Subscription System — Competition-Grade Blueprint

0) Purpose (one line)

Create a fast, reliable, rule-driven system for configuring monthly service subscriptions per unit and resident, in less than the time it takes to fill the current two-page paper form — while producing structured data ready for approval, billing, and analytics.

———

1) Core Principles (non-negotiables)

1. One Unit per Session: A subscription session always belongs to exactly one unit and month.
2. Clean Context: Changing unit or month hard-resets all in-progress selections.
3. Catalog Normalization: Services are normalized once (filter/trim/sort/derive) before the UI sees them.
4. Scope Truth: UNIT services only appear and apply to Customer #1; never copied to others.
5. Rule Engine, not scattered if/else: Exclusivity and visibility are data-driven (config).
6. Two-click speed: “Repeat last month” and “Repeat previous customer” are first-class actions.
7. Auditability: Every save has immutable master + line items, user + timestamp.

———

2) Information Model (DB / API-agnostic)

Entities

- Unit: unit_id, block, status
- Customer: customer_id, full_name, unit_id, active_status
- ServiceCategory: category_id, name, display_name, sort_order
- Service:
service_id (e.g., F&B206), name, category_id, scope (INDIVIDUAL|UNIT), subscription_type (only_subscription | subscription_and_open | open_only), code_number (last 3 digits), default_location, proof_required_default
- SubscriptionMaster:
subscription_id, unit_id, period_start, period_end, status (DRAFT→SUBMITTED→APPROVED→ACTIVE), created_by, created_at
- SubscriptionCustomer (link table): subscription_id, customer_id
- SubscriptionLine (atomic):
subscription_id, customer_id (nullable for UNIT scope), service_id, start_date, end_date, specific_dates (array), time, location, proof_required, special_instructions

Note: This mirrors what you already have, just formalized.

———

3) Business Rules

3.1 Category display order (global)

1. Food & Beverages
2. CARE (Housekeeping)
3. CARE (Laundry)
4. CARE (Health & Wellbeing)
5. Nursing & Medical
6. Accommodation Services
7. Concierge On Request Services
8. Resident Admin & Billing
9. Estate, Legal, Death & Rituals

3.2 Service sorting (within a category)

- Sort by code_number (numeric from last 3 digits of service_id), ascending.
- Any service without a valid code_number is excluded and logged for cleanup.

3.3 Eligibility filter

- Only show services where subscription_type ∈ { only_subscription, subscription_and_open }.

3.4 Scope visibility

- UNIT services visible/selectable only for Customer #1; never offered to others.
- When copying (“Repeat previous”), UNIT services are never copied.

3.5 Exclusivity (declarative)

Food & Beverages per customer

- One per meal group (auto-untick conflicting ones; show inline note):
- Breakfast: { Standard, Modified, Custom }
- Lunch: { Standard, Modified, Custom }
- Dinner: { Standard, Modified, Custom }
- Add-ons (may stack): Beverages, Snacks, Special Food Prep, Event/Festival meals.

CARE (Housekeeping) per customer

- One standard HK variant (mutually exclusive): { Std HK without materials, Std HK with materials }
- Add-on tasks may stack: Deep Cleaning variants, Cobweb removal, Post-event cleaning, etc.

Exclusivity is configured via a small map of exclusive groups keyed by category.

3.6 Dates & occurrences

- By default, start_date=first_of_month, end_date=last_of_month.
- Optional specific_dates overrides range (comma list → array of valid day numbers).
- Time and Location are optional; default locations pulled from service master when present.

3.7 Session integrity

- If unit_id changes while in progress → hard reset (customers, selections, preview).
- On save, reject if any selected customer’s real unit ≠ session unit.

3.8 State machine

- DRAFT (user can edit) → SUBMITTED (locked for user, visible for supervisor) → APPROVED (write-protected) → ACTIVE (operational).

———

4) UX Model — Paper-speed “One-Page Canvas”

Layout

- Top Bar: Month selector; Unit selector; actions: Repeat Last Month, Save Draft, Submit.
- Left Pane: Customers in the selected unit (checkboxes; max 4 if that’s a business cap).
- Middle Pane: Category cards in the order above. Each card lists all eligible services, sorted by code.
- Tick = select; untick = remove.
- When ticked, reveal inline config (date range or specific days, time, location, proof, notes).
- Tiny (Unit) or (Individual) tag after service name.
- Long lists get “Show more (N)” within the card (progressive reveal).

- Right Pane (optional): “Selected summary” for the currently focused customer.

Fast actions

- Repeat previous customer (appears after saving Customer #1): copies only Individual services to next customer.
- Repeat last month: bulk prefill from prior approved subscription for the same unit (with current month dates).

Validation UX

- Inline conflict hint for exclusives: “Only one Breakfast service can be selected.”
- Disabled “Submit” with reasons listed (e.g., “No customers selected”, “No lines configured”).

———

5) API Surface (FastAPI)

Read

- GET /units → { units: [...] }
- GET /customers/{unit_id} → { customers: [...] }
- GET /services →

```
{
  "services": {
    "Food & Beverages": [
      {
        "id": "F&B206",
        "name": "Breakfast Custom Menu",
        "scope": "INDIVIDUAL",
        "subscription_type": "subscription_and_open",
        "default_delivery_location": "Dining Hall",
        "proof_required": false
      }
    ],
    ...
  }
}
```

Write

- POST /subscriptions

Payload:

```
{
  "master": {
    "unit_id": "MMSVA01",
    "period_start": "2025-11-01",
    "period_end": "2025-11-30",
    "status": "DRAFT"
  },
  "customers": [{ "customer_id": 41 }, { "customer_id": 42 }],
  "lines": [
    {
      "customer_id": 41,              // null for UNIT scope
      "service_id": "F&B206",
      "category": "Food & Beverages",
      "service_scope": "INDIVIDUAL",
      "start_date": "2025-11-01",
      "end_date": "2025-11-30",
      "specific_dates": [1,4,6],
```

```
      "time": "08:00",
      "location": "Dining Hall",
      "proof_required": false,
      "special_instructions": null
    }
  ]
}
```

- POST /subscriptions/{id}/submit → transitions to SUBMITTED
- POST /subscriptions/{id}/approve → to APPROVED

Backend enforces: one unit per subscription, code_number extraction, and rejects open-only services.

———

6) Rule Engine Config (frontend, data-driven)

```
// category order (already enforced)
export const CATEGORY_FLOW = [
  'Food & Beverages',
  'CARE (Housekeeping)',
  'CARE (Laundry)',
  'CARE (Health & Wellbeing)',
  'Nursing & Medical',
  'Accommodation Services',
  'Concierge On Request Services',
  'Resident Admin & Billing',
  'Estate, Legal, Death & Rituals'
]

// exclusivity groups by category (service_id suffixes or exact IDs)
export const EXCLUSIVE_GROUPS = {
  'Food & Beverages': [
    ['F&B201','F&B205','F&B206'], // Breakfast: Std, Modified, Custom
    ['F&B207','F&B211','F&B209'], // Lunch
    ['F&B210','F&B212','F&B208']  // Dinner
  ],
  'CARE (Housekeeping)': [
    ['Care-HK301','Care-HK302']  // Std HK variants
  ]
}
// category order (already enforced)
export const CATEGORY_FLOW = [
  'Food & Beverages',
  'CARE (Housekeeping)',
  'CARE (Laundry)',
  'CARE (Health & Wellbeing)',
  'Nursing & Medical',
  'Accommodation Services',
  'Concierge On Request Services',
  'Resident Admin & Billing',
  'Estate, Legal, Death & Rituals'
]

// exclusivity groups by category (service_id suffixes or exact IDs)
export const EXCLUSIVE_GROUPS = {
  'Food & Beverages': [
    ['F&B201','F&B205','F&B206'], // Breakfast: Std, Modified, Custom
    ['F&B207','F&B211','F&B209'], // Lunch
```

```
    ['F&B210','F&B212','F&B208']  // Dinner
  ],
  'CARE (Housekeeping)': [
    ['Care-HK301','Care-HK302']  // Std HK variants
  ]
}
```

Algorithm:

- When user checks a service X, look up the category's exclusive groups; if X is in a group, untick others from the same group for that customer immediately and show a brief hint.

———

7) Guards (single place, predictable)

- onMonthChange → reset session.
- onUnitChange → reset session.
- onCustomerConfirm → if none selected, block.
- beforeSubmit → verify single unit + at least one line → else block with reasons.

All resets clear: selectedCustomers, selectedCategories, chosenServices, previewRows, activeCustomerIndex, and reload services if needed.

———

8) Performance & Reliability

- Catalog normalized and cached in memory on first load; reused per session.
- Payload size small (month scope); no need for pagination.
- Idempotent save: client includes a client_request_id UUID to prevent double posts on retries.
- Server validates code_number (3 digits) and subscription_type eligibility.

———

9) Security & Roles

- User (create/edit own DRAFT, submit).
- Supervisor (review/approve).
- Admin (override, reporting).
- All writes authenticated; all approvals logged.

———

10) Rollout & Testing (acceptance scenarios)

A. Unit isolation

- Change unit mid-session → verify hard reset; no mixing customers across units; submit blocked if mismatch.

B. Scope

- On Customer 2+, no UNIT services appear; repeat doesn't copy UNIT services.

C. Exclusivity

- Select Breakfast Standard, then Breakfast Modified → Standard auto-unticks; hint appears.

D. Sorting & Display

- Food shows before Housekeeping; within Food, codes 201/205/206 sorted ascending.

E. Save/Submit

- Valid DRAFT saves; submit transitions state; approve locks records.

F. Repeat last month
- Prefill lines from the last approved subscription for that unit, with current month's dates.

———

11) Implementation Plan (tight phases)
1. Phase 1 (done): Normalize /services; filter + sort; category order.
2. Phase 2: Unit isolation + scope enforcement + guardrails (reset on change; submit guard; (Unit)/(Individual) tags).
3. Phase 3: Exclusivity engine with data config (F&B + HK).
4. Phase 4: Repeat previous customer & Repeat last month.
5. Phase 5: One-page canvas UX polish (dividers, show-more), and approval flow screens.
6. Phase 6: Final acceptance tests and small fixes.

Each phase is self-contained and testable in minutes.

———

12) What you'll get (deliverables)
- /reports/SSDM9_Subscription_Blueprint.md (this document)
- Updated frontend (Vue) with rule engine & canvas UI
- Backend fixtures for /services structure (if needed)
- Test checklist you can run yourself

———

If this matches your vision, say:
"OK—make the blueprint file"
I'll add it to /ssdm9/_reports/, then implement Phase 2 immediately after your confirmation, step by step.

Addenda to the Blueprint (closing the gaps)

A) Multiple customers per unit (up to 4)

Rule A1 — Cap & validation
- A subscription session belongs to one unit + one month.
- You may select 1–4 customers from that unit for the session.
- Frontend prevents selecting >4; backend re-validates.
- Each customer can subscribe to a different set of services.

Model note
- SubscriptionCustomer (subscription_id, customer_id) already supports many customers per subscription.
- On submit, backend verifies all customer_id belong to master.unit_id.

———

B) Per-service preferences per customer

Each selected service for a customer carries its own configuration:
- start_date / end_date or specific_dates (array of day numbers)
- time (HH:MM)
- location (dropdown + free text fallback if needed)
- special_instructions (free text)

Model

- These fields live on SubscriptionLine (one per customer × service).
- Example (per your scenarios):
- Customer A: F&B201 Breakfast Standard → location=Unit, time=08:00
- Customer A: F&B209 Lunch Modified → location=Dining Hall, special_instructions="Chapathis"
- Customer A: F&B210 Dinner Standard → location=Dining Hall, note="Except if it rains" (see Occurrence handling below)

UI

- When a service is ticked, its config panel opens under it; defaults are prefilled (1st/last day of month; default location if any).

———

C) Unique, trackable “service instances” for delivery

To make each delivery visit trackable, we expand configured lines into occurrences (one per day/selected date).

New table: ServiceOccurrence

- occurrence_id (ULID/UUID) — unique per visit
- subscription_id, line_id (FK → SubscriptionLine)
- date (YYYY-MM-DD), time (optional)
- customer_id (nullable for UNIT scope), unit_id (denormalized for fast queries)
- service_id, category, scope
- location
- status (SCHEDULED → IN_PROGRESS → COMPLETED | MISSED | CANCELLED)
- proof_type (enum; see D), proof_ref (URL/blob id), proof_meta (json), ack_by, ack_at
- notes (e.g., “Skipped due to rain”)

When to expand

- On SUBMIT (user → supervisor), backend expands all SubscriptionLine rows into ServiceOccurrence for the month.
- If the draft is edited and re-submitted, occurrences are re-generated (idempotent per line+date).
- Option (later): expand daily for rolling windows if calendar months are large.

Delivery app endpoints

- GET /occurrences?unit_id=&date_from=&date_to=&status= — list work
- PATCH /occurrences/{id} — update status, attach proof, notes
- (Optional) POST /occurrences/{id}/ack — record signature/ack

———

D) Proof of service (ack/photo/video/etc.)

Some services require proof; capture it on the occurrence.

Proof options (enum)

- NONE
- ACK_SIGNATURE (touch-signature or typed name)
- PHOTO
- VIDEO
- (Optional) QR_ACK (scan resident’s QR)

Where set

- Default per service (master data): proof_required_default, default_proof_type.
- Overridable per SubscriptionLine (user can require it).
- Stored per ServiceOccurrence when delivery is done:
- proof_type
- proof_ref (S3/Cloud bucket key)
- proof_meta (mime, bytes, device, GPS if available)
- ack_by, ack_at

Storage

- Files go to an object store (e.g., S3). DB stores only references/metadata.

———

E) Tying the pieces together (flow recap)

1. User configures per customer per service (preferences captured on SubscriptionLine).
2. Submit → backend validates single unit, caps (≤4), exclusivity, eligibility, and then expands to ServiceOccurrence for every target date.
3. Supervisor approves → occurrences become actionable; delivery team sees them in the task list.
4. Delivery marks each occurrence's status and attaches proof as required.
5. Reporting aggregates by occurrence.status, proof completion, SLA, etc.

———

F) Frontend adjustments to support this

- Service tick panel already captures preferences; add:
- proof_required checkbox and, if checked, a Proof Type select.
- "Except if it rains" and similar real-world exceptions:
- Keep as special_instructions text; on delivery day, staff can mark CANCELLED with reason (weather).
- (Future) Simple rules like "skip on weekly holiday" can be a small rule engine, but not needed for version 1.

———

G) Validation & guards (updated)

- Max customers: 1–4 (frontend & backend).
- Unit isolation: changing unit/month triggers hard reset; backend rejects mixed units.
- Service eligibility: only "only_subscription" / "subscription_and_open" allowed.
- Exclusivity: enforced per customer as defined (F&B meal groups; HK standard variants).
- Occurrence sanity: expansion only creates dates within month; dedupe by (line_id,date).
- Proof: if proof required, occurrence cannot be COMPLETED without proof_ref (enforced server-side).

———

H) API additions (succinct)

- POST /subscriptions — create DRAFT (master + customers + lines).
- POST /subscriptions/{id}/submit — validate + expand to occurrences (status → SUBMITTED).
- POST /subscriptions/{id}/approve — approve (status → APPROVED).
- GET /occurrences — filters: unit, date range, status, customer.
- PATCH /occurrences/{id} — update status, proof, notes (delivery).
- (Optional) GET /subscriptions/{id}/preview — server-side preview of expanded occurrences before submit.