Restro Backend SaaS Documentation

Generated on: 2026-02-16 11:20:02

1. Overview

This backend is designed as a multi-tenant restaurant SaaS platform.

- One restaurant = one tenant.

- Owner registration creates user + tenant + owner membership + trial subscription.

- Staff are mapped through role-based memberships.

- Authentication uses access and refresh token cookies with rotating refresh sessions.

2. Roles

Supported roles: OWNER, MANAGER, KITCHEN, WAITER

3. Tenant Resolution Strategy

Restaurant tenant is resolved in this priority order:

- x-tenant-slug header

- tenantSlug in request body

- tenantSlug in query params

- subdomain from hostname (abc.myapp.com -> abc)

After login, JWT carries tenantId + role, so same URL can still safely separate restaurant data.

4. Database Design

4.x users

Field	Type	Notes
_id	ObjectId	Primary key
name	String	User name
email	String	Unique, lowercase
password	String	bcrypt hash, select false
isActive	Boolean	User status
createdAt/updatedAt	Date	Timestamps

4.x tenants

Field	Type	Notes
_id	ObjectId	Primary key
name	String	Restaurant name
slug	String	Unique URL-safe id
status	Enum	ACTIVE/SUSPENDED
ownerUserId	ObjectId	Ref users._id
createdAt/updatedAt	Date	Timestamps

4.x memberships

Field	Type	Notes

userId	ObjectId	Ref users._id
tenantId	ObjectId	Ref tenants._id
role	Enum	OWNER/MANAGER/KITCHEN/WAITER
isActive	Boolean	Membership status

4.x subscriptions

Field	Type	Notes
tenantId	ObjectId	Unique Ref tenants._id
planCode	String	TRIAL or paid plan code
status	Enum	TRIAL/ACTIVE/PAST_DUE/CANCELED/EXPIRED
startsAt	Date	Subscription start
endsAt	Date	Subscription end

4.x refreshsessions

Field	Type	Notes
_id	ObjectId	Session id used as JWT sid
userId	ObjectId	Ref users._id
tenantId	ObjectId	Ref tenants._id
role	String	Role for session scope
tokenHash	String	sha256(refresh token)
expiresAt	Date	TTL indexed
revokedAt	Date	Null unless revoked

5. API Endpoints

Method	Path	Access
GET	/api/health	Public
POST	/api/auth/register-owner	Public
POST	/api/auth/register	Public (alias)
POST	/api/auth/login	Public
POST	/api/auth/refresh	Public (cookie required)
POST	/api/auth/logout	Authenticated session cookie
GET	/api/auth/me	Authenticated
GET	/api/auth/staff-roles	Public
GET	/api/tenant/staff	OWNER or MANAGER + active subscription
POST	/api/tenant/staff	OWNER or MANAGER + active subscription

6. Sample Requests

6.1 Register Owner

{"name":"Uday","email":"uday@example.com","password":"StrongPass123","restaurantName":"Spicy Hub","restaurantSlug":"spicy-hub"}

6.2 Login

{"email":"waiter@example.com","password":"StrongPass123","role":"WAITER","tenantSlug":"spicy-hub"}

6.3 Create Staff

{"name":"Ravi","email":"ravi.waiter@example.com","password":"StrongPass123","role":"WAITER"}

7. Environment Variables

- Required: PORT, MONGO_URI (or MONGO_URL), JWT_ACCESS_SECRET, JWT_REFRESH_SECRET

- Optional: ACCESS_TOKEN_EXPIRES_IN, REFRESH_TOKEN_EXPIRES_IN, COOKIE_SECURE, COOKIE_DOMAIN

8. Production Checklist

- Enable HTTPS and set COOKIE_SECURE=true

- Store secrets in vault/secret manager

- Add rate limiting to auth routes

- Add audit logs and invitation workflow

- Wire billing webhook to update subscription status