BÀI KIỂM TRA 02 - PHIÊN BẢN NÂNG CAO (ADVANCED EDITION)

Mức độ: Khó (Hardcore) **Môn học:** Lập trình Fullstack Development **Công nghệ:** Vue.js (Frontend), ASP.NET Core Web API (Backend), Clean Architecture, Entity Framework Core, Identity, Redis, SignalR, Hangfire, Docker.

TÊN ĐỀ TÀI: HỆ THỐNG QUẢN LÝ CLB PICKLEBALL "VỢT THỦ PHỐ NÚI" (PCM) - PRO EDITION

TỔNG QUAN

Đây là phiên bản nâng cấp toàn diện của hệ thống PCM. Ngoài các nghiệp vụ cơ bản, sinh viên phải giải quyết các bài toán về **Hiệu năng (Performance)**, **Tính toàn vẹn dữ liệu (Data Integrity)**, **Trải nghiệm thời gian thực (Real-time)** và **Kiến trúc hệ thống (System Architecture)**.

PHẦN 1: MÔ TẢ BÀI TOÁN & NGHIỆP VỤ (FULL SCENARIO)

CLB "Vợt Thủ Phố Núi" hoạt động chuyên nghiệp hóa. Hệ thống cần đáp ứng các nghiệp vụ sâu sát sau:

1. Quản trị Nội bộ & Tài chính số (Operations & Fintech)

- **Quản lý Hội viên (Members):**
    - Mỗi thành viên có Hồ sơ số (Digital Profile) và Rank DUPR (tính theo thuật toán ELO).
    - Tích hợp Identity để quản lý tài khoản bảo mật.
- **Ví điện tử (E-Wallet) - *Nâng cao*:**
    - Thay vì quản lý thu chi thủ công, mỗi hội viên có một Ví điện tử nội bộ.
    - **Nạp tiền (Top-up):** Member tạo request nạp tiền (kèm ảnh bill). Admin/Treasurer duyệt -> Tiền cộng vào Ví (transaction safe).
    - **Thanh toán:** Hệ thống tự động trừ tiền trong Ví khi Đặt sân hoặc tham gia Giải đấu. Chặn hành động nếu số dư không đủ.
    - **Lịch sử giao dịch:** Ghi lại mọi biến động số dư (Nạp, Trừ tiền sân, Hoàn tiền, Thưởng giải).
- **Tin tức & Thông báo:** Đăng tải thông báo, vinh danh. Hỗ trợ ghim tin (IsPinned) và cache phía server.

2. Hoạt động Sân bãi Thông minh (Smart Booking)

- **Đặt sân (Booking):**
    - **Cơ bản:** Chọn sân, chọn giờ, check trùng lịch.
    - **Nâng cao (Recurring):** Cho phép đặt lịch cố định (VD: "Mỗi thứ 3, 5 lúc 17:00 trong 1 tháng").
    - **Conflict Handling:** Khi đặt định kỳ, nếu có ngày bị trùng, hệ thống phải chỉ rõ ngày nào trùng và cho phép tùy chọn (Bỏ qua ngày đó / Hủy tất cả).

    - **Concurrency Control:** Xử lý bài toán 2 người cùng bấm đặt 1 sân cùng lúc (Race Condition) bằng Optimistic Locking hoặc Pessimistic Locking.
    - **Auto-Cancel:** Booking ở trạng thái "Chờ thanh toán" quá 15 phút không thanh toán sẽ bị Job ngầm (Hangfire) tự động hủy.

- **Match Making & Giao hữu:**

    - Kết quả trận đấu ảnh hưởng trực tiếp đến Rank DUPR theo thuật toán ELO (thay vì cộng trừ điểm tĩnh).

3. Sàn đấu & Hệ thống Giải (Tournament System)

- **Kèo Thách đấu (Duel):** 1vs1 hoặc 2vs2 vui vẻ, phần thưởng nhỏ.
- **Giải đấu Mini (Mini-game):**
    - **Team Battle:** Chia phe A-B, đánh chạm mốc thắng.
    - **Round Robin:** Đánh vòng tròn tích điểm.
- **Giải đấu Chuyên nghiệp (Tournament Bracket) - *Nâng cao*:**
    - Hỗ trợ thể thức **Loại trực tiếp (Knockout)**.
    - Hệ thống tự động tạo và vẽ **Cây thi đấu (Bracket Tree)**: Tứ kết -> Bán kết -> Chung kết.
    - Check-in online trước giờ đấu (nếu không check-in sẽ bị loại).

PHẦN 2: YÊU CẦU KỸ THUẬT & KIẾN TRÚC (BẮT BUỘC)

Sinh viên **KHÔNG** viết code dồn hết vào Controller/Model đơn giản. Phải tuân thủ:

1. Kiến trúc Backend (Clean Architecture)

Phân chia Project thành các Layer rõ ràng:

- **Core/Domain:** Chứa Entities, Value Objects, Domain Services, Interfaces (Repository). Tuyệt đối không phụ thuộc vào Database hay Framework bên ngoài.
- **Application:** Chứa Use Cases, DTOs, Services Implementation, Validation, AutoMapper.
- **Infrastructure:** Triển khai DbContext, EfRepository, Redis, SignalR service, External API.
- **API (Presentation):** Controllers (rất mỏng), Middleware, Filters.

2. Database & Data Consistency

- Sử dụng **Code First** và **Migration**.
- Áp dụng **Repository Pattern** và **Unit of Work** để đảm bảo tính toàn vẹn dữ liệu (đặc biệt là giao dịch Ví và Đặt sân).
- **Concurrency Token:** Xử lý xung đột dữ liệu (RowVersion).

3. Hiệu năng & Real-time (Redis & SignalR)

- **Redis Caching:**
    - Cache thông tin Courts, Configs (ít thay đổi).
    - Cache Bảng xếp hạng Top Ranking (Sử dụng Redis Sorted Sets để truy vấn Top 100 cực nhanh).
- **SignalR (Real-time):**
    - **Live Scoreboard:** Cập nhật tỷ số trận đấu ngay lập tức cho khán giả.

- **Booking State:** Khi A vừa đặt Sân 1, màn hình B phải hiện Sân 1 "Đã khóa" ngay lập tức (không cần refresh).
- **Notifications:** Thông báo nạp tiền thành công, đến lượt thi đấu.

4. Background Jobs (Hangfire / Quartz)

- Job quét Booking "treo" (Pending quá 15p) để hủy.
- Job tổng kết ngày: Gửi báo cáo doanh thu, rank thay đổi vào cuối ngày.

PHẦN 3: THIẾT KẾ CƠ SỞ DỮ LIỆU (DATABASE SCHEMA)

Tên bảng bắt đầu bằng `3 số cuối MSSV`.

1. Core & Identity

- `[xxx]_Members`: Id, UserId (Identity), FullName, RankELO (double), WalletBalance (decimal), AvatarUrl.
- `[xxx]_RefreshTokens`: Id, UserId, Token, JwtId, IsUsed, IsRevoked, AddedDate, ExpiryDate. (Dùng cho cơ chế Refresh Token bảo mật).

2. Tài chính (Fintech)

- `[xxx]_Wallets` (Có thể gộp vào Member hoặc tách riêng): MemberId, Balance, EncryptedSign (mã hóa để chống sửa DB thủ công - *Bonus*).
- `[xxx]_WalletTransactions`: Id, WalletId, Amount (+/-), Type (Deposit/PayBooking/Prize/Refund), ReferenceId (BookingId/MatchId), Status, CreatedDate.

3. Sân bãi & Booking

- `[xxx]_Courts`: Id, Name, IsActive.
- `[xxx]_Bookings`: Id, CourtId, MemberId, StartTime, EndTime, TotalPrice, Status (Pending/Paid/Cancelled/CheckedIn), CreatedDate, **RowVersion** (timestamp).

4. Giải đấu & Trận đấu

- `[xxx]_Tournaments` (Nâng cấp từ Challenges cũ): Id, Name, Type (Duel/League/Knockout), Status, Configs.
- `[xxx]_TournamentMatches` (Cây đấu): Id, TournamentId, Round (1=Vòng bảng, 2=Tứ kết...), MatchId, NextMatchId (để vẽ cây), Player1Id, Player2Id, WinnerId.
- `[xxx]_Matches`: Id, Date, IsRanked, Player1Id, Player2Id (Single) / Team1Ids, Team2Ids (Double), ScoreT1, ScoreT2, WinnerSide, EloChange.

PHẦN 4: YÊU CẦU API & FRONTEND

1. Backend API (ASP.NET Core)

- **Auth:** Login/Register trả về JWT + Refresh Token. Endpoint `/refresh-token` để cấp lại Access Token mới.
- **Wallet Ops:**

    - `POST /api/wallet/deposit` (Upload ảnh bill).
    - `GET /api/wallet/history`.
- **Booking Ops:**
    - `POST /api/bookings` (Xử lý trừ tiền Ví + Lock row + SignalR broadcast).
    - `POST /api/bookings/recurring` (Xử lý lặp).
- **Tournament Ops:**
    - `GET /api/tournaments/{id}/bracket` (Trả về cấu trúc cây đấu).
- **Caching:** Áp dụng `[ResponseCache]` cho các API tin tức, config.

2. Frontend (Vue.js)

- **Kiến trúc:** Vue 3, Composition API, Pinia (Store).
- **Real-time UX:**
    - Kết nối SignalR Hub khi App start.
    - Lắng nghe event `BookingUpdate`, `ScoreUpdate`, `WalletUpdate`.
- **Bracket View:** Vẽ cây thi đấu trực quan cho giải Knockout.
- **Payment UX:** Confirm Modal "Bạn có chắc muốn thanh toán 50k từ Ví?" -> Animation trừ tiền -> Success.

3. Deployment (Docker)

- Viết **Dockerfile** cho API và Vue App (Multi-stage build).
- Viết **docker-compose.yml** khởi chạy stack:
    - `sql-server`
    - `redis`
    - `pcm-backend`
    - `pcm-frontend` (nginx)
- Chỉ cần chạy `docker-compose up` là hệ thống dựng sẵn sàng.

PHẦN 5: TIÊU CHÍ ĐÁNH GIÁ

1. **Kiến trúc & Code Quality (30%):** Tuân thủ Clean Architecture, Code tách biệt, dễ đọc, dễ mở rộng. DI, Service Pattern chuẩn.
2. **Nghiệp vụ Phức tạp (30%):**
    - Ví điện tử hoạt động đúng, không âm tiền.
    - Đặt sân định kỳ & xử lý tranh chấp (Concurrency) chuẩn.
    - Thuật toán ELO & Vẽ cây đấu loại.
3. **Công nghệ Nâng cao (30%):**
    - Redis Cache hoạt động hiệu quả.
    - SignalR chạy mượt (Test mở 2 trình duyệt, thay đổi bên này, bên kia cập nhật ngay).
    - Hangfire chạy đúng giờ.
4. **Deployment & UI (10%):**
    - Docker Compose chạy được.
    - Giao diện đẹp, chuyên nghiệp (Dark mode là điểm cộng).

Lưu ý: Đây là đề bài dành cho các bạn muốn đạt điểm tối đa (A+) và rèn luyện kỹ năng thực tế của một Senior/Lead Developer tương lai. Chúc may mắn!