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 việc đáp ứng các nghiệp vụ quản lý CLB 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 dựa trên tinh thần "Vui - Khỏe - Có Thưởng". Hệ thống PCM Pro cần quản lý sâu sát các hoạt động đặc thù sau:

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

- **Quản lý Hội viên (Members):**
    - Mỗi thành viên có một Hồ sơ số (Digital Profile) bao gồm thông tin cá nhân (Email, SĐT) và Rank DUPR (Trình độ).
    - Rank DUPR biến động tự động theo kết quả thi đấu dựa trên thuật toán **ELO Rating** (thay vì cộng trừ điểm tĩnh).
    - Tích hợp Identity để quản lý tài khoản bảo mật.
- **Ví điện tử (E-Wallet) - Thay thế quản lý sổ sách thủ công:**
    - Mỗi hội viên sở hữu một Ví điện tử nội bộ trong hệ thống.
    - **Nạp tiền (Top-up):** Member tạo yêu cầu nạp tiền (kèm ảnh bill chuyển khoản). Admin/Treasurer duyệt -> Tiền cộng vào Ví (Transaction Safe).
    - **Thanh toán tự động:** Hệ thống tự động kiểm tra và trừ tiền trong Ví khi thành viên Đặt sân hoặc Đăng ký giải đấu. Chặn hành động ngay lập tức nếu số dư không đủ.
    - **Lịch sử giao dịch (Transaction History):** Ghi lại minh bạch mọi biến động số dư (Nạp tiền, Trừ tiền sân, Hoàn tiền hủy sân, Nhận thưởng giải đấu).
    - **Cảnh báo:** Hệ thống không cho phép Ví bị âm.
- **Tin tức & Thông báo:**
    - Đăng tải thông báo, lịch nghỉ, vinh danh.
    - Hỗ trợ ghim tin (IsPinned).
    - **Yêu cầu hiệu năng:** Sử dụng Redis Cache để cache nội dung tin tức, giảm tải cho Database.

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

- **Đặt sân (Booking):**
    - Thành viên xem lịch trống, chọn khung giờ và đặt sân.
    - **Chặn trùng lịch:** Tuyệt đối không để xảy ra 2 đơn đặt chồng chéo thời gian trên cùng một sân.
    - **Tính năng Nâng cao - Đặt định kỳ (Recurring Booking):** Cho phép đặt lịch cố định (Ví dụ: "Mỗi thứ 3, 5 hàng tuần lúc 17:00 trong 1 tháng").
    - **Xử lý xung đột (Conflict Handling):** Khi đặt định kỳ, nếu có 1-2 ngày bị trùng lịch với người khác, hệ thống phải phát hiện và cho user tùy chọn (Bỏ qua ngày trùng / Hủy toàn bộ).
    - **Concurrency Control (Quan trọng):** Giải quyết bài toán Race Condition khi 2 người cùng bấm đặt 1 sân tại cùng 1 thời điểm. Áp dụng Optimistic Locking (RowVersion) 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 để nhả sân.
- **Match Making & Giao hữu:**
    - Kết quả trận đấu giao hữu (Daily Matches) sẽ ảnh hưởng trực tiếp đến Rank DUPR.
    - Hệ thống tính toán điểm cộng/trừ dựa trên chênh lệch trình độ giữa 2 bên (ELO).

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

- **Các hình thức thi đấu:**
    - **Kèo Thách đấu (Duel):** 1vs1 hoặc 2vs2, phần thưởng nhỏ vui vẻ.
    - **Giải đấu Mini (Mini-game):**
        - **Team Battle:** Chia 2 phe A-B, đánh chạm mốc thắng (VD: đội nào thắng 5 trận trước thì ăn trọn quỹ thưởng).
        - **Round Robin:** Đánh vòng tròn tích điểm cá nhân.
- **Giải đấu Chuyên nghiệp (Tournament Bracket) - *Tính năng Nâng cao*:**
    - Hỗ trợ thể thức **Loại trực tiếp (Knockout)**.
    - Hệ thống tự động tạo và vẽ **Cây thi đấu (Bracket Tree)**: Vòng Bảng -> Tứ kết -> Bán kết -> Chung kết.
    - Quản lý 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 tất cả logic vào Controller. Phải tuân thủ nghiêm ngặt các pattern sau:

1. Kiến trúc Backend (Clean Architecture)

Phân chia Project thành các Layer rõ ràng (Solution 4 project con trở lên):

- **Core (Domain):** Chứa Entities, Value Objects, Domain Services, Interfaces (Repository). Tuyệt đối không phụ thuộc vào Infrastructure hay Web Framework.
- **Application:** Chứa Use Cases, Services Implementation, DTOs, Validation (FluentValidation), AutoMapper.

- **Infrastructure:** Triển khai DbContext, EfRepository, Redis Service, SignalR Service, Email Service.
- **API (Presentation):** Controllers (rất mỏng, chỉ nhận request và gọi Service), 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 (Transaction management). Đặc biệt quan trọng khi xử lý giao dịch Ví và Đặt sân (phải commit cùng lúc hoặc rollback toàn bộ).
- **Concurrency Token:** Sử dụng `RowVersion` (Timestamp) để xử lý xung đột dữ liệu.

3. Hiệu năng & Real-time

- **Redis Caching:**
  - Cache thông tin Courts, Configurations (dữ liệu ít thay đổi).
  - **Leaderboard Cache:** Sử dụng Redis Sorted Sets để lưu và truy vấn BXH Top Ranking thời gian thực với tốc độ cao.
- **SignalR (Real-time):**
  - **Live Scoreboard:** Cập nhật tỷ số trận đấu ngay lập tức cho khán giả (không cần F5).
  - **Booking State:** Khi Members A vừa đặt thành công Sân 1, màn hình Members B phải chuyển Sân 1 sang trạng thái "Đã khóa" ngay lập tức.
  - **Notifications:** Thông báo nạp tiền thành công, đến lượt thi đấu.

4. Background Jobs (Hangfire)

- Job quét Booking "treo" (Pending quá 15p) để hủy.
- Job tổng kết ngày: Gửi báo cáo doanh thu, tính lại ranking định kỳ (nếu cần) vào cuối ngày.

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

Sinh viên thiết kế database tuân thủ quy tắc Clean Architecture. Tên bảng bắt đầu bằng 3 số cuối MSSV.

1. Nhóm Quản trị & Identity (Operations Core)

- `[xxx]_Members`**:**
  - Id, **UserId** (FK -> AspNetUsers - Bắt buộc), FullName.
  - Email, PhoneNumber (Có thể sync từ Identity hoặc lưu riêng).
  - DateOfBirth (DateTime?), JoinDate (DateTime).
  - **RankELO** (double) - Điểm trình độ thực tế (Mặc định 1200 hoặc theo input).
  - **WalletBalance** (decimal) - Số dư ví hiện tại (Mặc định 0). *Lưu ý: Có thể tách bảng Wallets riêng nếu muốn strict separation, nhưng để đây cho tiện truy vấn cũng được.*
  - AvatarUrl, IsActive (bool).
  - **RowVersion** (timestamp) - Dùng cho concurrency check khi update profile/balance.
- `[xxx]_RefreshTokens`**:** (Bắt buộc cho JWT Security)
  - Id, UserId (FK), Token (string), JwtId (string).
  - IsUsed (bool), IsRevoked (bool).

    - AddedDate, ExpiryDate.

- `[xxx]_News`:
    - Id, Title, Content, IsPinned (bool) - Tin ghim sẽ được cache.
    - CreatedDate, CreatedBy.

2. Nhóm Tài chính (Fintech Module)

- `[xxx]_TransactionCategories`:
    - Id, Name (VD: "Nạp tiền", "Phí sân", "Thưởng giải"), Type (Enum: Thu/Chi).
- `[xxx]_WalletTransactions`:
    - Id, Date (DateTime), Amount (decimal).
    - **WalletId** (hoặc MemberId), CategoryId (FK).
    - **Type** (Enum: Deposit, PayBooking, ReceivePrize, Refund).
    - **ReferenceId** (string/int) - ID của Booking/Match/Tournament liên quan.
    - Description, **Status** (Enum: Pending, Success, Failed, Cancelled).
    - **EncryptedSignature** (string) - *Bonus*: Chuỗi hash (SHA256) của giao dịch để chống sửa DB.

3. Nhóm Sân bãi & Đặt sân (Booking Module)

- `[xxx]_Courts`:
    - Id, Name, IsActive, Description.
- `[xxx]_Bookings`:
    - Id, **CourtId** (FK), **MemberId** (FK - Người đặt).
    - StartTime, EndTime (DateTime).
    - **TotalPrice** (decimal) - Số tiền đã bị trừ.
    - **Status** (Enum: PendingPayment, Confirmed, Cancelled, CheckedIn).
    - Note, CreatedDate.
    - **RowVersion** (timestamp) - **BẮT BUỘC** để xử lý Optimistic Locking khi 2 người cùng đặt.

4. Nhóm Giải đấu & Trận đấu (Tournament Module)

- `[xxx]_Tournaments` (Thay thế Challenges ở bản thường):
    - Id, Name, StartDate, EndDate.
    - **Type** (Enum: Duel, MiniGame, Professional).
    - **Format** (Enum: RoundRobin, Knockout, TeamBattle).
    - **Status** (Enum: Open, Ongoing, Finished).
    - EntryFee (decimal), PrizePool (decimal).
    - **ConfigData** (JSON) - Lưu cấu hình cây đấu hoặc rules.
- `[xxx]_Participants`: (Người tham gia giải)
    - Id, TournamentId (FK), MemberId (FK).
    - **TeamName** (hoặc Team A/B), Status (Registered, Paid, Eliminated).

    - SeedNo (Số hạt giống - nếu có).

- `[xxx]_TournamentMatches` (Cấu trúc Cây đấu - Cho Knockout):
    - Id, TournamentId (FK), **Round** (int: 1=Vòng bảng, 2=Tứ kết, 3=Bán kết...).
    - **MatchId** (FK -> Matches).
    - **NextMatchId** (FK) - Trận tiếp theo mà người thắng sẽ tiến vào.
    - **ParentMatchId** (FK) - Để truy vết ngược.
    - **BracketGroup** (string: WinnerBracket, LoserBracket - nếu đánh Double Elimination).
- `[xxx]_Matches` (Lõi lưu hoạt động thi đấu):
    - Id, Date, **IsRanked** (bool).
    - **MatchFormat** (Enum: Singles, Doubles).
    - **Team 1:** Player1Id, Player2Id (nullable).
    - **Team 2:** Player1Id, Player2Id (nullable).
    - **Score:** ScoreTeam1, ScoreTeam2.
    - **Result:** WinnerSide (Enum: None, Team1, Team2, Draw).
    - **EloChange** (double) - Điểm Elo thay đổi sau trận.
    - **TournamentId** (nullable) - Null nếu là kèo giao hữu hằng ngày.

PHẦN 4: YÊU CẦU API & FRONTEND

1. Backend API (ASP.NET Core Web API)

Sử dụng Clean Architecture, CQRS (nếu có thể), trả về chuẩn RESTful.

Auth & Identity:

- `POST /api/auth/login`: Trả về AccessToken + RefreshToken.
- `POST /api/auth/register`: Đăng ký thành viên mới.
- `POST /api/auth/refresh-token`: Cấp lại token mới.
- `POST /api/auth/revoke-token`: Thu hồi token (Logout).

Member & Wallet:

- `GET /api/members/me`: Profile cá nhân + Số dư ví.
- `PUT /api/members/profile`: Cập nhật avatar, thông tin.
- `POST /api/wallet/deposit`: Member gửi yêu cầu nạp tiền (kèm ảnh bằng chứng).
- `POST /api/wallet/approve-deposit`: Admin duyệt yêu cầu -> Cộng tiền (+SignalR notify).
- `GET /api/wallet/transactions`: Lịch sử giao dịch cá nhân.

Booking (Quy trình phức tạp nhất):

- `GET /api/bookings/slots`: Lấy danh sách slot trống theo ngày/sân (Check Redis cache trước).
- `POST /api/bookings`: Đặt sân đơn lẻ.
    - Logic: Check Valid -> Lock Row -> Trừ tiền Ví -> Create Booking -> Commit Transaction -> Bắn SignalR update slot.
- `POST /api/bookings/recurring`: Đặt lịch định kỳ (VD: Thứ 2-4-6).
    - Logic: Loop date -> Check conflict từng ngày -> Trả về list Success/Fail dates.

- `PUT /api/bookings/{id}/cancel`: Hủy đặt sân (Hoàn tiền % theo quy định).

Tournament & Bracket:

- `GET /api/tournaments`: Danh sách giải đấu.
- `POST /api/tournaments/{id}/join`: Đăng ký & Trừ tiền EntryFee.
- `GET /api/tournaments/{id}/bracket`: **QUAN TRỌNG**. Trả về cấu trúc cây đấu (Nodes & Edges) để frontend vẽ.
- `POST /api/tournaments/{id}/generate-bracket`: Admin kích hoạt tạo cây đấu ngẫu nhiên/theo hạt giống.

Matches (Referee/Admin):

- `POST /api/matches`: Tạo trận giao hữu (Daily Match).
- `PUT /api/matches/{id}/result`: Cập nhật kết quả.
    - Logic: Tính lại ELO 2 bên -> Update Member Rank -> Update Bracket (nếu là giải đấu) -> Push SignalR Live Score.

2. Frontend (Vue.js 3 + Pinia)

- **Kiến trúc:** Modular (tách views, components, stores, services).
- **Core Features:**
    - Login/Register/Profile.
    - **My Wallet:** Trang ví, hiện QR chuyển khoản, upload bill, xem lịch sử biến động.
- **Booking UI:**
    - Lịch trực quan (Calendar view), kéo thả hoặc click chọn slot.
    - Hiển thị realtime: Slot nào vừa có người đặt phải chuyển màu Đỏ ngay lập tức.
- **Tournament UI:**
    - **Bracket View:** Sử dụng thư viện (như `vue-tournament-bracket` hoặc tự vẽ bằng SVG/Canvas) để hiển thị sơ đồ thi đấu Knockout.
- **Real-time Widget:**
    - Góc màn hình hiển thị Toast Notification khi: Được cộng tiền, Có người thách đấu, Trận đấu đang theo dõi có kết quả.

3. Deployment (Dockerize)

- **Dockerfile Backend:** Multi-stage build (Build image -> Runtime image).
- **Dockerfile Frontend:** Build Vue -> Production (Dist) -> Serve bằng Nginx.
- **docker-compose.yml:** Orchestration toàn bộ:
    - SQL Server (Volume persistent).
    - Redis (Cache).
    - Backend API.
    - Frontend (Nginx).

PHẦN 5: QUY ĐỊNH NỘP BÀI & ĐÁNH GIÁ

Cách thức nộp bài:

1. Code đầy đủ đẩy lên GitHub (Chia folder rõ ràng: `/backend`, `/frontend`, `/docker`).

2. Có file `README.md` chi tiết hướng dẫn cách run Docker Compose và tài khoản Admin/Member mẫu.
3. Gửi link GitHub vào nhóm Zalo đúng cú pháp.

Tiêu chí đánh giá (Thang 10):

1. **Kiến trúc & Code Quality (30%):**
    - Cấu trúc Clean Architecture chuẩn, tách biệt layer.
    - Sử dụng Dependency Injection, Service Pattern hợp lý.
    - Code clean, naming convention chuẩn.
2. **Nghiệp vụ Chuyên sâu (30%):**
    - Logic Ví điện tử chặt chẽ (Transactional), không xảy ra lỗi âm tiền hay mất tiền.
    - Logic Đặt sân định kỳ & Xử lý tranh chấp (Optimistic Locking) hoạt động đúng.
    - Thuật toán ELO & Logic vẽ/update cây đấu loại (Knockout).
3. **Công nghệ Nâng cao (30%):**
    - **Redis:** Có cache các dữ liệu tĩnh và Leaderboard.
    - **SignalR:** Real-time hoạt động mượt mà giữa 2 trình duyệt.
    - **Hangfire:** Job chạy ngầm đúng lịch (hủy booking treo).
    - **Docker:** Build và Run thành công trên máy giảng viên.
4. **UI/UX & Hoàn thiện (10%):**
    - Giao diện đẹp, hiện đại, Responsive.
    - Các hiệu ứng loading, notification mượt mà.

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 các em thành công!