Daily notes

Weekly Progress Report (Day 11-15)

This week, the project achieved major milestones in automation and functionality, underscored by significant performance enhancements.

- **CI/CD Automation:** A full **CI/CD pipeline** is now operational using **Jenkins on Kubernetes**. The system automates multi-platform (**ARM64/AMD64**) builds via **Docker-in-Docker** and handles deployments seamlessly.
- **Accountant Portal Launch:** The new portal is live, managing all core financial workflows including **Cash**, **Stripe QR Code**, **Advance Payments**, and **Refunds**. All operations are secured with **Database Transactions** and **Race Condition Prevention**.
- **Performance Breakthrough:** Achieved **near-instant** query speeds by implementing **Composite Indexes** across key tables and leveraging **Redis Caching** for the new **Medical Record History** feature.
- **Patient Experience Upgrades:** Patients can now access their full medical history and utilize an advanced **Appointment Filtering** system built with the **Specification Pattern**.

Day 11

- Thêm CreatedAt, UpdatedAt cho prescriptionItems và hiển thị thời gian tạo, cập nhật status của prescriptionItems.
- Thêm function cho phép complete xét nghiệm để khi trạng thái xét nghiệm là đã xác nhận thì cho bác sĩ yêu cầu huỷ hoặc complete xét nghiệm và không cho xoá nữa.
- Sửa Lỗi Trùng Xét Nghiệm
    - **Vấn đề**: Thêm lại xét nghiệm đã completed/cancelled → Biến mất khi lưu
    - **Giải pháp**: Kiểm tra trùng dựa trên ItemID + Status
- Cải thiện performance cho stored procedure “get_available_doctors” (thời gian truy vấn tăng nhanh)

    - Loại Bỏ JOIN, Dùng EXISTS (EXISTS dừng ngay khi tìm thấy kết quả đầu tiên (short-circuit), không cần xử lý hết tất cả rows như JOIN)
    - Tính Toán Trước Các Giá Trị Thời Gian (Tính 1 lần duy nhất trong phần DECLARE)
    - Dùng Indexes để kiểm tra thời gian (PostgreSQL có thể dùng B-tree index để tìm kiếm nhanh)
    - Thêm Keyword STABLE (Cho PostgreSQL biết function không thay đổi dữ liệu → Tối ưu hóa tốt hơn)
    - Thêm indexes: Doctors (Lọc nhanh bác sĩ), DoctorShifts (Kiểm tra ca làm việc), Appointments (Tìm lịch hẹn trùng)
- Xây Dựng Tính Năng "Xem Hồ Sơ Bệnh Án" Cho Bệnh Nhân
    - Performance:
        - **Backend Caching (Redis)**: Cache kết quả lần đầu, lần sau lấy từ in-memory cache (nhanh gần tức thì). Tự động xóa cache khi bác sĩ cập nhật.
        - **Pagination**: Tải từng trang (10 records/lần), giảm tải ban đầu và network payload.
        - **DTOs**: Chỉ gửi dữ liệu cần thiết qua API, giảm data truyền tải.
    - Maintainability:
        - **Separation of Concerns**: Tách biệt rõ ràng theo Clean Architecture
        - **Controller**: `PatientPortalController` - xử lý request và authorization
        - **Service**: `MedicalRecordApplicationService` - business logic, caching, mapping
        - **Repository**: `MedicalRecordRepository` - truy vấn database
        - **Single Responsibility**: Endpoint mới `GET /api/patient-portal/medical-records` có mục đích duy nhất, rõ ràng

Day 12

- Xây dựng hệ thống lọc appointments cho bệnh nhân theo **Status** và **Date Range** với hiệu suất cao
  - **Database-Level Filtering**
    - Lọc trực tiếp trên DB bằng WHERE clause động, không load toàn bộ data lên app
    - Tận dụng query optimizer của database
  - **Indexing**
    - Index trên `PatientId`, `Status`, `Date` tăng tốc độ query
  - **Không Cache**
    - Filter combinations quá nhiều nên cache hit rate thấp
    - Database đã optimize với indexes đã performance đủ tốt
  - **Specification Pattern**
    - Mỗi filter = 1 class (`AppointmentByStatusSpecification`, `AppointmentByDateRangeSpecification`)
    - Kết hợp linh hoạt: `spec1.And(spec2)`
    - Thêm filter mới: Tạo class mới, không sửa code cũ
  - **Filter DTO**
    - `AppointmentFilterDto` chứa params → Tránh nhiều arguments, code tự document

- Đặt Lịch Theo Bác Sĩ
  - **Quy trình đặt lịch mới:**
    - Chọn chuyên khoa
    - Chọn bác sĩ trong chuyên khoa
    - Chọn ngày bác sĩ có làm việc
    - Chọn khung giờ trống
    - Xác nhận và thanh toán
  - **Maintainability:**

        - Tạo `AppointmentApplicationService`: Tách toàn bộ business logic ra khỏi Controller
        - Controller: Chỉ xử lý HTTP request, delegate sang Service
        - DTOs tập trung: Di chuyển tất cả DTOs vào folder chuyên dụng trong Application layer
    - **Performance & Fix Bugs:**
        - **Query bác sĩ nhanh hơn**: Method `GetActiveShiftDaysAsync` mới, chỉ query data cần thiết
        - **Chuẩn hóa timezone**: Dùng VN timezone (Asia/Ho_Chi_Minh) cho logic, UTC cho database
        - **Tối ưu Bảng DoctorShifts**: Chúng ta thường xuyên truy vấn để tìm các ca làm việc (Shifts) của một bác sĩ (DoctorId) vào một ngày cụ thể trong tuần (DayOfWeek).
            - Giải pháp: Tạo một chỉ mục phức hợp (composite index) trên hai cột DoctorId và DayOfWeek.
            - Lợi ích: Chỉ mục này cho phép PostgreSQL định vị ngay lập tức tất cả các ca làm việc của một bác sĩ cụ thể trong một ngày nhất định mà không cần phải quét toàn bộ bảng. Tốc độ truy vấn sẽ gần như tức thời.
        - **Tối ưu Bảng Appointments**:
            - Tạo một chỉ mục phức hợp tiêu chuẩn (standard composite index) trên hai cột DoctorId và Date.
            - Lợi ích: Giải pháp này mang lại hiệu suất cao. PostgreSQL sẽ sử dụng phần DoctorId của chỉ mục để thu hẹp phạm vi tìm kiếm ngay lập tức, sau đó mới áp dụng hàm date_trunc trên một tập dữ liệu đã rất nhỏ.

Day 13

Accountant Portal

1. **Quy trình nghiệp vụ mới**:
    - Patient thanh toán Tiền mặt (Cash):

    - Kế toán viên (KTV) khởi tạo một thanh toán Pending.
    - KTV xác nhận đã nhận tiền, hệ thống cập nhật trạng thái Payment và MedicalRecord thành Completed.
    - KTV có thể hủy giao dịch đang Pending.
  - Patient thanh toán Stripe (QR Code):
    - KTV khởi tạo thanh toán Stripe, hệ thống tạo một link thanh toán an toàn với hạn sử dụng 30 phút.
    - URL hiển thị dưới dạng mã QR cho bệnh nhân quét.
    - Hệ thống tự động cập nhật trạng thái (Completed/Failed) thông qua Stripe Webhook, đảm bảo dữ liệu luôn chính xác ngay cả khi người dùng đóng trình duyệt.
2. **Maintainability & Security**:
  - Tách biệt Service và Controller: Toàn bộ business logic (khởi tạo, xác nhận, hủy thanh toán) đã được chuyển vào AccountantApplicationService, giúp Controller tinh gọn, chỉ còn nhiệm vụ xử lý HTTP request.
  - Ngăn chặn Race Condition: Triển khai cơ chế kiểm tra và ngăn chặn việc tạo nhiều thanh toán Pending trùng lặp cho cùng một hồ sơ bệnh án.
  - Tái sử dụng & Xử lý Session hết hạn: Hệ thống sẽ thông minh tái sử dụng các phiên thanh toán Stripe còn hạn, và tự động hủy các phiên đã hết hạn trước khi tạo phiên mới.
  - Đảm bảo toàn vẹn dữ liệu: Mọi thao tác thay đổi dữ liệu (tạo, cập nhật payment, medical record) đều được bao bọc trong Database Transaction. Nếu bất kỳ bước nào thất bại, toàn bộ giao dịch sẽ được rollback, đảm bảo dữ
    liệu không bao giờ ở trạng thái không nhất quán.
3. **Performance & Sửa lỗi**:
  - Sửa lỗi nghiêm trọng ở Webhook: Khắc phục một lỗi nghiêm trọng khi thanh toán viện phí qua Stripe thành công nhưng không cập nhật trạng thái cho MedicalRecord. Logic webhook đã được hợp nhất và sửa lỗi để xử lý chính xác cho cả phí đặt lịch và viện phí.

- Tối ưu Bảng `Payments`: Chúng ta thường xuyên phải truy vấn để tìm một thanh toán đang Pending cho một MedicalRecordId cụ thể.
    - Giải pháp: Tạo một chỉ mục phức hợp (composite index) trên hai cột (MedicalRecordId, Status).
    - Lợi ích: Giúp việc truy vấn một thanh toán đang chờ xử lý cho một hồ sơ bệnh án gần như tức thời, cải thiện đáng kể hiệu năng khi KTV thao tác.

Day 14: CI/CD Jenkins

Day 15:

Quy trình Tạm ứng (Advance Payment)

a. Quy trình nghiệp vụ mới:

- Cho phép bệnh nhân đóng trước một khoản phí khám bệnh dựa trên chuyên khoa của bác sĩ.
- Luồng hoạt động:
    1. Lấy danh sách chờ tạm ứng: Hệ thống hiển thị các lịch hẹn đã được xác nhận nhưng chưa hoàn tất tạm ứng.
    2. Gợi ý số tiền: Dựa trên chuyên khoa của bác sĩ, hệ thống tự động gợi ý số tiền tạm ứng (AdvancePaymentSuggestions trong config), giúp KTV thao tác nhanh chóng.
    3. Thanh toán Tiền mặt (Cash):
        - KTV khởi tạo một thanh toán Pending cho khoản tạm ứng.
        - KTV xác nhận đã nhận tiền, hệ thống cập nhật trạng thái thanh toán thành Completed.
        - KTV có thể hủy giao dịch tạm ứng đang Pending.
    4. Thanh toán Stripe (QR Code):

- KTV khởi tạo thanh toán, hệ thống tạo một link thanh toán Stripe an toàn (hạn dùng 30 phút) và hiển thị dưới dạng mã QR.
- Hệ thống tự động lắng nghe và cập nhật trạng thái qua Stripe Webhook khi bệnh nhân hoàn tất thanh toán.

b. Maintainability & Security:

- Tách biệt Service và Controller: Toàn bộ logic nghiệp vụ (lấy danh sách, gợi ý số tiền, khởi tạo, xác nhận, hủy) được đóng gói trong AccountantApplicationService, giúp Controller gọn nhẹ và dễ bảo trì.
- Ngăn chặn Race Condition: Hệ thống kiểm tra chặt chẽ, không cho phép tạo thanh toán tạm ứng mới nếu đã tồn tại một thanh toán khác đang ở trạng thái Pending cho cùng một lịch hẹn.
- Xử lý Session Stripe thông minh:
  - Tái sử dụng: Nếu một phiên thanh toán Stripe cho lịch hẹn vẫn còn hạn, hệ thống sẽ tái sử dụng link QR cũ thay vì tạo mới.
  - Tự động hủy: Nếu phiên đã hết hạn, hệ thống sẽ tự động đánh dấu thanh toán cũ là Failed trước khi tạo phiên mới, tránh treo thanh toán.
- Đảm bảo toàn vẹn dữ liệu: Mọi thao tác tạo/cập nhật thanh toán đều được thực thi trong một Database Transaction. Nếu có lỗi xảy ra, toàn bộ thay đổi sẽ được rollback, đảm bảo dữ liệu luôn nhất quán.

c. Performance:

- Tối ưu truy vấn: Để tìm kiếm nhanh các thanh toán tạm ứng đang chờ xử lý cho một lịch hẹn, một chỉ mục phức hợp (composite index) được đề xuất trên các cột (AppointmentId, Status) của bảng Payments. Điều này giúp giảm
  độ trễ khi KTV cần kiểm tra hoặc tái sử dụng một phiên thanh toán.
- Webhook hợp nhất: Logic của Stripe Webhook đã được cải tiến để xử lý đồng nhất cho cả hai loại thanh toán (Deposit và FinalPayment), đảm bảo trạng thái của Payment và MedicalRecord luôn được cập nhật chính xác và kịp
  thời.

Quy trình Hoàn tiền (Refund Process)

a. Quy trình nghiệp vụ mới:

- Xử lý các trường hợp bệnh nhân đã trả viện phí nhiều hơn chi phí thực tế (PaidAmount > TotalFee).
- Luồng hoạt động:
    1. Lấy danh sách chờ hoàn tiền: KTV truy cập danh sách các hồ sơ bệnh án (MedicalRecord) có đủ điều kiện hoàn tiền.
    2. Khởi tạo hoàn tiền: KTV chọn hồ sơ và khởi tạo yêu cầu hoàn tiền. Hệ thống sẽ tạo một giao dịch âm (Payment với số tiền âm) ở trạng thái Pending.
    3. Hoàn tất hoàn tiền: Sau khi đã trả lại tiền mặt cho bệnh nhân, KTV xác nhận trên hệ thống. Trạng thái của giao dịch hoàn tiền sẽ chuyển thành Completed.
    4. Cập nhật hồ sơ: PaidAmount của MedicalRecord được tính toán lại. Trạng thái PaymentStatus của hồ sơ được cập nhật thành Paid (vì lúc này PaidAmount đã bằng TotalFee).

b. Maintainability & Security:

- Logic tập trung: Toàn bộ quy trình được quản lý trong AccountantApplicationService, đảm bảo tính nhất quán và dễ dàng nâng cấp.
- Ngăn chặn yêu cầu trùng lặp: Hệ thống không cho phép KTV khởi tạo một yêu cầu hoàn tiền mới cho một hồ sơ bệnh án nếu đã có một yêu cầu hoàn tiền khác đang Pending.
- Toàn vẹn dữ liệu sau hoàn tiền: Quy trình hoàn tất hoàn tiền và cập nhật lại MedicalRecord được bảo vệ bởi Database Transaction. Điều này đảm bảo hồ sơ bệnh án không bao giờ bị kẹt ở trạng thái "đã trả thừa" sau khi tiền đã được hoàn.

c. Performance & Sửa lỗi:

- Sửa lỗi logic hiển thị: Khắc phục vấn đề các hồ sơ đã được hoàn tiền hoặc đang chờ hoàn tiền vẫn có thể xuất hiện trong danh sách cần thanh toán cuối cùng. Logic của GetUnpaidMedicalRecordsAsync đã được điều chỉnh để
  lọc bỏ các hồ sơ có PaidAmount >= TotalFee, đảm bảo danh sách thanh toán chỉ chứa các khoản nợ hợp lệ.

- Tối ưu truy vấn tìm kiếm: Tương tự các quy trình khác, việc tìm kiếm một thanh toán Pending (bao gồm cả thanh toán hoàn tiền) cho một MedicalRecordId được tối ưu bằng chỉ mục phức hợp (composite index) trên (MedicalRecordId, Status) của bảng Payments, giúp các thao tác của KTV diễn ra nhanh chóng.