GENFITY Online Ordering System

Dokumentasi Lengkap: Desain Database, Sistem, dan Backend

Daftar Isi

1. Gambaran Besar Modul Data {#gambaran-besar}

Sistem GENFITY Online Ordering dibangun dengan modularitas tinggi, memisahkan setiap aspek bisnis ke dalam tabel-tabel terstruktur. Berikut adalah blok utama data:

Auth & Roles

- **users**: Menyimpan semua akun (Super Admin, Merchant Owner, Merchant Staff, Customer)
- **user_sessions**: Menyimpan sesi login dengan JWT tracking dan status aktif/revoked
- **merchant_users**: Mapping user dengan merchant (relasi many-to-many) dengan role OWNER atau STAFF

Merchant & Konfigurasi

- **merchants**: Profil merchant (nama, kode, alamat, pajak, jam buka)
- **merchant_opening_hours**: Jam operasional per hari (senin-minggu)

Menu & Addons

- **menu_categories**: Kategori menu per merchant
- **menus**: Item menu dengan harga, stok, foto
- **addon_categories**: Kategori tambahan/opsi (misal: "Toppings", "Sauce Level")
- **menu_addon_categories**: Relasi many-to-many antara menu dan addon_categories

- **addon_items**: Item opsi tambahan dengan harga

Order & Checkout

- **orders**: Header pesanan dengan status, total, customer info
- **order_items**: Item yang dipesan (snapshot menu name & price untuk invoice history)
- **order_item_addons**: Addon yang dipilih untuk setiap item
- **order_status_history**: Riwayat perubahan status untuk tracking

Currency & Monetary

- Semua nilai uang menggunakan tipe **NUMERIC(10,2)** dalam satuan AUD
- Snapshot harga di tabel order memastikan history tetap akurat meski harga berubah

2. Relasi Inti (ERD) {#relasi-inti}

Diagram hubungan antar entitas:

```
users (1) ──→ (N) merchant_users
                      ↓
                  merchants

merchants (1) ──→ (N) merchant_opening_hours
merchants (1) ──→ (N) menu_categories ──→ (N) menus
merchants (1) ──→ (N) addon_categories ──→ (N) addon_items
                      ↑
                      └── menu_addon_categories ──← (N) menus

merchants (1) ──→ (N) orders ←─── (1) users (role=CUSTOMER)
orders    (1) ──→ (N) order_items ──→ (1) menus (via menu_id snapshot)
order_items (1) ──→ (N) order_item_addons ──→ (1) addon_items (snapshot)
orders    (1) ──→ (N) order_status_history

users     (1) ──→ (N) user_sessions
```

Penjelasan Relasi Kunci:

1. **users ↔ merchant_users ↔ merchants**: Satu user bisa mengelola multiple merchants (owner beberapa outlet), satu merchant bisa punya beberapa staff.
2. **menus ↔ menu_addon_categories ↔ addon_categories**: Addon category (misal "Extra Sauce") bisa digunakan di berbagai menu tanpa duplikasi data.
3. **orders ↔ order_items ↔ order_item_addons**: Struktur standar cart dengan snapshot data untuk historical accuracy.
4. **user_sessions**: Memastikan setiap JWT token dapat di-track, di-revoke, dan di-validasi terhadap session yang aktif.

3. Detail Tabel dan SQL PostgreSQL {#detail-tabel}

3.1 Auth & Access Control

Tabel: users

```
CREATE TABLE users (
    id              BIGSERIAL PRIMARY KEY,
    name            VARCHAR(150) NOT NULL,
    email           VARCHAR(255) UNIQUE NOT NULL,
    phone           VARCHAR(50),
    password_hash   VARCHAR(255),  -- bcrypt hash, 10 rounds
    role            VARCHAR(30) NOT NULL CHECK (
                        role IN ('SUPER_ADMIN', 'MERCHANT_OWNER', 'MERCHANT_STAFF', 'CUST
                    ),
    is_active       BOOLEAN NOT NULL DEFAULT TRUE,

    -- optional: untuk logging/audit
    last_login_at   TIMESTAMPTZ,
    last_login_ip   VARCHAR(100),

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
```

Tabel: user_sessions

```
CREATE TABLE user_sessions (
    id              BIGSERIAL PRIMARY KEY,            -- ini yang dipakai sebagai "sid" c

    user_id         BIGINT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    merchant_id     BIGINT REFERENCES merchants(id) ON DELETE SET NULL,  -- optional: con

    -- token identity (bukan token mentah)
    access_jti      VARCHAR(255),                   -- JWT ID unique per token
    refresh_token_hash VARCHAR(255),                -- hash dari refresh token (opsional

    -- connection info
    ip_address      VARCHAR(100),
    user_agent      TEXT,
    device_info     TEXT,                           -- misal: "Chrome on Windows", "Safa

    -- status tracking
    status          VARCHAR(20) NOT NULL CHECK (
                        status IN ('ACTIVE', 'REVOKED', 'EXPIRED')
                    ) DEFAULT 'ACTIVE',

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    expires_at      TIMESTAMPTZ NOT NULL,           -- sama dengan exp di JWT accessToke
```

```
    revoked_at      TIMESTAMPTZ,
    revoked_reason  TEXT,

    -- optional: untuk refresh token
    refresh_expires_at TIMESTAMPTZ
);

CREATE INDEX idx_user_sessions_user_id ON user_sessions(user_id);
CREATE INDEX idx_user_sessions_status ON user_sessions(status);
CREATE INDEX idx_user_sessions_access_jti ON user_sessions(access_jti);
```

Tabel: merchant_users

```
CREATE TABLE merchant_users (
    id              BIGSERIAL PRIMARY KEY,
    merchant_id     BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,
    user_id         BIGINT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role            VARCHAR(30) NOT NULL CHECK (role IN ('OWNER', 'STAFF')),

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    UNIQUE (merchant_id, user_id)
);

CREATE INDEX idx_merchant_users_merchant_id ON merchant_users(merchant_id);
CREATE INDEX idx_merchant_users_user_id ON merchant_users(user_id);
```

3.2 Merchant & Konfigurasi

Tabel: merchants

```
CREATE TABLE merchants (
    id                  BIGSERIAL PRIMARY KEY,
    name                VARCHAR(150) NOT NULL,
    code                VARCHAR(50) NOT NULL UNIQUE,  -- untuk /[merchantCode]
    description         TEXT,
    logo_url            TEXT,
    cover_image_url     TEXT,

    -- lokasi
    address_line        TEXT,
    city                VARCHAR(100),
    state               VARCHAR(100),
    postcode            VARCHAR(20),
    country             VARCHAR(100) DEFAULT 'Australia',
    phone               VARCHAR(50),
    maps_url            TEXT,

    -- konfigurasi operasional
    is_open             BOOLEAN NOT NULL DEFAULT TRUE,  -- manual buka/tutup
    tax_enabled         BOOLEAN NOT NULL DEFAULT FALSE,
```

```
    tax_rate_percent    NUMERIC(5,2) DEFAULT 0.00,     -- contoh: 10.00 = 10%
    currency_code       VARCHAR(10) NOT NULL DEFAULT 'AUD',

    created_by_user_id  BIGINT NOT NULL REFERENCES users(id),  -- super admin yang buat

    created_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at          TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_merchants_code ON merchants(code);
CREATE INDEX idx_merchants_created_by ON merchants(created_by_user_id);
```

Tabel: merchant_opening_hours

```
CREATE TABLE merchant_opening_hours (
    id              BIGSERIAL PRIMARY KEY,
    merchant_id     BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,

    weekday         SMALLINT NOT NULL CHECK (weekday BETWEEN 0 AND 6),  -- 0=Sunday, 6=Sa
    open_time       TIME,
    close_time      TIME,
    is_closed       BOOLEAN NOT NULL DEFAULT FALSE,

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    UNIQUE (merchant_id, weekday)
);
```

3.3 Menu & Addons

Tabel: menu_categories

```
CREATE TABLE menu_categories (
    id              BIGSERIAL PRIMARY KEY,
    merchant_id     BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,

    name            VARCHAR(100) NOT NULL,
    description     TEXT,
    sort_order      INTEGER NOT NULL DEFAULT 0,
    is_active       BOOLEAN NOT NULL DEFAULT TRUE,

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_menu_categories_merchant_id ON menu_categories(merchant_id);
```

Tabel: menus

```
CREATE TABLE menus (
    id              BIGSERIAL PRIMARY KEY,
    merchant_id     BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,
    category_id     BIGINT NOT NULL REFERENCES menu_categories(id) ON DELETE CASCADE,

    name            VARCHAR(150) NOT NULL,
    description     TEXT,
    price           NUMERIC(10,2) NOT NULL,           -- harga base
    photo_url       TEXT,

    is_active       BOOLEAN NOT NULL DEFAULT TRUE,
    is_featured     BOOLEAN NOT NULL DEFAULT FALSE,    -- untuk promo/header

    track_stock     BOOLEAN NOT NULL DEFAULT FALSE,
    stock_qty       INTEGER,                           -- nullable = unlimited stok

    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_menus_merchant_id ON menus(merchant_id);
CREATE INDEX idx_menus_category_id ON menus(category_id);
CREATE INDEX idx_menus_is_active ON menus(is_active);
```

Tabel: addon_categories

```
CREATE TABLE addon_categories (
    id              BIGSERIAL PRIMARY KEY,
    merchant_id     BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,

    name            VARCHAR(100) NOT NULL,
    description     TEXT,
    min_select      INTEGER NOT NULL DEFAULT 0,        -- minimal pilihan
    max_select      INTEGER,                           -- maksimal pilihan
    is_required     BOOLEAN NOT NULL DEFAULT FALSE,    -- apakah wajib dipilih

    sort_order      INTEGER NOT NULL DEFAULT 0,
    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_addon_categories_merchant_id ON addon_categories(merchant_id);
```

Tabel: menu_addon_categories

```
CREATE TABLE menu_addon_categories (
    id                  BIGSERIAL PRIMARY KEY,
    menu_id             BIGINT NOT NULL REFERENCES menus(id) ON DELETE CASCADE,
    addon_category_id   BIGINT NOT NULL REFERENCES addon_categories(id) ON DELETE CASCADE
```

```
    UNIQUE (menu_id, addon_category_id)
);

CREATE INDEX idx_menu_addon_categories_menu_id ON menu_addon_categories(menu_id);
CREATE INDEX idx_menu_addon_categories_addon_category_id ON menu_addon_categories(addon_c
```

Tabel: addon_items

```
CREATE TABLE addon_items (
    id                  BIGSERIAL PRIMARY KEY,
    addon_category_id   BIGINT NOT NULL REFERENCES addon_categories(id) ON DELETE CASCADE

    name                VARCHAR(150) NOT NULL,
    description         TEXT,
    price               NUMERIC(10,2) NOT NULL,

    track_stock         BOOLEAN NOT NULL DEFAULT FALSE,
    stock_qty           INTEGER,
    max_per_order_item  INTEGER,                    -- max qty addon per 1 menu item

    is_active           BOOLEAN NOT NULL DEFAULT TRUE,
    sort_order          INTEGER NOT NULL DEFAULT 0,

    created_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at          TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_addon_items_addon_category_id ON addon_items(addon_category_id);
```

3.4 Orders & Checkout

Tabel: orders

```
CREATE TABLE orders (
    id                  BIGSERIAL PRIMARY KEY,
    merchant_id         BIGINT NOT NULL REFERENCES merchants(id) ON DELETE CASCADE,
    customer_id         BIGINT NOT NULL REFERENCES users(id),  -- role = CUSTOMER

    -- nomor order unik per merchant (misal: "GF-000001")
    order_number        VARCHAR(50) NOT NULL,
    UNIQUE (merchant_id, order_number),

    -- mode pemesanan
    order_mode          VARCHAR(20) NOT NULL CHECK (
                            order_mode IN ('DINE_IN', 'TAKEAWAY')
                        ),
    table_number        VARCHAR(20),                -- required untuk DINE_IN

    -- status order
    status              VARCHAR(30) NOT NULL CHECK (
                            status IN ('PENDING', 'ACCEPTED', 'IN_PROGRESS', 'READY', 'CO
                        ) DEFAULT 'PENDING',
```

```
    -- payment status
    payment_status      VARCHAR(30) NOT NULL CHECK (
                            payment_status IN ('UNPAID', 'PAID', 'REFUNDED', 'CANCELLED')
                        ) DEFAULT 'UNPAID',

    payment_method      VARCHAR(30) NOT NULL DEFAULT 'PAY_AT_COUNTER',
    CHECK (payment_method = 'PAY_AT_COUNTER'),         -- dibatasi hanya kasir

    -- snapshot data customer untuk invoice/history
    customer_name       VARCHAR(150) NOT NULL,
    customer_email      VARCHAR(255) NOT NULL,
    customer_phone      VARCHAR(50),

    -- ringkasan nominal
    subtotal_amount     NUMERIC(10,2) NOT NULL DEFAULT 0.00,
    tax_amount          NUMERIC(10,2) NOT NULL DEFAULT 0.00,
    total_amount        NUMERIC(10,2) NOT NULL DEFAULT 0.00,

    tax_rate_percent    NUMERIC(5,2) DEFAULT 0.00,    -- snapshot tax saat order dibuat
    currency_code       VARCHAR(10) NOT NULL DEFAULT 'AUD',

    -- catatan
    notes               TEXT,                          -- catatan customer
    internal_note       TEXT,                          -- catatan internal merchant

    -- QR code &amp;amp; timestamp
    qr_code_data        TEXT,                          -- encoded string (order_number at
    placed_at           TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    created_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at          TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_orders_merchant_id ON orders(merchant_id);
CREATE INDEX idx_orders_customer_id ON orders(customer_id);
CREATE INDEX idx_orders_order_number ON orders(order_number);
CREATE INDEX idx_orders_status ON orders(status);
CREATE INDEX idx_orders_placed_at ON orders(placed_at);
```

Tabel: order_items

```
CREATE TABLE order_items (
    id                  BIGSERIAL PRIMARY KEY,
    order_id            BIGINT NOT NULL REFERENCES orders(id) ON DELETE CASCADE,
    menu_id             BIGINT REFERENCES menus(id),   -- nullable buat referensi

    -- snapshot info menu (kalau harga/nama berubah, history tetap akurat)
    menu_name           VARCHAR(150) NOT NULL,
    menu_description    TEXT,
    unit_price          NUMERIC(10,2) NOT NULL,
    quantity            INTEGER NOT NULL CHECK (quantity &gt; 0),

    -- total
    line_subtotal       NUMERIC(10,2) NOT NULL DEFAULT 0.00,  -- unit_price * qty (belum
```

```
    line_total          NUMERIC(10,2) NOT NULL DEFAULT 0.00,  -- sudah + addons

    -- catatan spesial item
    notes               TEXT,

    created_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at          TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_order_items_order_id ON order_items(order_id);
CREATE INDEX idx_order_items_menu_id ON order_items(menu_id);
```

Tabel: order_item_addons

```
CREATE TABLE order_item_addons (
    id                  BIGSERIAL PRIMARY KEY,
    order_item_id       BIGINT NOT NULL REFERENCES order_items(id) ON DELETE CASCADE,
    addon_item_id       BIGINT REFERENCES addon_items(id),     -- nullable buat referensi

    -- snapshot addon info
    addon_name          VARCHAR(150) NOT NULL,
    unit_price          NUMERIC(10,2) NOT NULL,
    quantity            INTEGER NOT NULL CHECK (quantity &gt; 0),
    line_total          NUMERIC(10,2) NOT NULL DEFAULT 0.00,

    created_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at          TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_order_item_addons_order_item_id ON order_item_addons(order_item_id);
CREATE INDEX idx_order_item_addons_addon_item_id ON order_item_addons(addon_item_id);
```

Tabel: order_status_history

```
CREATE TABLE order_status_history (
    id                  BIGSERIAL PRIMARY KEY,
    order_id            BIGINT NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

    from_status         VARCHAR(30),
    to_status           VARCHAR(30) NOT NULL,

    changed_by_user_id  BIGINT REFERENCES users(id),   -- merchant/staff yang ubah status
    changed_at          TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    note                TEXT
);

CREATE INDEX idx_order_status_history_order_id ON order_status_history(order_id);
CREATE INDEX idx_order_status_history_changed_at ON order_status_history(changed_at);
```

4. Authentication dengan JWT dan User Sessions {#auth-jwt}

4.1 Konsep Auth Flow

Modern authentication untuk GENFITY menggunakan JWT (JSON Web Token) dengan session tracking:

1. **User Login**: Email + Password
2. **Backend Verification**:
   - Cek password hash
   - Create user_session record di DB
   - Generate JWT accessToken dengan claim `sid` (session id)
3. **Client Storage**: AccessToken disimpan di localStorage atau SessionStorage
4. **Request Cycle**: Setiap request include `Authorization: Bearer &lt;accessToken&gt;`
5. **Server Validation**:
   - Verify JWT signature dan expiration
   - Cek session di DB masih ACTIVE
   - Validate bahwa session belum di-revoke
6. **Logout**: Update `user_sessions.status = 'REVOKED'` → token otomatis invalid

4.2 JWT Payload Structure

```
{
  "sub": "123",                    // user_id
  "sid": "456",                    // session_id (dari user_sessions.id)
  "role": "MERCHANT_OWNER",        // dari users.role
  "mid": "789",                    // merchant_id (opsional, untuk context)
  "jti": "abc-def-ghi",            // JWT ID (untuk tracking)
  "iat": 1699500000,               // issued at
  "exp": 1699503600                // expires at (misal: 1 jam)
}
```

4.3 Middleware Validasi Token

```
Pseudo-code:
┌─────────────────────────────────────────┐
│ 1. Extract token dari header    │
│    Authorization: Bearer &amp;lt;token&amp;gt;│
└─────────────────────────────────────────┘
        ↓
┌─────────────────────────────────────────┐
│ 2. Verify JWT signature &amp;amp; exp   │
│    Gunakan secret key           │
└─────────────────────────────────────────┘
        ↓ (gagal) → 401
```

```
        ↓ (sukses)
┌─────────────────────────────────────────┐
│ 3. Extract sid dari payload     │
└─────────────────────────────────────────┘
        ↓
┌──────────────────────────────────────────────────────────┐
│ 4. Query DB:                                  │
│    SELECT * FROM user_sessions                │
│    WHERE id = :sid AND status = 'ACTIVE'    │
│    AND expires_at &gt; NOW()                    │
└──────────────────────────────────────────────────────────┘
        ↓ (not found/expired/revoked) → 401
        ↓ (found &amp; active)
┌─────────────────────────────────────────┐
│ 5. Set request context:          │
│    req.user.id = payload.sub     │
│    req.user.role = payload.role  │
│    req.sessionId = payload.sid   │
└─────────────────────────────────────────┘
        ↓
        ✓ Lanjut ke handler
```

4.4 Email Notifikasi Password untuk Register

Ketika super admin membuat akun merchant atau staff baru:

1. **Generate Temporary Password**:
   - Random string 12-16 karakter (mix alphanumeric + special chars)
   - Contoh: `GenF1ty@2024#Tmp`
2. **Hash Password**:

```
bcrypt_hash = bcrypt.hash(temp_password, 10)
INSERT INTO users (email, password_hash, ...) VALUES (..., bcrypt_hash, ...)
```

3. **Kirim Email**:

```
Subject: GENFITY - Akun Merchant Baru Anda

Halo [Nama Merchant],

Selamat! Akun Anda telah dibuat di platform GENFITY.

Berikut adalah credentials login Anda:
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
Email: merchant@example.com
Password: GenF1ty@2024#Tmp
Link Dashboard: https://genfity.com/dashboard
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

 PENTING: Ubah password Anda saat login pertama kali!
Jangan bagikan password ini kepada siapapun.

Jika ada pertanyaan, hubungi support@genfity.com
```

```
Salam,
Tim GENFITY
```

4. **Force Change Password**:
   - Tambah flag opsional di users: `must_change_password BOOLEAN DEFAULT TRUE`
   - Saat login pertama: system redirect ke halaman change password
   - Setelah ubah: set `must_change_password = FALSE`

4.5 Optional: Refresh Token Flow

Jika di masa depan ingin upgrade dengan refresh token:

```
-- Existing: user_sessions sudah ada
-- Tambahan field:
ALTER TABLE user_sessions
ADD COLUMN refresh_expires_at TIMESTAMPTZ;

-- Saat login: generate 2 token
{
  "accessToken": "eyJ...",        -- short-lived (15 menit)
  "refreshToken": "xyz...",       -- long-lived (7 hari)
  "expiresIn": 900                -- 15 menit dalam detik
}

-- Refresh flow:
POST /api/auth/refresh
  body: { refreshToken: "xyz..." }

  ↓ (validate refresh token, cek session)
  ↓ (generate new accessToken)

  response: { accessToken: "new...", expiresIn: 900 }
```

5. Flow Bisnis vs Database {#flow-bisnis}

Berikut validasi bahwa skema DB nyambung dengan business flow:

Scenario 1: Super Admin Membuat Merchant Baru

```
1. Super admin form: name, email, merchant_code, password_temp
2. Backend:
   - Insert users:
     INSERT INTO users (name, email, password_hash, role, is_active)
     VALUES (name, email, bcrypt(password_temp), 'MERCHANT_OWNER', true)
   - Insert merchants:
     INSERT INTO merchants (name, code, created_by_user_id)
     VALUES (name, code, super_admin_user_id)
   - Insert merchant_users:
     INSERT INTO merchant_users (merchant_id, user_id, role)
```

```
     VALUES (merchant_id, user_id, 'OWNER')
   - Send email dengan temp password
3. Merchant login dengan email &amp;amp; password_temp
4. System prompt: Change password
5. Merchant bisa setup dashboard
```

✅ **Database supports**: users + merchant_users + merchants + email flow

Scenario 2: Merchant Setup Menu

```
1. Merchant create kategori:
   INSERT INTO menu_categories (merchant_id, name) VALUES (...)
2. Merchant create menu:
   INSERT INTO menus (merchant_id, category_id, name, price, ...) VALUES (...)
3. Merchant create addon category (misal: "Toppings"):
   INSERT INTO addon_categories (merchant_id, name, max_select=5) VALUES (...)
4. Merchant add opsi addon:
   INSERT INTO addon_items (addon_category_id, name, price) VALUES (...)
5. Link addon ke menu (misal menu Burger bisa tambah Toppings):
   INSERT INTO menu_addon_categories (menu_id, addon_category_id) VALUES (...)
```

✅ **Database supports**: menu_categories + menus + addon_categories + addon_items + menu_addon_categories

Scenario 3: Customer Browse & Order

```
1. Customer akses genfity.com/[merchantCode]
2. Frontend resolve merchant:
   SELECT * FROM merchants WHERE code = :merchantCode
3. Customer pilih mode (DINE_IN / TAKEAWAY)
   - If DINE_IN: minta table_number
   - Simpan di localStorage
4. Customer browse menu &amp;amp; addons:
   SELECT mc.*, m.* FROM menu_categories mc
   JOIN menus m ON mc.id = m.category_id
   WHERE m.merchant_id = :merchant_id AND m.is_active = TRUE

   SELECT mac.*, ai.* FROM menu_addon_categories mac
   JOIN addon_items ai ON mac.addon_category_id = ai.addon_category_id
   WHERE mac.menu_id = :menu_id
5. Customer add to cart (localStorage), then checkout
6. If belum login: form isi name, email, phone
   - Backend INSERT users (role='CUSTOMER')
7. Backend create order:
   INSERT INTO orders (
     merchant_id, customer_id, order_mode, table_number,
     customer_name, customer_email, customer_phone,
     subtotal_amount, tax_amount, total_amount, tax_rate_percent
   ) VALUES (...)

   INSERT INTO order_items (order_id, menu_id, menu_name, unit_price, quantity, ...)
   INSERT INTO order_item_addons (order_item_id, addon_name, unit_price, quantity, ...)
```

```
   Generate order_number = "GF-000001"
   Generate qr_code_data (encode order_number or order_id)
   UPDATE orders SET order_number=..., qr_code_data=...

   Create session (optional) buat auto-login
8. Return order summary + QR code ke customer
```

✓ **Database supports**: semua step di atas

Scenario 4: Merchant Manage Orders

```
1. Merchant login
2. Dashboard query:
   SELECT * FROM orders
   WHERE merchant_id = :merchant_id
   AND status IN ('PENDING', 'ACCEPTED', 'IN_PROGRESS')
   ORDER BY placed_at DESC
3. Merchant click order → lihat order_items + order_item_addons (snapshot data)
4. Merchant ubah status (misal PENDING → ACCEPTED):
   UPDATE orders SET status='ACCEPTED', updated_at=NOW()
   WHERE id=:order_id

   INSERT INTO order_status_history (
     order_id, from_status, to_status, changed_by_user_id
   ) VALUES (:order_id, 'PENDING', 'ACCEPTED', :merchant_user_id)
5. Customer bisa lihat order status berubah (real-time via polling/websocket)
```

✓ **Database supports**: orders + order_status_history + tracking

Scenario 5: Laporan Pendapatan

```
Merchant query revenue:
SELECT
  DATE(placed_at) as date,
  COUNT(*) as total_orders,
  SUM(total_amount) as revenue
FROM orders
WHERE merchant_id = :merchant_id
  AND status = 'COMPLETED'
  AND placed_at &gt;= :date_from
  AND placed_at &lt;= :date_to
GROUP BY DATE(placed_at)
ORDER BY date DESC
```

✓ **Database supports**: orders dengan placed_at dan total_amount

Scenario 6: Stok Management

```
Jika track_stock=TRUE:
Saat order sukses:
1. Update menu stock (kalau ada):
   UPDATE menus SET stock_qty = stock_qty - :qty
   WHERE id=:menu_id AND track_stock=TRUE
2. Update addon stock:
   UPDATE addon_items SET stock_qty = stock_qty - :qty
   WHERE id=:addon_id AND track_stock=TRUE
3. Check low stock / out of stock
   SELECT * FROM menus WHERE merchant_id=:merchant_id
   AND track_stock=TRUE AND (stock_qty IS NULL OR stock_qty &lt; 10)
```

✅ **Database supports**: track_stock + stock_qty fields di menus dan addon_items

6. Email Notifikasi untuk Password {#email-password}

Proses pengiriman password dilakukan saat super admin membuat akun baru (merchant/staff):

6.1 Flow Teknis

```
┌──────────────────────────────────────────────────────────────────────┐
│ 1. Super Admin Input Form                                            │
│    - Nama merchant/staff                                             │
│    - Email                                                           │
│    - Merchant code (buat merchant baru)                              │
└──────────────────────────────────────────────────────────────────────┘
        ↓
┌──────────────────────────────────────────────────────────────────────┐
│ 2. Backend POST /api/admin/merchants (create)                        │
│    - Generate random temp password                                   │
│    - Password = generateRandomPassword(16)                           │
│      Hasil: contoh "Genfity@2024#aBc1"                               │
└──────────────────────────────────────────────────────────────────────┘
        ↓
┌──────────────────────────────────────────────────────────────────────┐
│ 3. Hash Password &amp;amp; Save                                      │
│    - hash = bcrypt(temp_password, 10)                                │
│    - INSERT INTO users (email, password_hash, ...)                   │
│    - INSERT INTO merchants (...)                                     │
│    - INSERT INTO merchant_users (...)                                │
└──────────────────────────────────────────────────────────────────────┘
        ↓
┌──────────────────────────────────────────────────────────────────────┐
│ 4. Send Email                                                        │
│    - Email service (SMTP / SendGrid / AWS SES)                       │
│    - Tempate email berisi:                                           │
│      * Greeting dengan nama                                          │
│      * Email address                                                 │
│      * Temp password (plain text)                                    │
│      * Link dashboard                                                │
│      * Security reminder                                             │
```

```
│      * Support contact                                          │
└─────────────────────────────────────────────────────────────────┘
        ↓
┌─────────────────────────────────────────────────────────────────┐
│ 5. Email Delivered ✓                                            │
│    Merchant/Staff bisa langsung login                           │
└─────────────────────────────────────────────────────────────────┘
        ↓
┌─────────────────────────────────────────────────────────────────┐
│ 6. First Login                                                  │
│    - Email + password_temp                                      │
│    - Create user_session                                        │
│    - Generate JWT accessToken                                   │
│    - Backend set flag: must_change_password = TRUE              │
└─────────────────────────────────────────────────────────────────┘
        ↓
┌─────────────────────────────────────────────────────────────────┐
│ 7. Change Password (mandatory)                                  │
│    - Frontend redirect to /change-password                      │
│    - Input: password_old, password_new, confirm_new             │
│    - Backend validate old password                              │
│    - Hash new password &amp;amp; update users.password_hash     │
│    - Set must_change_password = FALSE                           │
│    - Redirect to dashboard                                      │
└─────────────────────────────────────────────────────────────────┘
        ↓
✓ Account fully activated
```

6.2 Database Schema Addition (Opsional)

```
-- Tambah field di users untuk flag change password
ALTER TABLE users
ADD COLUMN must_change_password BOOLEAN DEFAULT TRUE;

-- Track password change history (opsional untuk audit)
CREATE TABLE user_password_history (
    id              BIGSERIAL PRIMARY KEY,
    user_id         BIGINT NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    password_hash   VARCHAR(255) NOT NULL,             -- hash lama
    changed_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

6.3 Email Template HTML

```
&lt;html&gt;
&lt;head&gt;
    &lt;style&gt;
        body { font-family: Arial, sans-serif; background: #f5f5f5; }
        .container { max-width: 600px; margin: 20px auto; background: white; padding: 20p
        .header { text-align: center; border-bottom: 2px solid #ff9800; padding-bottom: 1
        .header img { height: 40px; }
        .content { padding: 20px 0; }
```

```
        .credentials { background: #f9f9f9; padding: 15px; border-left: 4px solid #ff9800
        .credentials .label { font-weight: bold; color: #333; }
        .credentials .value { font-family: monospace; color: #ff9800; font-weight: bold;
        .warning { background: #fff3cd; padding: 12px; border-radius: 4px; margin: 15px 0
        .footer { text-align: center; font-size: 12px; color: #999; border-top: 1px solid
        .button { display: inline-block; background: #ff9800; color: white; padding: 12px
    &lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;
    <div>
        <div>
            <h1>🍽️ GENFITY</h1>
            <p>Platform Pemesanan Online untuk Restoran</p>
        </div>

        <div>
            <h2>Selamat! Akun Anda Telah Dibuat</h2>
            <p>Halo <strong>[NAMA_MERCHANT/STAFF]</strong>,</p>
            <p>Kami dengan senang hati memberitahu bahwa akun Anda di platform GENFITY te

            <div>
                <p>📧 Email:</p>
                <p>[EMAIL]</p>

                <p>🔑 Password Sementara:</p>
                <p>[TEMP_PASSWORD]</p>
            </div>

            <p><a href="https://genfity.com/dashboard">Masuk ke Dashboard</a></p>

            <div>
                <strong>⚠️ PENTING:</strong><br>
                ✓ Ubah password Anda saat login pertama kali!<br>
                ✓ Jangan bagikan password ini kepada siapapun<br>
                ✓ Simpan email dan password ini di tempat yang aman
            </div>

            <h3>Langkah Selanjutnya:</h3>
            <ol>
                <li>Buka <a href="https://genfity.com/dashboard">https://genfity.com/dash
                <li>Login dengan email dan password sementara di atas</li>
                <li>Ubah password Anda ke password yang lebih kuat</li>
                <li>Mulai setup menu dan konfigurasi merchant Anda</li>
            </ol>

            <p>Jika Anda memiliki pertanyaan atau mengalami masalah login, silakan hubung
            <p>📧 <strong>support@genfity.com</strong></p>
            <p>📱 <strong>+61-XXX-XXX-XXXX</strong></p>
        </div>

        <div>
            <p>&copy; 2024 GENFITY. All rights reserved.</p>
            <p>Powered by GENFITY | <a href="https://genfity.com/privacy">Privacy Policy<
        </div>
    </div>
```

```
&lt;/body&gt;
&lt;/html&gt;
```

6.4 Backend Implementation Pseudocode

```
// POST /api/admin/merchants (create new merchant)
async function createMerchant(req: Request, res: Response) {
  const { name, email, merchantCode, phone } = req.body;

  try {
    // 1. Generate temp password
    const tempPassword = generateRandomPassword(16);

    // 2. Hash password
    const passwordHash = await bcrypt.hash(tempPassword, 10);

    // 3. Create user (transaction start)
    const user = await db.users.create({
      name,
      email,
      phone,
      password_hash: passwordHash,
      role: 'MERCHANT_OWNER',
      is_active: true,
      must_change_password: true  // wajib ubah password saat login pertama
    });

    // 4. Create merchant
    const merchant = await db.merchants.create({
      name,
      code: merchantCode,
      created_by_user_id: req.user.id  // super admin
    });

    // 5. Link user to merchant
    await db.merchant_users.create({
      merchant_id: merchant.id,
      user_id: user.id,
      role: 'OWNER'
    });

    // 6. Send email dengan temp password
    await sendEmailPasswordNotification({
      to: email,
      name,
      email,
      tempPassword,
      dashboardUrl: 'https://genfity.com/dashboard',
      supportEmail: 'support@genfity.com'
    });

    res.json({
      success: true,
      message: 'Merchant created successfully. Email dengan password dikirim.',
      merchant: {
```

```
        id: merchant.id,
        code: merchant.code,
        email
      }
    });

  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
}

// Helper function: Generate random password
function generateRandomPassword(length: number = 16): string {
  const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&amp
  let password = '';
  for (let i = 0; i &lt; length; i++) {
    password += chars.charAt(Math.floor(Math.random() * chars.length));
  }
  return password;
}

// Helper function: Send email
async function sendEmailPasswordNotification(params: any) {
  const htmlContent = renderEmailTemplate({
    name: params.name,
    email: params.email,
    tempPassword: params.tempPassword,
    dashboardUrl: params.dashboardUrl,
    supportEmail: params.supportEmail
  });

  await emailService.send({
    to: params.to,
    subject: 'GENFITY - Akun Merchant Baru Anda',
    html: htmlContent
  });
}
```

7. API Endpoints {#api-endpoints}

Auth Endpoints

- `POST /api/auth/login`: Login dengan email & password
- `POST /api/auth/logout`: Logout (revoke session)
- `POST /api/auth/refresh`: Refresh access token (jika pakai refresh token)
- `GET /api/auth/me`: Get current user profile

Admin Endpoints

- `POST /api/admin/merchants`: Create merchant (send password email)
- `GET /api/admin/merchants`: List all merchants
- `PUT /api/admin/merchants/:id`: Update merchant
- `DELETE /api/admin/merchants/:id`: Delete merchant

Merchant Endpoints

- `GET /api/merchant/profile`: Get merchant profile
- `PUT /api/merchant/profile`: Update merchant profile
- `POST /api/merchant/menu-categories`: Create menu category
- `GET /api/merchant/menus`: List menus
- `POST /api/merchant/menus`: Create menu
- `PUT /api/merchant/menus/:id`: Update menu
- `DELETE /api/merchant/menus/:id`: Delete menu
- `GET /api/merchant/orders`: List orders
- `PUT /api/merchant/orders/:id/status`: Update order status
- `GET /api/merchant/revenue`: Revenue report

Public Customer Endpoints

- `GET /api/public/merchants/:code`: Get merchant info
- `GET /api/public/merchants/:code/menu`: Get merchant menu
- `POST /api/public/orders`: Create order
- `GET /api/public/orders/:orderNumber`: Get order details

8. Struktur Folder Backend {#struktur-folder}

```
genfity-backend/
├── src/
│   ├── app/
│   │   ├── api/
│   │   │   ├── auth/
│   │   │   │   ├── login/
│   │   │   │   │   └── route.ts
│   │   │   │   ├── logout/
│   │   │   │   │   └── route.ts
│   │   │   │   ├── refresh/
│   │   │   │   │   └── route.ts
│   │   │   │   └── me/
│   │   │   │       └── route.ts
│   │   │   ├── admin/
│   │   │   │   ├── merchants/
```

```
│   │   │   │   │   ├── route.ts
│   │   │   │   │   └── [id]/
│   │   │   │   │       └── route.ts
│   │   │   │   └── stats/
│   │   │   │       └── route.ts
│   │   │   ├── merchant/
│   │   │   │   ├── profile/
│   │   │   │   │   └── route.ts
│   │   │   │   ├── menu-categories/
│   │   │   │   │   └── route.ts
│   │   │   │   ├── menus/
│   │   │   │   │   ├── route.ts
│   │   │   │   │   └── [id]/
│   │   │   │   │       └── route.ts
│   │   │   │   ├── orders/
│   │   │   │   │   ├── route.ts
│   │   │   │   │   └── [id]/
│   │   │   │   │       ├── route.ts
│   │   │   │   │       └── status/
│   │   │   │   │           └── route.ts
│   │   │   │   └── revenue/
│   │   │   │       └── route.ts
│   │   │   └── public/
│   │   │       ├── merchants/
│   │   │       │   ├── [code]/
│   │   │       │   │   ├── route.ts
│   │   │       │   │   └── menu/
│   │   │       │   │       └── route.ts
│   │   │       │   └── route.ts
│   │   │       ├── orders/
│   │   │       │   ├── route.ts
│   │   │       │   └── [orderNumber]/
│   │   │       │       └── route.ts
│   │   │       └── health/
│   │   │           └── route.ts
│   │   └── layout.tsx
│   ├── lib/
│   │   ├── db/
│   │   │   ├── client.ts          // Database connection
│   │   │   ├── schema.ts          // Type definitions
│   │   │   └── migrations/
│   │   ├── services/
│   │   │   ├── AuthService.ts
│   │   │   ├── MerchantService.ts
│   │   │   ├── MenuService.ts
│   │   │   ├── OrderService.ts
│   │   │   ├── PaymentService.ts
│   │   │   └── EmailService.ts
│   │   ├── repositories/
│   │   │   ├── UserRepository.ts
│   │   │   ├── MerchantRepository.ts
│   │   │   ├── MenuRepository.ts
│   │   │   ├── OrderRepository.ts
│   │   │   └── SessionRepository.ts
│   │   ├── middleware/
│   │   │   ├── auth.ts            // JWT verification
```

```
│   │   │   ├── errorHandler.ts
│   │   │   ├── rateLimiter.ts
│   │   │   └── cors.ts
│   │   ├── utils/
│   │   │   ├── passwordHasher.ts
│   │   │   ├── jwtManager.ts
│   │   │   ├── qrCodeGenerator.ts
│   │   │   ├── validators.ts
│   │   │   └── emailTemplates.ts
│   │   ├── types/
│   │   │   ├── user.ts
│   │   │   ├── merchant.ts
│   │   │   ├── menu.ts
│   │   │   ├── order.ts
│   │   │   ├── auth.ts
│   │   │   └── api.ts
│   │   └── constants/
│   │       ├── roles.ts
│   │       ├── status.ts
│   │       └── errors.ts
│   ├── config/
│   │   ├── database.ts
│   │   ├── email.ts
│   │   └── jwt.ts
│   └── public/
│       ├── emails/
│       │   └── password-notification.html
│       └── images/
│           └── logo.png
├── prisma/
│   ├── schema.prisma              // Prisma schema (alternative to raw SQL)
│   └── migrations/
├── .env.example
├── .env.local                     // (git-ignored)
├── package.json
├── tsconfig.json
├── next.config.js
└── README.md
```

Kesimpulan

Dokumentasi ini menyajikan blueprint lengkap untuk sistem GENFITY Online Ordering dengan:

✅ **Database Design**: 13 tabel terstruktur dengan relasi normalisasi & indexing
✅ **Authentication**: JWT-based dengan user_sessions tracking untuk security & audit
✅ **Email Integration**: Otomatis pengiriman password temporary ke merchant/staff baru
✅ **Business Logic**: Flow yang jelas dari customer order hingga merchant reporting
✅ **API Architecture**: RESTful endpoints dengan role-based access control
✅ **Production Ready**: Scalable, secure, dan easy to maintain

Sistem ini siap untuk development fase berikutnya dengan implementasi di Next.js + PostgreSQL.