Assignment 09

Last updated by | Hashim Javed | Aug 27, 2025 at 2:21 PM GMT+5

1. Campus Bookstore “Flash-Sale” Ordering

Scenario

The University Bookstore runs 15-minute flash sales for limited-stock items (hoodies, mugs). students order via mobile; payment is online; pickup is at the store within 24 hours.

Goals (what your system must achieve)

- handle traffic spikes during flash windows.
- ensure no overselling.
- allow payment retries and refunds for stockouts.
- notify customers about order status and pickup reminders.

Core Flows

- browse → add to cart → checkout
- pay → reserve stock → confirm order → send receipt
- cancel/refund (customer or stockout)
- pickup confirmation at counter

Constraints & Rules

- inventory is authoritative for on-hand units; reservations expire in **10 minutes** if payment not captured.
- payment provider can return late webhooks and duplicate callbacks.
- pickup must be confirmed on a store device (no customer self-confirm).
- flash sale window: **start/end timestamps**; orders outside window rejected.

Design Task

Propose **3–5 microservices** with clear boundaries and APIs; define the events they publish/consume and the data each one owns.
typical service candidates (use only what you need). Create architecture and event flow diagram.

- Catalog (read-heavy product data, price, flash windows)
- Order (order lifecycle, SAGA coordinator)
- Inventory (stock levels, reservations, releases)
- Payment (charge, refund, webhook intake, idempotency)
- Notification (email/sms/push)

Integration Moments to think about

- `OrderPlaced` → attempt `ReserveStock` → if success, request `AuthorizePayment` ; otherwise cancel.
- webhook from gateway → `PaymentCaptured` → finalize order.

- unpicked orders after 24h → auto cancel + `ReleaseStock` + refund.

2. Multi-Clinic Appointment Scheduling (Outpatient)

Scenario

A small healthcare network with 3 clinics offers general practice. patients book 15/30-minute slots with specific providers. appointments can require a pre-visit form and an optional co-pay. reminders go by sms.

Goals

- prevent double-booking per provider and room.
- enforce clinic hours, provider schedules, and blackout dates.
- handle cancellations, reschedules, and waitlists.
- send reminders (24h and 2h before).

Core Flows

- search availability (by clinic, provider, or service type)
- book appointment (+ optional co-pay authorization)
- complete pre-visit intake
- cancel/reschedule; auto-fill from waitlist
- visit completed → mark as attended/no-show → settle co-pay

Constraints & Rules

- each provider has a repeating weekly template with exceptions (vacation).
- co-pay is **authorized at booking, captured after visit** (or voided if canceled in time).
- a patient can't hold more than **one active appointment** with the same provider within 7 days.
- no-show fee applies if cancel < 2 hours before.

Design Task

Propose **3–5 microservices** and their contracts. define ownership of schedules, appointments, and patient contact info. Create -architecture and event flow diagram.

Integration Moments to think about

- hold slot for **5 minutes** during booking while co-pay auth runs.
- when provider calendar changes, **invalidate** affected availabilities and notify.
- post-visit, encounter outcome triggers Billing capture or void.

3. City Bike-Share: Passes & Checkouts

Scenario

A city runs a dock-based bike-share. users buy passes (single ride, day, monthly). they unlock bikes at docks via app or QR. fees depend on ride duration and pass type. overage charges apply.

Goals

- activate passes instantly.
- ensure only one active bike per user at a time.
- compute ride cost (including free minutes, overages).
- handle lost bike workflows (unreturned > 24h).
- near-real-time dock availability for the app.

Core Flows

- buy/activate pass
- at dock: unlock bike → start trip
- end trip: return to dock → compute fare → charge
- anomalies: ride exceeds 24h, bike returned to maintenance dock, charge dispute

Constraints & Rules

- passes:
  - **single ride**: 30 minutes included; overage per 15 minutes.
  - **day pass**: unlimited rides, each up to 45 minutes free.
  - **monthly**: same as day pass; auto-renews.
- a user may have only **one active trip** at any time.
- some docks are **maintenance-only**; returning there ends trip but flags bike for service.
- payment gateway sends **asynchronous** success/failure webhooks.

Design Task

Design **3–5 microservices** and their APIs/events. decide who owns pass entitlements, trip state, and pricing rules. Create architecture and event flow diagram.

Integration Moments to think about

- Passes publishes `PassActivated` ; Trips checks entitlement on unlock.
- Trip end → Pricing calculates cost → Billing charges → receipt notification.
- unreturned bike after 24h → Trips emits `TripOverdue` → Fleet marks bike missing → Billing places hold/fee.