabase**

Your authentication requirements are all feasible:

1. ✅ Admin coaches (email/password) - native feature
2. ✅ Non-admin coaches (phone/SMS) - requires custom flow but well-supported
3. ✅ Riders (phone/SMS) - same as coaches

4. Authentication Implementation Details

This document explains how the authentication system works and addresses the specific requirements for admin coaches, non-admin coaches, and riders.

Authentication Requirements Summary

1. **Admin Coaches**: Email + Password → Full site access
2. **Non-Admin Coaches**: Phone + SMS Code → "Coach Assignments" tab only
3. **Riders**: Phone + SMS Code → "Rider Assignments" tab only

Implementation Approach

Challenge: Supabase Phone Auth Limitations

Supabase's phone authentication has these characteristics:

- Uses phone number as the primary identifier (like email for email auth)
- Automatically creates auth users when phone auth succeeds
- Requires phone numbers to be unique across all users
- Requires an SMS provider (Twilio, MessageBird, etc.)

Solution: Hybrid Approach with Edge Function

We'll implement a custom flow that:

1. Validates phone numbers against existing records BEFORE authentication
2. Uses Supabase's phone auth for OTP delivery
3. Auto-creates user_roles entries after successful phone auth
4. Assigns appropriate role based on phone match (coach vs rider)

Detailed Flow Diagrams

Admin Coach Login Flow

```
User enters email + password
    ↓
Supabase email auth
    ↓
Success → Load user_roles
    ↓
Check role = 'admin'
    ↓
Grant full access
```

Non-Admin Coach Login Flow

```
User enters phone number
    ↓
Call Edge Function: verify-phone
    ↓
Check if phone exists in coaches table
    ↓
If YES:
    Send OTP via Supabase phone auth
    ↓
User enters OTP code
    ↓
Verify OTP with Supabase
    ↓
Create/link auth user
    ↓
Auto-create user_roles entry (role = 'coach')
    ↓
Grant access to "Coach Assignments" tab only
```

Rider Login Flow

```
User enters phone number
    ↓
Call Edge Function: verify-phone
    ↓
Check if phone exists in riders table
    ↓
If YES:
    Send OTP via Supabase phone auth
    ↓
User enters OTP code
    ↓
Verify OTP with Supabase
    ↓
Create/link auth user
    ↓
Auto-create user_roles entry (role = 'rider')
    ↓
Grant access to "Rider Assignments" tab only
```

Twilio Setup for SMS

Step 1: Create Twilio Account

1. Go to https://www.twilio.com
2. Sign up for free trial (includes $15.50 credit)
3. Verify your phone number

Step 2: Get Credentials

1. In Twilio Console → Settings → General
2. Copy:
    - **Account SID**
    - **Auth Token**

Step 3: Get Phone Number

1. In Twilio Console → Phone Numbers → Manage → Buy a number
2. Select a number with SMS capability
3. Note the phone number

Step 4: Configure in Supabase

1. In Supabase Dashboard → Authentication → Providers → Phone
2. Enter:
    - **Twilio Account SID**
    - **Twilio Auth Token**
    - **Twilio Phone Number**
3. Save

> **Note**: Free tier has limits. For production, consider upgrading Twilio plan.

Testing Checklist

- ☐ Admin coach can login with email/password
- ☐ Admin coach has full access to all tabs
- ☐ Non-admin coach can login with phone number
- ☐ OTP code is received via SMS
- ☐ Non-admin coach can verify OTP
- ☐ Non-admin coach only sees "Coach Assignments" tab
- ☐ Rider can login with phone number
- ☐ Rider only sees "Rider Assignments" tab
- ☐ Invalid phone numbers are rejected
- ☐ Users without matching phone numbers cannot access app
- ☐ Role changes are reflected immediately after login

- ☐ Logout works correctly
- ☐ Session persists across page refreshes

5. Data Migration Guide

This guide explains how to migrate your existing localStorage data to Supabase.

Pre-Migration Checklist

- ☐ Supabase project is set up
- ☐ Database schema has been run (see Complete Setup Guide Step 3)
- ☐ You have exported your localStorage data (backup)
- ☐ You have your Supabase project URL and anon key

Migration Options

Option 1: Browser Console Script (Recommended)

This script runs in your browser console and migrates data directly to Supabase.

1. Open your application in the browser
2. Open Developer Console (F12)
3. Make sure Supabase is configured in `scripts/supabase-config.js`
4. Paste and run the migration script (see complete script in COMPLETE_GUIDE_COMBINED.md)

> **Note**: The complete migration script is available in the combined markdown guide. It migrates:
>
> - All riders
> - All coaches
> - All rides

> - All routes
> - Season settings
> - Auto-assign settings

Post-Migration Tasks

1. Verify Data

Check that all data was migrated using the verification script provided in the complete guide.

2. Create Admin Coach Account

See Complete Setup Guide Step 4.3 for instructions on creating your first admin account.

3. Normalize Phone Numbers

Ensure all phone numbers are in consistent format (E.164):

```
-- Run in Supabase SQL Editor
UPDATE coaches
SET phone = '+' || REGEXP_REPLACE(phone, '[^0-9]', '', 'g')
WHERE phone IS NOT NULL
  AND phone NOT LIKE '+%'
  AND LENGTH(REGEXP_REPLACE(phone, '[^0-9]', '', 'g')) >= 10;

UPDATE riders
SET phone = '+' || REGEXP_REPLACE(phone, '[^0-9]', '', 'g')
WHERE phone IS NOT NULL
  AND phone NOT LIKE '+%'
  AND LENGTH(REGEXP_REPLACE(phone, '[^0-9]', '', 'g')) >= 10;
```

4. Handle ID Mapping (If Needed)

> **Important**: If your rides reference rider/coach IDs, those IDs will have changed after migration.

You have two options:

- **Option A**: Re-run assignments (recommended)
  - The application will re-assign riders/coaches to rides based on availability
  - Old assignments may need to be recreated
- **Option B**: Create ID mapping and update rides
  - Create a mapping of old IDs → new IDs
  - Update ride.available_coaches and ride.available_riders arrays
  - This is complex and error-prone

Troubleshooting

"RLS policy violation" errors

This means your user doesn't have proper permissions. Ensure:

1. You've created an admin user account
2. The user_roles table has an entry for your user
3. The role is set to 'admin' or 'coach'

"Duplicate key" errors

Data already exists in the database. Options:

1. Clear existing data (dangerous - backup first!)
2. Skip duplicates (modify script to check before insert)
3. Use upsert instead of insert

Phone number format issues

Phone numbers must be consistent. Run the normalization SQL (see Post-Migration Task #3).

6. Deployment Guide

This guide covers deploying your application and connecting it to your custom domain (teamridepro.com).

Deployment Options

Option 1: GitHub Pages (Recommended for Testing)

GitHub Pages is free and perfect for hosting static sites.

Step 1: Create GitHub Repository

1. Go to https://github.com and sign in
2. Click **"New repository"**
3. Repository name: `team-ride-pro` (or your preferred name)
4. Choose **Public** or **Private** (Public is free)
5. **Don't** initialize with README (we'll push existing code)
6. Click **"Create repository"**

Step 2: Push Your Code

If you don't have Git set up locally:

1. Install Git: https://git-scm.com/downloads
2. Open terminal/command prompt in your project folder
3. Run these commands:

```
# Initialize git repository
git init
```

```
# Add all files
git add .

# Create initial commit
git commit -m "Initial commit - Team Ride Pro"

# Add GitHub remote (replace USERNAME and REPO_NAME)
git remote add origin https://github.com/USERNAME/REPO_NAME.git

# Push to GitHub
git branch -M main
git push -u origin main
```

Step 3: Enable GitHub Pages

1. Go to your repository on GitHub
2. Click **Settings** (in repository tabs)
3. Scroll to **Pages** (in left sidebar)
4. Under **Source**, select:
    - **Branch**: `main` (or `master`)
    - **Folder**: `/ (root)`
5. Click **Save**
6. Your site will be available at: `https://USERNAME.github.io/REPO_NAME`

Step 4: Update Supabase Redirect URLs

1. In Supabase Dashboard → **Authentication** → **URL Configuration**
2. Add to **Redirect URLs**:
    - `https://USERNAME.github.io/REPO_NAME/**`
    - `https://USERNAME.github.io/REPO_NAME/teamridepro_v2.html`
3. Click **Save**

Connecting Custom Domain (teamridepro.com)

Since your domain is registered through Wix, you have a few options:

Option A: Use Wix DNS (Recommended if staying on Wix)

1. Log into Wix account
2. Go to **Domains** → **teamridepro.com**
3. Click **"Manage DNS"** or **"DNS Settings"**
4. Add/Edit DNS records:

For GitHub Pages:

- **Type**: CNAME
- **Name**: `@` or `www` (or both)
- **Value**: `USERNAME.github.io`
- **TTL**: 3600 (default)

For Netlify:

- **Type**: CNAME
- **Name**: `@`
- **Value**: `your-site-name.netlify.app`
- **TTL**: 3600

5. Save changes
6. Wait 24-48 hours for DNS propagation

Option B: Transfer DNS to Hosting Provider

1. In Wix, find your domain's nameservers
2. Update nameservers at your hosting provider (GitHub/Netlify)

3. Let hosting provider manage DNS

> **Note**: This gives hosting provider full control but is more flexible.

Option C: Use Subdomain (Easiest)

If DNS changes are complicated, use a subdomain:

1. In Wix DNS, add:
   - **Type**: CNAME
   - **Name**: `app` (or `team` or `admin`)
   - **Value**: `USERNAME.github.io`
2. Access site at: `app.teamridepro.com`
3. Update Supabase redirect URLs to include subdomain

Production Checklist

Security

- ☐ Supabase RLS policies are enabled and tested
- ☐ User roles are properly configured
- ☐ Admin accounts have strong passwords
- ☐ Phone authentication is configured with production SMS provider
- ☐ Email verification is enabled (optional but recommended)

Configuration

- ☐ Supabase redirect URLs include production domain
- ☐ Site URL is set to production domain in Supabase
- ☐ Environment variables are set (if using)

- ☐ API keys are configured correctly

Testing

- ☐ Admin login works (email/password)
- ☐ Coach login works (phone/SMS)
- ☐ Rider login works (phone/SMS)
- ☐ All tabs are accessible to appropriate users
- ☐ Data loads correctly from Supabase
- ☐ Data saves correctly to Supabase
- ☐ Logout works
- ☐ Session persists across page refreshes

DNS

- ☐ Domain is pointing to hosting provider
- ☐ SSL certificate is active (automatic with GitHub/Netlify/Vercel)
- ☐ Both www and non-www work (or redirect configured)

Troubleshooting Deployment

Issue: "Redirect URL mismatch"

Solution:

- Check Supabase redirect URLs include your production domain
- Ensure URL format matches exactly (including https://)
- Wait a few minutes after updating (cache)

Issue: "Site not loading"

Solution:

- Check DNS propagation (can take 24-48 hours)
- Verify CNAME/A records are correct
- Check hosting provider status page

Issue: "API key errors"

Solution:

- Verify SUPABASE_URL and SUPABASE_ANON_KEY are correct
- Check environment variables are set (if using)
- Ensure config file is loaded before other scripts

End of Guide

Good luck with your Supabase integration! 🚵

For questions or issues, refer to the troubleshooting sections in each guide or check the Supabase documentation.