Sequence Diagrams — Full Documentation

This document contains five distinct sequence diagram scenarios derived from the Manage Courses / Registration system. Each scenario includes actors, a step-by-step sequence of messages (arrows), notes on alt/loop fragments, and guidance for drawing recursion where required. Use this as the authoritative documentation for diagrams to submit.

Add Course — Success

Actors: Student, System, Registrar, Finance

Sequence (in order):

- Student -> System: Login
- System -> Student: Display available courses
- loop [until registered]
- Student -> System: Select course & Register
- System -> Registrar: Check seats
- Registrar -> System: Seats OK
- alt [seats OK]
- System -> Finance: Verify payment status
- Finance -> System: Payment OK
- System -> Student: Registration complete — confirmation shown
- alt [seats full] Display seats are full

Notes: Happy path: seats available and payment succeeds. Use a loop only if you allow retries (not necessary here).

Drop Course — Minimum Rule

Actors: Student, System, Registrar

Sequence (in order):

- Student -> System: Open My Schedule / Select course to drop
- System -> Registrar: Check current course count if dropped
- Registrar -> System: Would drop below MIN (4)?
- alt [allowed]
- Registrar -> System: Dropping allowed
- System -> Student: Course dropped — schedule updated
- alt [would drop below MIN]
- Registrar -> System: Would drop below 4
- System -> Student: Cannot drop — minimum 4 courses required; show approval option

Notes: Shows allowed vs blocked drop. If approval flow exists, include 'Request approval to drop' arrows.

Add Course — Prerequisite Missing

Actors: Student, System, Registrar

Sequence (in order):

- Student -> System: Select course & Register
- System -> Registrar: Check prerequisites
- Registrar -> System: Prerequisite NOT MET
- alt [prerequisite missing] block registration
- System -> Student: Registration blocked — 'Prerequisite X required'
- Optional: student -> system: Request instructor override

Notes: Prerequisite check must occur BEFORE seat check. Keep this flow short and direct.

Pay Tuition Fees — WITH Recursion

Actors: Student, System, Finance, Bank

Sequence (in order):

- Student -> System: log in
- Student -> System: open billing
- System -> Finance: getBalance
- Finance -> System: balance
- System -> Student: showBalance
- System -> System: validatePayment() <-- self-call, double activation bar
- loop [attempt < 3]
- System -> System: retryPayment() <-- recursive retry inside loop
- Student -> System: confirmPayment
- System -> Bank: processPayment
- Bank -> System: paymentStatus
- System -> Finance: updateBalance
- System -> Student: result
- alt [success] System -> Student: payment complete
- alt [failed] System -> Student: payment failed

Notes: Recursion is represented as self-calls (System -> System). Use a loop with attempt counter for retry limit. Place double activation bar on System lifeline for validatePayment().

Registration Blocked by Finance Hold

Actors: Student, System, Finance

Sequence (in order):

- Student -> System: Select course & Register
- System -> Finance: Check account status

- Finance -> System: Outstanding fees / Hold
- System -> Student: Account on hold — clear fees first
- Optional: System -> Student: Show balance + payment link

Notes: Finance hold prevents registration; show clear instructions to student.
