🍱 OZ Kitchen Backend System Design with Supabase & Partner Integration

---

📋 Executive Summary

Project: OZ Kitchen Backend with Partner Integration and Student Super App Referral System
Technology Stack: Supabase + PostgreSQL + Edge Functions
Integration Partner: Student Super App (15% Commission System)
Privacy Compliance: Anonymous referral tracking, GDPR-compliant
Payment Gateways: Telebirr, Chapa
External Services: SMS/Email API, Delivery API

🏗️ 1. System Architecture Overview

```
None
mermaid
External Services

Partner Integration Layer

Supabase Backend Core

OZ Kitchen Frontend

Partner Ecosystem

Referral Links

Student Super App

Other Partner Apps
```

```
React Frontend

Telegram Mini App

Admin Dashboard

Supabase Auth

PostgreSQL Database

Row Level Security

Real-time Subscriptions

Edge Functions

Storage Buckets

Referral System

Commission Engine

Partner APIs

Telebirr API

Chapa Payment API

SMS/Email Service

Delivery Service API
```

🗄️ 2. Database Schema Design

Core Tables

SQL

```
-- Profiles
CREATE TABLE public.profiles (
    id UUID REFERENCES auth.users(id) PRIMARY KEY,
    first_name TEXT,
    last_name TEXT,
    phone_number TEXT UNIQUE,
    telegram_id BIGINT UNIQUE,
    delivery_address JSONB,
    preferences JSONB,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Meals, Categories, and Subscriptions
CREATE TABLE public.meal_categories (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    image_url TEXT,
    is_active BOOLEAN DEFAULT true,
    sort_order INTEGER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE public.meals (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    category_id UUID REFERENCES meal_categories(id),
    base_price DECIMAL(10,2) NOT NULL,
    image_url TEXT,
    ingredients TEXT[],
    nutritional_info JSONB,
    dietary_tags TEXT[],
    preparation_time INTEGER,
    is_available BOOLEAN DEFAULT true,
    availability_schedule JSONB,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE public.subscription_plans (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name TEXT NOT NULL,
    duration_days INTEGER NOT NULL,
    meals_per_week INTEGER NOT NULL,
    base_price DECIMAL(10,2) NOT NULL,
    discount_percentage DECIMAL(5,2) DEFAULT 0,
    is_active BOOLEAN DEFAULT true,
    features JSONB,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE public.user_subscriptions (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    user_id UUID REFERENCES profiles(id) NOT NULL,
    plan_id UUID REFERENCES subscription_plans(id) NOT NULL,
    status TEXT CHECK (status IN
('active','paused','cancelled','expired')) DEFAULT 'active',
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    budget_limit DECIMAL(10,2),
    auto_renew BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Partner Integration Tables

```sql
CREATE TABLE public.partners (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name TEXT NOT NULL,
    partner_code TEXT UNIQUE NOT NULL,
    api_key TEXT UNIQUE NOT NULL,
    commission_rate DECIMAL(5,2) NOT NULL DEFAULT 15.00,
    status TEXT CHECK (status IN
('active','suspended','inactive')) DEFAULT 'active',
    contact_email TEXT,
    webhook_url TEXT,
    settings JSONB DEFAULT '{}',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE public.referrals (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    partner_id UUID REFERENCES partners(id) NOT NULL,
    referral_token TEXT NOT NULL,
    anon_user_id TEXT,
    user_id UUID REFERENCES profiles(id),
    status TEXT CHECK (status IN
('pending','converted','expired')) DEFAULT 'pending',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE public.partner_commissions (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    partner_id UUID REFERENCES partners(id) NOT NULL,
    referral_id UUID REFERENCES referrals(id),
    payment_id UUID REFERENCES payments(id) NOT NULL,
    payment_amount DECIMAL(10,2) NOT NULL,
    commission_rate DECIMAL(5,2) NOT NULL,
    commission_amount DECIMAL(10,2) NOT NULL,
    status TEXT CHECK (status IN
('pending','approved','paid','reversed')) DEFAULT 'pending',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

🔐 3. Security & Privacy Implementation

TypeScript
```typescript
function generateReferralToken(partnerId: string) {
  const payload = { partner_id: partnerId, timestamp: Date.now()
};
  const secret = process.env.REFERRAL_TOKEN_SECRET;
  return jwt.sign(payload, secret, { algorithm: 'HS256' });
}
```

Row Level Security (RLS) Policies:

SQL
```sql
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE orders ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Users can view own profile" ON profiles FOR SELECT
USING (auth.uid() = id);
CREATE POLICY "Users can update own profile" ON profiles FOR
UPDATE USING (auth.uid() = id);
CREATE POLICY "Admins can view all data" ON profiles FOR ALL
USING (EXISTS (SELECT 1 FROM admin_users WHERE id = auth.uid()));
```

🌐 4. API Endpoints

Authentication

TypeScript
```
POST /auth/signup
POST /auth/signin
POST /auth/signout
POST /auth/telegram
```

Meals & Subscriptions

TypeScript
```
GET /api/meals
GET /api/meals/:id
GET /api/categories
POST /api/subscriptions
GET /api/subscriptions/current
```

Orders & Payments

TypeScript
```
POST /api/orders
GET /api/orders/:id
POST /api/payments/initiate
POST /api/payments/callback
POST /api/payments/verify
```

Partner APIs

TypeScript
```
POST /api/v1/referrals/capture
GET /api/v1/partners/:id/summary
GET /api/v1/partners/:id/ledger
```

⚙️ 5. Supabase Edge Functions

Payment Processing

TypeScript
```
serve(async (req) => {
  const { orderId, paymentMethod, amount } = await req.json();
  // Integrate with Telebirr or Chapa
  return new Response(JSON.stringify({ success: true }));
});
```

Order Updates

TypeScript
```typescript
serve(async (req) => {
  const { orderId, status } = await req.json();
  // Update status and send notifications
  return new Response(JSON.stringify({ success: true }));
});
```

🔄 6. Real-time Features

TypeScript
```typescript
const orderSubscription = supabase
  .channel('order-updates')
  .on('postgres_changes', { event: 'UPDATE', table: 'orders' },
(payload) => {
    // UI update logic
  })
  .subscribe();
```

☁️ 7. Storage Configuration

SQL
```sql
CREATE BUCKET meal-images WITH (public = true);
CREATE BUCKET delivery-proofs WITH (public = false);
CREATE BUCKET profile-pictures WITH (public = true);
```

🧩 8. Partner Commission System Logic

SQL
```sql
CREATE OR REPLACE FUNCTION calculate_commission() RETURNS TRIGGER
AS $$
```

```
BEGIN
  IF NEW.status = 'completed' THEN
    INSERT INTO partner_commissions (partner_id, payment_id,
payment_amount, commission_amount)
    VALUES (r.partner_id, NEW.id, NEW.amount, NEW.amount * 0.15);
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

🧠 9. Monitoring & Analytics

- Query performance and connection pool monitoring
- Payment success rates and delivery metrics
- Partner commission audit logs
- Supabase log-based alerting

🚀 10. Deployment & Environment

```
None
SUPABASE_URL=your-url
SUPABASE_SERVICE_ROLE_KEY=your-role-key
TELEBIRR_API_KEY=your-key
CHAPA_API_KEY=your-key
REFERRAL_TOKEN_SECRET=super-secret
```

Deployment Strategy:

- Local dev → Staging → Production Supabase project

- PITR backups and read replicas enabled

🎯 Success Metrics

Metric	Target	Status
Referral Conversion Rate	>25%	🎯
Commission Accuracy	99.9%	🎯
API Response Time	<200ms	🎯
Partner Satisfaction	>4.5/5	🎯
Data Privacy Compliance	100%	🎯

Document Version: 2.0
Prepared By: Backend Development Team
Approved By: Technical Lead & Product Manager