Digital eMenu : Order Menu Module Requirements

1. Introduction

   The Order Menu module is the transactional foundation of the Digital eMenu platform. It enables customers to place food and beverage orders digitally by selecting items, customizing them, and completing payments through well-defined ordering and payment flows.

   The module supports restaurants operating with or without a POS system and accommodates multiple business models, including prepaid, post-paid, partial-paid, and hybrid ordering flows. It ensures that pricing, taxation, discounts, and payments are handled accurately while maintaining a seamless customer experience.

   By connecting the guest-facing ordering journey with restaurant billing rules, payment processes, and settlement requirements, the Order Menu module plays a central role in enabling reliable and scalable digital ordering for restaurants.

2. Module Purpose & Objectives

   2.1. Purpose
   The Order Menu module allows restaurants to:
   - Accept digital orders for multiple order types.
   - Support prepaid and post-paid ordering models
   - Apply taxes, discounts, and coupon codes
   - Support single or split payments for a single bill
   - Operate with or without a POS system
   - Maintain accurate and auditable order records
   - Integrate seamlessly with POS and payment gateways
   - Manage order lifecycle from creation to settlement

   2.2. Objectives
   - Support a wide range of restaurant ordering scenarios without creating a mandatory dependency on POS systems
   - Enable flexible payment experiences, including single and split payments, across supported ordering models
   - Ensure accurate and transparent billing through correct application of pricing, taxes, discounts, and rounding rules
   - Provide a consistent and intuitive ordering experience for guests across all touchpoints
   - Accommodate different restaurant billing and settlement practices with clear order and payment states
   - Minimize operational ambiguity during order creation, updates, and settlement
   - Enable a scalable ordering foundation that can be reused across dine-in, takeaway, and delivery journeys

3. Scope Overview
   - Order creation & cart management
   - Pricing computation
   - Tax calculation
   - Discount & coupon application
   - Prepaid & Post-Paid logic
   - Split payments
   - Partial payments
   - Order updates (post-paid)
   - POS sync (optional)
   - Payment gateway integration
   - Order state management

- Error handling & recovery
- Edge case handling

3.1. Supported Order Types

- **Prepaid Order**
  - Full payment before order is placed
  - Common for QR ordering, takeaway, delivery
- **Post-Paid Order**
  - Order is created without payment
  - Bill is settled later (dine-in)

3.2. Supported Workflows

- **Restaurants Using a POS System**
  - The restaurant uses a POS system for billing and settlement
  - Prices, taxes, and final bill values are governed by the POS
  - Digital eMenu facilitates order placement and digital payment
- **Restaurants Without a POS System**
  - The restaurant does not use any external POS
  - Digital eMenu manages order creation, pricing, tax calculation, and payment settlement

4. Functional Requirements

4.1. Order Creation & Cart Management

- Customers can initiate an order via QR code or digital menu link.
- Orders must be associated with branch, order type.
- Customers can select items, choose variations, add-ons, and specify quantities.
- Mandatory selections must be completed.
- Orders cannot proceed with invalid configurations.

4.2. Pricing & Bill Calculation

- The system calculates item prices, modifier prices, quantity-based totals, and subtotal.
- Pricing sources:
  - POS-enabled restaurants: price and tax aligned with POS system.
  - Non-POS restaurants: prices defined in Digital eMenu.

4.3. Tax Management

- Support for inclusive and exclusive taxes, multiple tax rates.
- Taxes can apply at item, category, or order level.
- Taxes are recalculated whenever items, discounts, or order updates occur.

4.4. Discounts & Coupon Handling

- Support percentage and flat discounts at item, category, or order level, including coupon codes.
- Discounts validated for eligibility, expiry, usage limits, and order rules.
- Total recalculated after discount application.

4.5. Payment Models

4.5.1. Prepaid Orders

- Payment collected before order confirmation.
- Order created only after successful payment.

Flow

- Customer reviews bill
- Full payment is completed
- Order is confirmed
- Order cannot be modified after payment (unless restaurant allows cancellations)

4.5.2. Post-Paid Orders

- Order created without payment; remains open.
- Updates allowed per restaurant configuration (add/remove items, update quantities, apply discounts).
- Full bill settled later; order closed only after payment.

Flow

- Order Creation
  - Customer places an order without payment
  - Order remains open
- Order Updates
  The following may be allowed based on restaurant configuration:
  - Add items
  - Increase / Decrease quantities
  - Remove items (delete the entire item from the order despite the quantity)
  - *Apply discounts*
- Bill Settlement
  - Customer completes one full payment for the order
  - Order is closed only after full settlement
  - Tips of percentage or custom amount has to be also an option within the payment page.

4.6. Split Payments

- Support equal, custom, and item-based splits.
- Total payments must cover the full bill.
- Order closed only when the total amount is fully paid.
- Over-payment not allowed.

4.7. Order Updates – Edit Order (Post-Paid Only)

- Open orders can be updated: add/remove items, update quantities, apply discounts (as permitted).
- Automatic recalculation of totals and taxes after updates.

4.8. POS-Enabled Restaurant Scope

- Align prices, taxes, and final bill with POS.
- Support digital order placement and payment against POS-managed bills.
- Prepaid and post-paid workflows supported per restaurant configuration.

4.9. Non-POS Restaurant Scope

- Full order management within Digital eMenu.
- Internal handling of pricing, taxes, and bill settlement.
- Support prepaid and post-paid workflows without external dependencies.

4.10. Order Status Management

- The system must maintain the following order statuses:
  - Draft
  - Open

    - Paid
    - Cancelled
    - Refunded
- Statuses must always reflect the actual settlement state of the order.

4.11. Error & Exception Handling

- The system must handle:
    - Payment failures
    - Discount validation issues
    - Order update conflicts
    - Temporary service unavailability
- Clear, user-friendly messages must be shown in all cases.

5. Acceptance Criteria

1. Orders can be placed with or without a POS system.
2. Both prepaid and post-paid flows are supported.
3. Full bill settlement is enforced.
4. Split payments cover the total bill and close orders only after full payment.
5. Discounts and taxes applied accurately at all levels.
6. Order statuses correctly reflect real-time settlement and modifications.
7. The system prevents duplicate or invalid orders.
8. User-facing error messages are clear and actionable in all edge cases.

6. User Stories

6.1 Order Creation

- As a customer, I want to scan a QR code or open a digital menu link so that I can start an order.
- As a system, I must associate the order with branch, order type, and table so that the restaurant can process it accurately.

6.2 Item Selection & Validation

- As a customer, I want to select items, variations, add-ons, and quantities so that my order is customized.
- As a system, I must enforce mandatory selections and availability rules so that invalid orders are prevented.

6.3 Pricing & Billing

- As a customer, I want to see item prices, modifiers, and subtotal updated in real-time so that I know my total cost.
- As a system, I must calculate prices using POS or eMenu rules depending on the restaurant type.

6.4 Tax & Discounts

- As a customer, I want taxes and discounts applied accurately so that the final bill is correct.
- As a system, I must support inclusive/exclusive taxes, multiple tax rates, and discount validation rules.

6.5 Payment

- As a customer, I want to pay upfront (prepaid) or later (post-paid) depending on my preference and restaurant rules.
- As a system, I must only confirm prepaid orders after successful payment and allow post-paid orders to remain open until settled.

6.6 Split Payments

- As a customer, I want to split the bill equally, by item, or by custom amounts so that multiple payers can settle one order.
- As a system, I must ensure total payment equals the full bill before closing the order.

6.7 Order Updates (Post-Paid)

- As a customer, I want to update my post-paid order (add/remove items, apply discounts) so that my order reflects my preferences.
- As a system, I must recalculate totals and taxes automatically after each update.

6.8 Error Handling

- As a customer, I want clear messages when payment fails, discounts are invalid, or updates conflict so that I understand what went wrong.
- As a system, I must prevent duplicate or invalid orders and handle temporary failures gracefully.

6.9 Order Status

- As a customer, I want to see the current status of my order (Draft, Open, Paid, Cancelled, Refunded) so that I know if it's been processed.
- As a system, I must update order status accurately based on payment and updates.

Additional Requirements: Dashboard Order, Table & Payment Visibility - Refer to the similar behavior in order status from emenu legacy

1. Order List (Tenant / Customer Dashboard)

- The dashboard must display a **real-time list of orders**.
- Orders must be filterable by:
  - Restaurant branch
  - Table number
- Each order in the list must show:
  - Order ID
  - Branch
  - Table number (if dine-in)
  - Order type (Prepaid / Post-Paid)
  - Order status (Draft, Open, Paid, Cancelled, Refunded)
  - Payment state (Unpaid, Partially Paid, Fully Paid, Failed, Refunded)
  - Total amount
  - Paid amount
  - Pending amount

---

2. Table State Visibility

- The dashboard must display **table-level status** for dine-in orders.
- Table states:
  - Vacant
  - Occupied – Unpaid
  - Occupied – Partially Paid
  - Occupied – Fully Paid
- Table state must be derived from the payment and order state of associated orders.

---

3. Payment & Split Payment Visibility

- Orders must clearly indicate:
  - Single, partial, or split payments
  - Number of payments made
  - Remaining payable amount
- Orders are marked **Fully Paid** only when the total bill amount is completely settled.

---

4. Order Detail View (Optional Drill-down)

- Selecting an order must show:
  - Item summary
  - Tax and discount breakdown
  - Payment transaction details

---

5. Real-Time Updates

- Order list, table state, and payment state must update in near real-time without manual refresh.