Table of Contents

# PNG GREEN FEES SYSTEM

## TECHNICAL DOCUMENTATION

**Document Version:** 1.0  
**Date:** October 2025  
**System Version:** Production Release  
**Prepared by:** PNG Green Fees Development Team

## TABLE OF CONTENTS

1. [SYSTEM ARCHITECTURE](#system-architecture)
2. [DATABASE SCHEMA](#database-schema)
3. [API DOCUMENTATION](#api-documentation)
4. [DEPLOYMENT GUIDE](#deployment-guide)
5. [SECURITY IMPLEMENTATION](#security-implementation)
6. [PERFORMANCE OPTIMIZATION](#performance-optimization)
7. [MAINTENANCE PROCEDURES](#maintenance-procedures)
8. [TROUBLESHOOTING GUIDE](#troubleshooting-guide)

## SYSTEM ARCHITECTURE

### Technology Stack

**Frontend:** - React 18.2.0 - Vite 4.5.14 - TypeScript - Tailwind CSS - Shadcn UI Components - React Router DOM - React Query

**Backend:** - Supabase (PostgreSQL 15) - Edge Functions (Deno) - Row Level Security (RLS) - Real-time subscriptions - Authentication & Authorization

**Infrastructure:** - Cloud hosting (Supabase Cloud) - CDN for static assets - Automated backups - SSL/TLS encryption - Global edge locations

### System Components

**Frontend Application:**

src/  
├── components/ # Reusable UI components  
├── pages/ # Application pages  
├── contexts/ # React contexts  
├── lib/ # Utility functions  
├── hooks/ # Custom React hooks  
└── assets/ # Static assets

**Backend Services:**

supabase/  
├── functions/ # Edge Functions  
├── migrations/ # Database migrations  
└── config/ # Configuration files

### Data Flow Architecture

**Request Flow:** 1. User interacts with React frontend 2. API calls made to Supabase backend 3. Edge Functions process business logic 4. PostgreSQL database stores/retrieves data 5. Real-time updates sent to frontend 6. UI updates automatically

## DATABASE SCHEMA

### Core Tables

**Users & Authentication:**

-- profiles table (extends auth.users)  
CREATE TABLE profiles (  
 id UUID PRIMARY KEY REFERENCES auth.users(id),  
 email TEXT UNIQUE NOT NULL,  
 full\_name TEXT,  
 role TEXT CHECK (role IN ('Admin', 'Finance\_Manager', 'Counter\_Agent', 'Read\_Only')),  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
 updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()  
);

**Passport Management:**

-- passports table  
CREATE TABLE passports (  
 id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),  
 passport\_number TEXT UNIQUE NOT NULL,  
 surname TEXT NOT NULL,  
 given\_name TEXT NOT NULL,  
 nationality TEXT NOT NULL,  
 date\_of\_birth DATE NOT NULL,  
 gender TEXT CHECK (gender IN ('Male', 'Female')),  
 place\_of\_birth TEXT,  
 date\_of\_issue DATE,  
 place\_of\_issue TEXT,  
 file\_number TEXT,  
 email TEXT,  
 phone TEXT,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
 created\_by UUID REFERENCES profiles(id)  
);

**Transaction Management:**

-- purchases table  
CREATE TABLE purchases (  
 id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),  
 customer\_name TEXT NOT NULL,  
 passport\_id UUID REFERENCES passports(id),  
 service\_type TEXT NOT NULL,  
 amount DECIMAL(10,2) NOT NULL,  
 payment\_method TEXT NOT NULL,  
 amount\_paid DECIMAL(10,2) NOT NULL,  
 change\_due DECIMAL(10,2) DEFAULT 0,  
 receipt\_number TEXT UNIQUE NOT NULL,  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
 created\_by UUID REFERENCES profiles(id)  
);

**Quotation System:**

-- quotations table  
CREATE TABLE quotations (  
 id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),  
 quotation\_number TEXT UNIQUE NOT NULL,  
 customer\_name TEXT NOT NULL,  
 customer\_email TEXT NOT NULL,  
 customer\_phone TEXT,  
 company\_address TEXT,  
 total\_amount DECIMAL(10,2) NOT NULL,  
 status TEXT CHECK (status IN ('draft', 'sent', 'approved', 'rejected', 'converted')) DEFAULT 'draft',  
 valid\_until DATE,  
 sent\_at TIMESTAMP WITH TIME ZONE,  
 approved\_at TIMESTAMP WITH TIME ZONE,  
 approved\_by UUID REFERENCES profiles(id),  
 created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),  
 created\_by UUID REFERENCES profiles(id)  
);

### Relationships

**Foreign Key Relationships:** - passports.created\_by → profiles.id - purchases.passport\_id → passports.id - purchases.created\_by → profiles.id - quotations.approved\_by → profiles.id - quotations.created\_by → profiles.id

### Indexes

**Performance Optimization:**

-- Passport lookups  
CREATE INDEX idx\_passports\_number ON passports(passport\_number);  
CREATE INDEX idx\_passports\_created\_at ON passports(created\_at);  
  
-- Purchase queries  
CREATE INDEX idx\_purchases\_created\_at ON purchases(created\_at);  
CREATE INDEX idx\_purchases\_passport\_id ON purchases(passport\_id);  
  
-- Quotation searches  
CREATE INDEX idx\_quotations\_number ON quotations(quotation\_number);  
CREATE INDEX idx\_quotations\_status ON quotations(status);  
CREATE INDEX idx\_quotations\_created\_at ON quotations(created\_at);

## API DOCUMENTATION

### Authentication Endpoints

**Login:**

POST /auth/v1/token?grant\_type=password  
Content-Type: application/json  
  
{  
 "email": "user@example.com",  
 "password": "password123"  
}

**Logout:**

POST /auth/v1/logout  
Authorization: Bearer <access\_token>

### Passport Management

**Create Passport:**

POST /rest/v1/passports  
Authorization: Bearer <access\_token>  
Content-Type: application/json  
  
{  
 "passport\_number": "P123456",  
 "surname": "Smith",  
 "given\_name": "John",  
 "nationality": "Papua New Guinea",  
 "date\_of\_birth": "1990-01-15",  
 "gender": "Male"  
}

**Get Passports:**

GET /rest/v1/passports?select=\*&order=created\_at.desc  
Authorization: Bearer <access\_token>

**Search Passports:**

GET /rest/v1/passports?passport\_number=eq.P123456  
Authorization: Bearer <access\_token>

### Purchase Processing

**Create Purchase:**

POST /rest/v1/purchases  
Authorization: Bearer <access\_token>  
Content-Type: application/json  
  
{  
 "customer\_name": "John Smith",  
 "passport\_id": "uuid-here",  
 "service\_type": "Green Fee",  
 "amount": 50.00,  
 "payment\_method": "Cash",  
 "amount\_paid": 100.00,  
 "change\_due": 50.00  
}

### Quotation Management

**Create Quotation:**

POST /rest/v1/quotations  
Authorization: Bearer <access\_token>  
Content-Type: application/json  
  
{  
 "quotation\_number": "Q2025001",  
 "customer\_name": "ABC Corporation",  
 "customer\_email": "corporate@abc.com",  
 "total\_amount": 500.00,  
 "status": "draft"  
}

### Edge Functions

**Bulk Passport Upload:**

POST /functions/v1/bulk-passport-upload  
Authorization: Bearer <access\_token>  
Content-Type: application/json  
  
{  
 "fileData": "base64-encoded-csv",  
 "fileName": "passports.csv"  
}

**Send Quotation:**

POST /functions/v1/send-quotation  
Authorization: Bearer <access\_token>  
Content-Type: application/json  
  
{  
 "quotationId": "uuid-here",  
 "email": "customer@company.com"  
}

## DEPLOYMENT GUIDE

### Environment Setup

**Required Environment Variables:**

# Supabase Configuration  
VITE\_SUPABASE\_URL=https://your-project.supabase.co  
VITE\_SUPABASE\_ANON\_KEY=your-anon-key  
  
# Application Settings  
VITE\_APP\_NAME=PNG Green Fees System  
VITE\_APP\_VERSION=1.0.0  
NODE\_ENV=production

### Database Migration

**Apply Migrations:**

# Install Supabase CLI  
npm install -g supabase  
  
# Login to Supabase  
supabase login  
  
# Link to project  
supabase link --project-ref your-project-ref  
  
# Apply migrations  
supabase db push

### Frontend Deployment

**Build Process:**

# Install dependencies  
npm install  
  
# Build for production  
npm run build  
  
# Output will be in dist/ directory

**Nginx Configuration:**

server {  
 listen 80;  
 server\_name your-domain.com;  
 root /var/www/png-green-fees/dist;  
 index index.html;  
  
 # Handle client-side routing  
 location / {  
 try\_files $uri $uri/ /index.html;  
 }  
  
 # Static assets  
 location /assets/ {  
 expires 1y;  
 add\_header Cache-Control "public, immutable";  
 }  
  
 # Security headers  
 add\_header X-Frame-Options "SAMEORIGIN";  
 add\_header X-XSS-Protection "1; mode=block";  
 add\_header X-Content-Type-Options "nosniff";  
}

### Edge Functions Deployment

**Deploy Functions:**

# Deploy all functions  
supabase functions deploy  
  
# Deploy specific function  
supabase functions deploy function-name

**Function Configuration:**

// Example Edge Function structure  
import { serve } from 'https://deno.land/std@0.224.0/http/server.ts';  
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2';  
  
const SUPABASE\_URL = Deno.env.get('SUPABASE\_URL')!;  
const SUPABASE\_ANON\_KEY = Deno.env.get('SUPABASE\_ANON\_KEY')!;  
  
serve(async (req: Request) => {  
 // Handle CORS  
 if (req.method === 'OPTIONS') {  
 return new Response(null, { status: 200 });  
 }  
  
 try {  
 // Business logic here  
 return new Response(JSON.stringify({ success: true }), {  
 headers: { 'Content-Type': 'application/json' }  
 });  
 } catch (error) {  
 return new Response(JSON.stringify({ error: error.message }), {  
 status: 500,  
 headers: { 'Content-Type': 'application/json' }  
 });  
 }  
});

## SECURITY IMPLEMENTATION

### Row Level Security (RLS)

**Profiles Table Policy:**

-- Enable RLS  
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;  
  
-- Users can read their own profile  
CREATE POLICY "Users can view own profile" ON profiles  
 FOR SELECT USING (auth.uid() = id);  
  
-- Admins can manage all profiles  
CREATE POLICY "Admins can manage profiles" ON profiles  
 FOR ALL USING (  
 EXISTS (  
 SELECT 1 FROM profiles   
 WHERE id = auth.uid()   
 AND role = 'Admin'  
 )  
 );

**Passports Table Policy:**

-- Enable RLS  
ALTER TABLE passports ENABLE ROW LEVEL SECURITY;  
  
-- Authenticated users can read passports  
CREATE POLICY "Authenticated users can read passports" ON passports  
 FOR SELECT USING (auth.role() = 'authenticated');  
  
-- Counter agents and admins can insert passports  
CREATE POLICY "Agents can insert passports" ON passports  
 FOR INSERT WITH CHECK (  
 EXISTS (  
 SELECT 1 FROM profiles   
 WHERE id = auth.uid()   
 AND role IN ('Admin', 'Counter\_Agent')  
 )  
 );

### Data Encryption

**At Rest:** - All data encrypted using AES-256 - Automatic encryption in Supabase - Encrypted backups

**In Transit:** - TLS 1.3 for all communications - HTTPS enforcement - Certificate pinning

### Authentication Security

**Password Policy:** - Minimum 8 characters - Mixed case, numbers, symbols - Regular rotation requirements - Account lockout after failed attempts

**Session Management:** - JWT tokens with expiration - Refresh token rotation - Automatic session timeout - Secure cookie settings

## PERFORMANCE OPTIMIZATION

### Database Optimization

**Query Optimization:**

-- Use proper indexes  
CREATE INDEX CONCURRENTLY idx\_passports\_search   
ON passports USING gin(to\_tsvector('english',   
 passport\_number || ' ' || surname || ' ' || given\_name));  
  
-- Optimize complex queries  
EXPLAIN ANALYZE SELECT \* FROM passports   
WHERE to\_tsvector('english', passport\_number || ' ' || surname)   
@@ plainto\_tsquery('english', 'search term');

**Connection Pooling:** - Supabase handles connection pooling - Automatic scaling based on load - Connection limits per project

### Frontend Optimization

**Code Splitting:**

// Lazy load components  
const Passports = lazy(() => import('./pages/Passports'));  
const Reports = lazy(() => import('./pages/Reports'));  
  
// Route-based splitting  
const routes = [  
 { path: '/passports', component: Passports },  
 { path: '/reports', component: Reports }  
];

**Caching Strategy:**

// React Query for data caching  
const { data: passports } = useQuery({  
 queryKey: ['passports'],  
 queryFn: fetchPassports,  
 staleTime: 5 \* 60 \* 1000, // 5 minutes  
 cacheTime: 10 \* 60 \* 1000, // 10 minutes  
});

### CDN Configuration

**Static Asset Optimization:** - Gzip compression enabled - Brotli compression for modern browsers - Cache headers for static assets - Image optimization and WebP support

## MAINTENANCE PROCEDURES

### Regular Maintenance Tasks

**Daily:** - Monitor system performance metrics - Check error logs and alerts - Verify backup completion - Review security logs

**Weekly:** - Analyze performance trends - Review user activity logs - Update security patches - Clean temporary files

**Monthly:** - Database maintenance and optimization - Security audit and review - Performance analysis and tuning - Backup restoration testing

### Backup Procedures

**Automated Backups:** - Daily full database backups - Incremental backups every 6 hours - Point-in-time recovery available - Cross-region backup replication

**Manual Backup:**

# Create manual backup  
supabase db dump --data-only --file backup-$(date +%Y%m%d).sql  
  
# Restore from backup  
supabase db reset --file backup-20250101.sql

### Monitoring and Alerting

**Key Metrics:** - Response time < 2 seconds - Uptime > 99.9% - Error rate < 0.1% - Database connections < 80%

**Alert Conditions:** - High error rates - Slow response times - Database connection issues - Security incidents

## TROUBLESHOOTING GUIDE

### Common Issues

**Database Connection Issues:**

# Check connection status  
supabase status  
  
# Reset database  
supabase db reset  
  
# Check logs  
supabase logs db

**Edge Function Errors:**

# View function logs  
supabase functions logs function-name  
  
# Deploy function  
supabase functions deploy function-name  
  
# Test function locally  
supabase functions serve

**Frontend Build Issues:**

# Clear cache  
npm run clean  
rm -rf node\_modules package-lock.json  
npm install  
  
# Check dependencies  
npm audit  
npm audit fix  
  
# Rebuild  
npm run build

### Performance Issues

**Slow Queries:**

-- Identify slow queries  
SELECT query, mean\_time, calls   
FROM pg\_stat\_statements   
ORDER BY mean\_time DESC   
LIMIT 10;  
  
-- Analyze query performance  
EXPLAIN ANALYZE SELECT \* FROM passports WHERE passport\_number = 'P123456';

**Memory Issues:**

# Check memory usage  
free -h  
  
# Monitor processes  
top -p $(pgrep node)  
  
# Clear system cache  
echo 3 > /proc/sys/vm/drop\_caches

### Security Issues

**Unauthorized Access:** 1. Check authentication logs 2. Review user permissions 3. Verify RLS policies 4. Update security patches

**Data Breach Response:** 1. Isolate affected systems 2. Assess scope of breach 3. Notify stakeholders 4. Implement containment measures 5. Conduct forensic analysis

**END OF TECHNICAL DOCUMENTATION**

*This technical documentation provides comprehensive information for system administrators, developers, and technical support personnel working with the PNG Green Fees System.*

**Document Control:** - **Version:** 1.0 - **Last Updated:** October 2025 - **Next Review:** January 2026 - **Technical Lead:** PNG Green Fees Development Team