ENROLLMENT SYSTEM
-- ENR1 --

Timotej Kotlín & Sara Alić-Ekinović
Group: - EXA1 -

MODIFICATIONS TO C4 MODEL & QUALITY SCENARIOS

EnrollmentSystem - Architecture Extensions & Justifications

This document describes **quality requirement scenarios** and the corresponding **architectural reasoning** based on the provided **C4 architecture** of the Enrollment System.

Selected Quality Dimensions

1. **Modifiability (Design-time)**
2. **Performance (Run-time)**
3. **Reliability (Run-time)**

SCENARIO 0 - MODIFIABILITY (DESIGN-TIME)

Frontend Replacement Without Backend Changes

Quality Dimension

Modifiability

Scenario Description

Element	Description
Stimulus	A UI team replaces **100% of the Enrollment frontend** by implementing a new client (new UI framework + new interaction design) that still supports the same **4 user flows**: (1) browse tickets, (2) enroll, (3) cancel enrollment, (4) join/leave waiting list.
Source	Product owner / frontend development team.
Environment	Design-time; backend containers are in **feature freeze** (no functional changes allowed); backend API specifications are published; automated API contract tests exist; staging environment available.
Artifact	**Presenter-side artifacts only**: `Enrollment Presenter` container and its UI components/controllers (notably `courseTicketController` and `waitingListController`).
Response	New frontend integrates using the existing backend API contracts; **no backend container requires modification** (`Enrollment Manager`, `Conditions Manager`, `Notification Service`, and their databases).
Response Measure	**0 backend production code changes**; **0 backend DB schema changes**; **100% backend API contract tests pass**; end-to-end acceptance suite for the 4 enrollment flows passes in staging with **≥99% success rate**.

SCENARIO 0 - MODIFIABILITY (DESIGN-TIME)

Existing Architecture Support

Strengths:

- The C4 model separates the UI into **Presenter containers** and the business logic into backend service containers.
- The frontend communicates with backend services via stable API boundaries, enabling independent frontend evolution.

Weaknesses:

- None identified for this scenario as long as API contracts remain stable.

Required Architectural Extensions

None.

What was added:

- **No new C4 elements**. (This scenario is satisfied by the existing separation of concerns.)

Reasoning

The C4 architecture already follows a **client-server / layered architectural style**.
Because the frontend is isolated in the `Enrollment Presenter` container and depends only on backend API contracts, it can be replaced without changing backend containers or persistence.

SCENARIO 3 - RELIABILITY (RUN-TIME)

Notification Service Outage During Enrollment

Quality Dimension

Reliability

Scenario Description

Element	Description
Stimulus	`Notification Service` becomes unavailable for **20 minutes** while enrollments and waiting-list auto-enrollments continue.
Source	Internal infrastructure failure.
Environment	Enrollment ongoing; `auto-EnrollWorker` processing active; Notification Service down; database and message bus available.
Artifact	Enrollment-to-notification delivery path: `Enrollment Manager` writers (including `enrollmentWriter` and `auto-EnrollWorker`), the `notificationOutbox`, the `Message Bus / Event Broker`, and the `Notification Service` container.
Response	Enrollment continues; notifications are delayed but never lost; delivery resumes after recovery.
Response Measure	**0 lost notifications**; **100% delivered** within **30 minutes** after service recovery; **0 enrollment failures** attributable to the outage.

SCENARIO 3 - RELIABILITY (RUN-TIME)

Existing Architecture Support

Weaknesses:

- Synchronous calls to Notification Service would block enrollment workflows during outages.
- Without durable buffering, notification events could be lost on failures.

Required Architectural Extensions

1. New Container: `Message Bus / Event Broker`

- `Enrollment Manager` publishes events (e.g., `EnrollmentCreated`, `AutoEnrolledFromWaitingList`).
- `Notification Service` consumes events asynchronously.

2. New Component: `notificationOutbox` (in Enrollment Manager)

- Durable storage of notification events (Outbox Pattern).
- Guarantees events are published even if the service crashes mid-flow.

What was added:

- `Message Bus / Event Broker` (container)
- `notificationOutbox` (component, Enrollment Manager)

SCENARIO 3 - RELIABILITY (RUN-TIME)

Existing Architecture Support

Weaknesses:

- Synchronous calls to Notification Service would block enrollment workflows during outages.
- Without durable buffering, notification events could be lost on failures.

Required Architectural Extensions

1. New Container: `Message Bus / Event Broker`

- `Enrollment Manager` publishes events (e.g., `EnrollmentCreated`, `AutoEnrolledFromWaitingList`).
- `Notification Service` consumes events asynchronously.

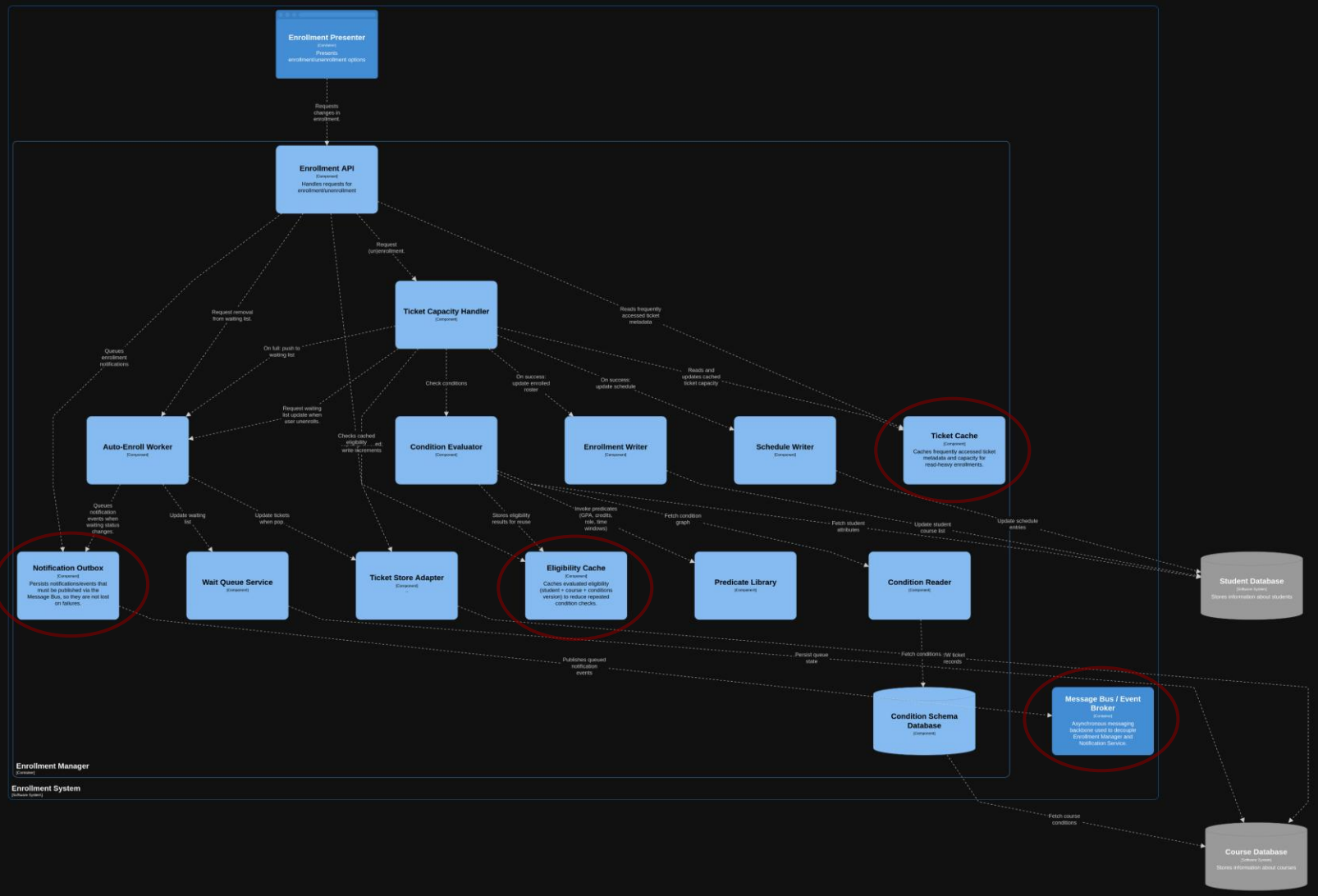
2. New Component: `notificationOutbox` (in Enrollment Manager)

- Durable storage of notification events (Outbox Pattern).
- Guarantees events are published even if the service crashes mid-flow.

What was added:

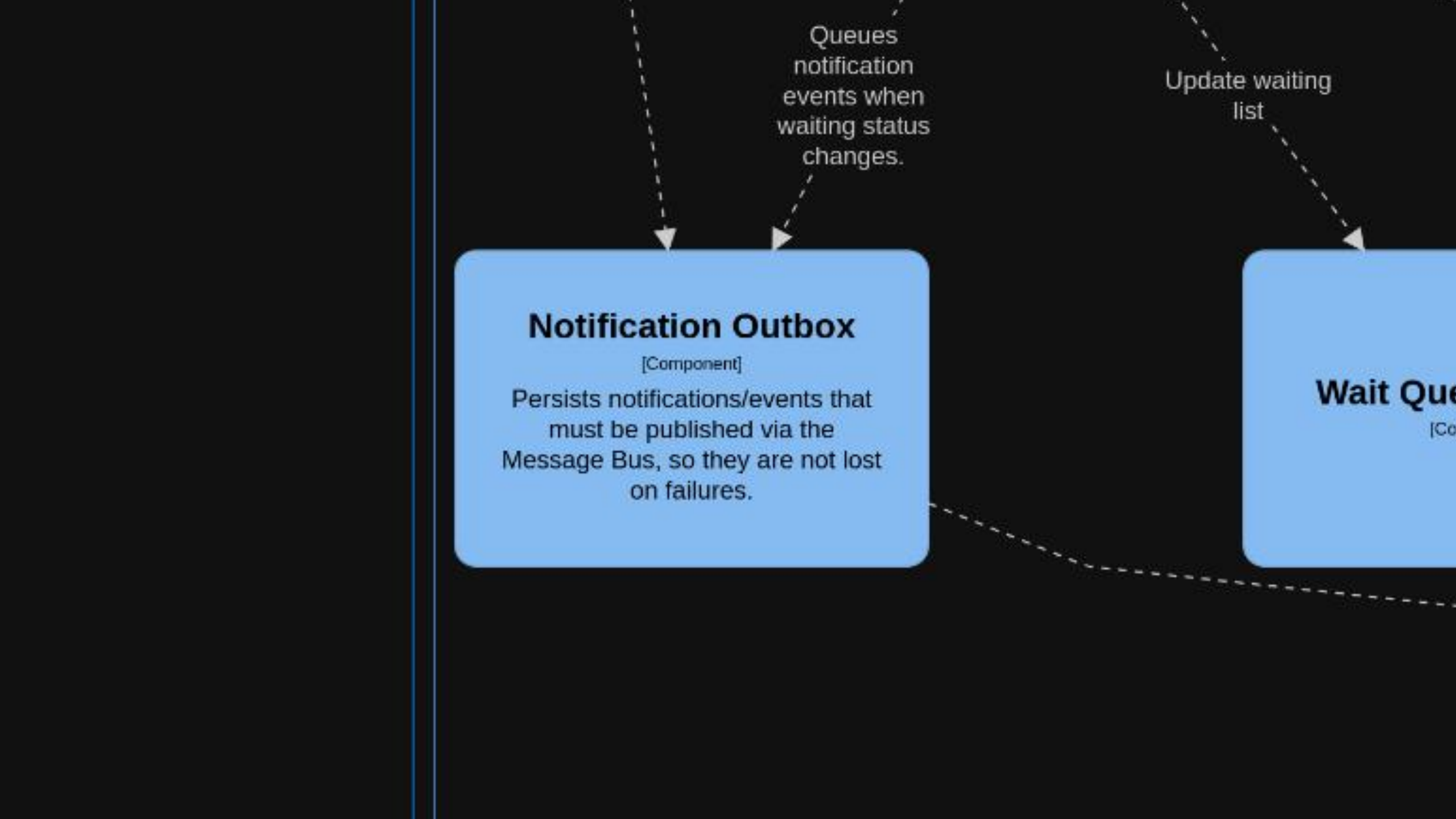
- `Message Bus / Event Broker` (container)
- `notificationOutbox` (component, Enrollment Manager)

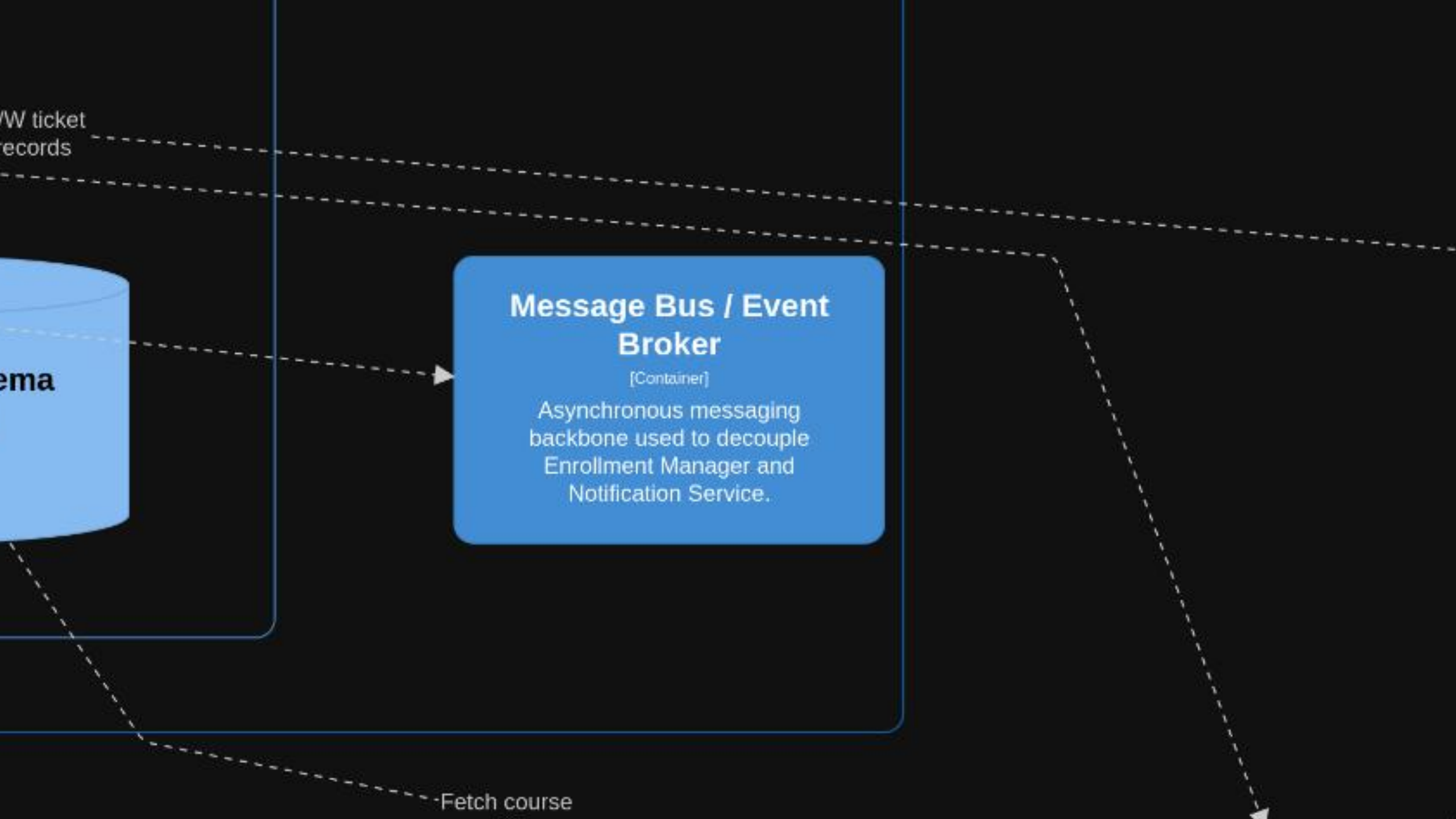
SCENARIO 3 - RELIABILITY (RUN-TIME)

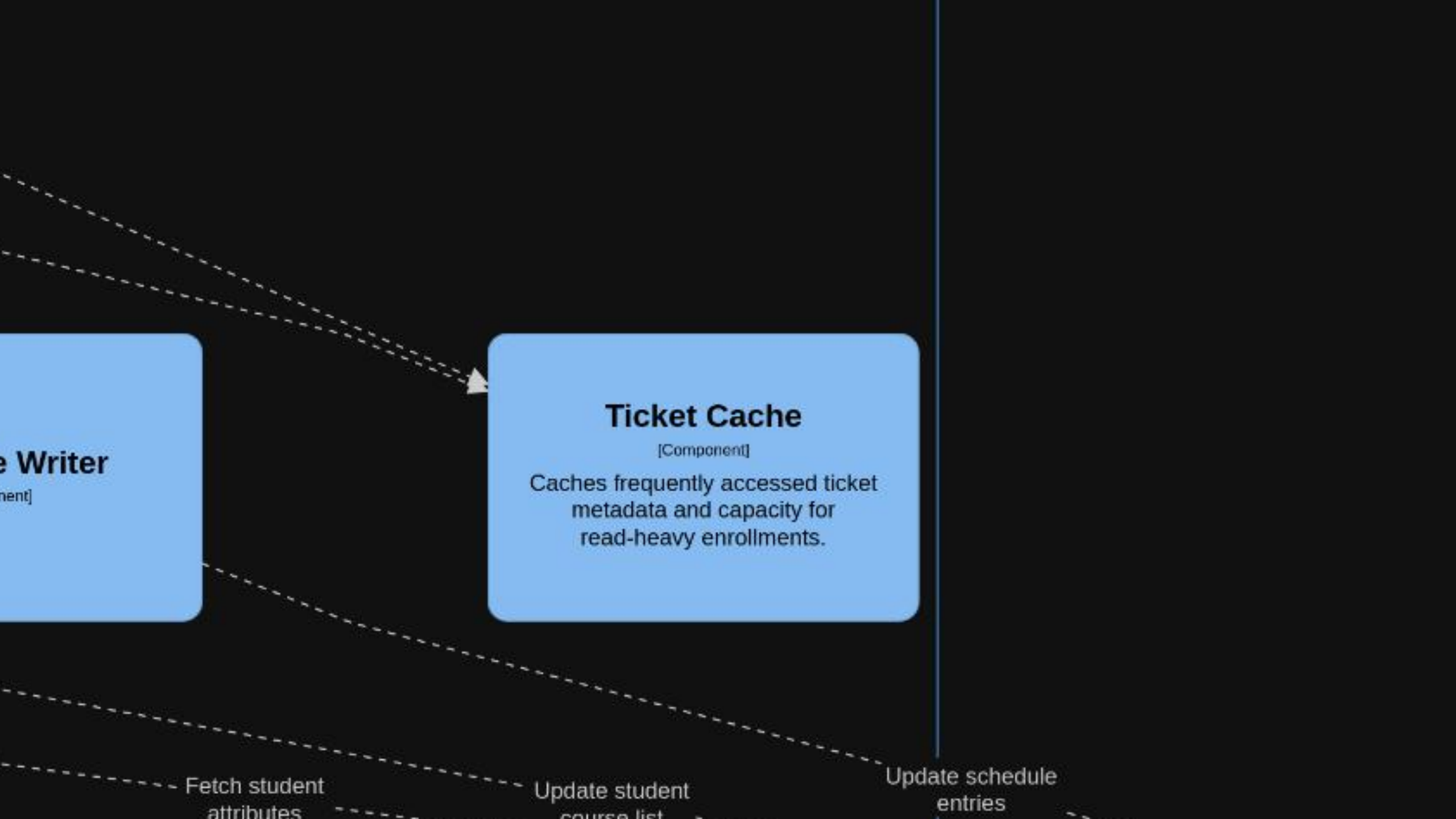


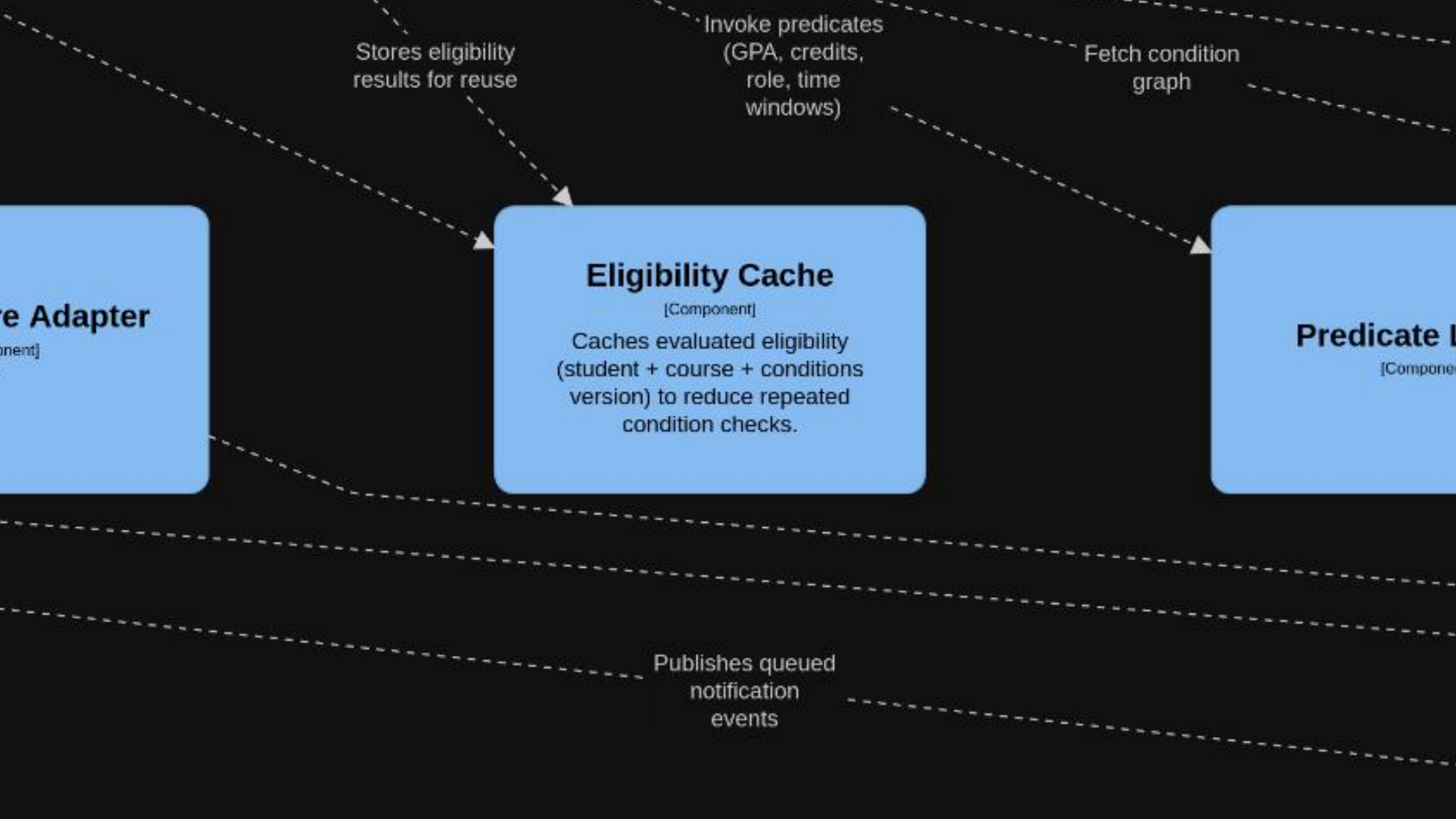
Component View: Enrollment System - Enrollment Manager

Monday, December 8, 2025 at 11:03 AM Central European Standard Time


Queues notification events when waiting status changes.
Update waiting list
Notification Outbox
[Component]
Persists notifications/events that must be published via the Message Bus, so they are not lost on failures.
Wait Que
[Co


W ticket
ecords
ema
Message Bus / Event Broker
[Container]
Asynchronous messaging backbone used to decouple Enrollment Manager and Notification Service.
Fetch course


e Writer
nent]
Ticket Cache
[Component]
Caches frequently accessed ticket metadata and capacity for read-heavy enrollments.
Fetch student attributes
Update student course list
Update schedule entries


Stores eligibility results for reuse
Invoke predicates (GPA, credits, role, time windows)
Fetch condition graph
e Adapter
onent]
Eligibility Cache
[Component]
Caches evaluated eligibility (student + course + conditions version) to reduce repeated condition checks.
Predicate
[Compone
Publishes queued notification events

THANK YOU FOR YOUR ATTENTION