# BÀI KIỂM TRA 02

**Môn học:** Lập trình Fullstack Development
**Công nghệ:** Vue.js (Frontend), ASP.NET Core Web API (Backend), Entity Framework Core, Identity, SQL Server, JWT Authentication.

---

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

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

CLB "Vợt Thủ Phố Núi" hoạt động dựa trên tinh thần "Vui - Khỏe - Có Thưởng". Hệ thống PCM cần quản lý sâu sát các hoạt động đặc thù sau:

### 1. Quản trị Nội bộ (Operations) - "Xương sống của CLB"

- **Quản lý Hội viên:**
    - Mỗi thành viên có một Hồ sơ số (Digital Profile) gồm thông tin cá nhân và Rank DUPR (Trình độ) biến động theo kết quả thi đấu.
    - Mỗi hội viên phải có tài khoản Identity để đăng nhập hệ thống. Thông tin trong bảng Members (Email, PhoneNumber) có thể lấy từ Identity User hoặc lưu riêng để bổ sung.
- **Quản lý Tài chính (Treasury):**
    - Quản lý dòng tiền Thu/Chi minh bạch. Mỗi giao dịch phải có danh mục (Category) để phân loại.
    - Admin có thể thêm mới các Danh mục Thu/Chi (Ví dụ: Thêm mục "Tiền sửa đèn", "Tiền tất niên").
    - Hệ thống phải Cảnh báo đỏ ngay lập tức nếu Quỹ bị âm (tổng Thu - tổng Chi < 0).
- **Tin tức & Thông báo:** Đăng tải thông báo đóng quỹ, lịch nghỉ, vinh danh thành viên... Tin có thể được ghim (IsPinned) để hiển thị ưu tiên.

### 2. Hoạt động Thường nhật (Daily Activities)

- **Đặt sân (Booking):**
    - Thành viên xem lịch theo từng sân (Court), chọn khung giờ (StartTime - EndTime) và đặt sân.
    - Hệ thống **chặn trùng lịch** trên cùng sân: Không cho phép hai booking có thời gian giao nhau trên cùng một Court.
    - Ví dụ: Nếu Sân 1 đã có booking từ 8:00-9:00, thì không thể đặt Sân 1 từ 8:30-9:30 (trùng lịch).
    - Booking cũng có thể mang ý nghĩa là lời hẹn nhau xuống thi đấu (khung giờ) để cho mọi người biết.
- **Trận đấu Giao hữu (Daily Matches):**
    - Đây là hoạt động diễn ra nhiều nhất. Hội viên ra sân đánh Đơn (1vs1) hoặc Đôi (2vs2).
    - Kết quả trận đấu sẽ làm tăng/giảm điểm Rank DUPR của người chơi (nếu IsRanked = true).

### 3. Sàn đấu & Sự kiện (Challenges & Tournaments)

- **Kèo Thách đấu (Duel):**
  - Một người thách đấu người khác (1vs1 hoặc 2vs2).
  - Phần thưởng vui vẻ (Nước, Trứng) - không có Entry Fee hoặc PrizePool lớn.
  - Type = Duel, GameMode = None.
- **Giải đấu Mini (Mini-game):**
  - Nhóm hội viên (10-20 người) góp tiền (Entry Fee) tạo giải.
  - **Entry Fee:** Số tiền mỗi người phải đóng để tham gia (VD: 50.000đ/người).
  - **PrizePool:** Tổng quỹ thưởng = Entry Fee × Số người tham gia (tự động tính hoặc admin nhập).
  - **Thể thức Team Battle:**
    - Chia 2 phe A-B. Có thể phân chia tự động theo Rank để cân sức (người rank cao nhất → Team A, rank thứ 2 → Team B, rank thứ 3 → Team A...).
    - Đánh chạm mốc thắng: Phe nào thắng đủ số trận (Config_TargetWins, VD: 5 trận) trước thì ăn trọn quỹ.
    - Type = MiniGame, GameMode = TeamBattle.
  - **Thể thức Vòng tròn (RoundRobin):**
    - Đánh xoay vòng tích điểm cá nhân. Mỗi người đấu với nhiều người khác, tích điểm theo kết quả.
    - Xếp hạng theo tổng điểm, top 1/2/3 nhận thưởng.
    - Type = MiniGame, GameMode = RoundRobin.
  - **Lưu ý:** Dù là giải đấu kiểu gì, thì các trận đấu con bên trong (Matches) vẫn phải tuân thủ luật Đơn/Đôi.

---

## PHẦN 2: THIẾT KẾ CƠ SỞ DỮ LIỆU (DATABASE SCHEMA)

Sinh viên sử dụng Code First và Migration. Tên bảng nghiệp vụ bắt đầu bằng 3 số cuối MSSV.

### 1. Nhóm Quản trị (Operations)

- `[xxx]_Members`:
  - **Cơ bản:** Id, FullName, JoinDate, RankLevel (Double), Status (hoặc IsActive - bool).
  - **Bổ sung (bắt buộc):**
    - UserId (FK → AspNetUsers) - Liên kết với tài khoản Identity để đăng nhập.
    - Email, PhoneNumber - Có thể lấy từ Identity hoặc lưu riêng để bổ sung thông tin.
    - DateOfBirth (DateTime?, nullable) - Ngày sinh.
  - **Khuyến nghị:** TotalMatches (int), WinMatches (int), CreatedDate, ModifiedDate.
- `[xxx]_News`:
  - Id, Title, Content, IsPinned (bool) - Tin được ghim sẽ hiển thị ưu tiên.
  - Khuyến nghị thêm CreatedDate.
- `[xxx]_TransactionCategories`:
  - Id, Name (VD: "Tiền sân", "Nước", "Quỹ tháng", "Phạt"...), Type (Enum: Thu / Chi).

- `[xxx]_Transactions`:
    - Id, Date (DateTime), Amount (decimal), Description (string), CategoryId (FK → TransactionCategories).
    - Khuyến nghị: CreatedBy (FK → Members), CreatedDate.

## 2. Bảng Sân bóng (Courts) – BẮT BUỘC

- `[xxx]_Courts`:
    - Id, Name (string, VD: "Sân 1", "Sân 2"), IsActive (bool), Description (string, nullable).

## 3. Nhóm Thi đấu (Sports Core) – QUAN TRỌNG

- `[xxx]_Bookings`:
    - Id, **CourtId (FK → Courts)** - Bắt buộc phải chọn sân, MemberId (FK → Members), StartTime (DateTime), EndTime (DateTime).
    - **Bổ sung:**
        - Status (Enum: Pending / Confirmed / Cancelled) - Trạng thái booking.
        - Notes (string, nullable) - Ghi chú.
        - CreatedDate.
- `[xxx]_Challenges` (Sự kiện/Giải đấu):
    - Id, Title (string), Type (Enum: Duel / MiniGame).
    - GameMode (Enum: None / TeamBattle / RoundRobin) - None cho Duel, TeamBattle/RoundRobin cho MiniGame.
    - Status (Enum: Open / Ongoing / Finished) - Open: Đang mở đăng ký, Ongoing: Đang diễn ra, Finished: Đã kết thúc.
    - Config_TargetWins (int, nullable) - Cho TeamBattle: Số trận thắng cần đạt (VD: 5).
    - CurrentScore_TeamA (int), CurrentScore_TeamB (int) - Điểm hiện tại của 2 phe (cho TeamBattle).
    - **Bổ sung (bắt buộc):**
        - EntryFee (decimal) - Số tiền mỗi người đóng để tham gia.
        - PrizePool (decimal) - Tổng quỹ thưởng (có thể tự động = EntryFee × số participants hoặc admin nhập).
        - CreatedBy (FK → Members) - Người tạo challenge.
        - StartDate, EndDate (DateTime?, nullable) - Thời gian bắt đầu/kết thúc.
        - CreatedDate, ModifiedDate.
- `[xxx]_Matches` (Bảng lưu trữ trận đấu – Thiết kế chặt chẽ):
    - Id, Date (DateTime) - Ngày giờ trận đấu.
    - IsRanked (bool) - **Quan trọng:** Nếu = true thì trận này sẽ ảnh hưởng đến Rank DUPR. Thường thì Daily Matches và Challenge Matches đều có thể ranked.
    - ChallengeId (FK → Challenges, nullable) - Null nếu là trận giao hữu, có giá trị nếu thuộc Challenge.
    - MatchFormat: Enum (Singles = 1vs1, Doubles = 2vs2).
    - **ĐỘI HÌNH 1 (TEAM 1):**
        - Team1_Player1Id (FK → Members, bắt buộc).

        - Team1_Player2Id (FK → Members, nullable) - Chỉ có dữ liệu khi MatchFormat = Doubles.
    - **ĐỘI HÌNH 2 (TEAM 2):**
        - Team2_Player1Id (FK → Members, bắt buộc).
        - Team2_Player2Id (FK → Members, nullable) - Chỉ có dữ liệu khi MatchFormat = Doubles.
    - **KẾT QUẢ:**
        - WinningSide: Enum (None - Chưa đấu, Team1, Team2).

- `[xxx]_Participants`:
    - Id, ChallengeId (FK → Challenges), MemberId (FK → Members).
    - **Team (Enum: TeamA / TeamB / None)** - Cho TeamBattle: thuộc phe nào. None cho RoundRobin hoặc Duel.
    - **EntryFeePaid (bool)** - Đã đóng Entry Fee chưa.
    - **EntryFeeAmount (decimal)** - Số tiền đã đóng (thường = Challenge.EntryFee, nhưng có thể khác nếu có giảm giá/ưu đãi).
    - **JoinedDate (DateTime)** - Ngày tham gia.
    - **Status (Enum: Pending / Confirmed / Withdrawn)** - Pending: Chờ xác nhận, Confirmed: Đã xác nhận, Withdrawn: Đã rút.

### 4. Bảng bổ sung (Khuyến nghị / Điểm cộng)

- `[xxx]_MatchScores`:
    - Id, MatchId (FK → Matches), SetNumber (int), Team1Score (int), Team2Score (int), IsFinalSet (bool).
    - Dùng để lưu chi tiết điểm từng set (phục vụ DUPR nâng cao).

- `[xxx]_Notifications`:
    - Id, MemberId (FK → Members, nullable) - null = thông báo chung cho tất cả, Title, Content, Type (Enum: Info/Warning/Success/Error), IsRead (bool), CreatedDate, LinkUrl (string, nullable).

- `[xxx]_ActivityLogs`:
    - Id, UserId (FK → AspNetUsers), Action (string), EntityType (string), EntityId (int?, nullable), Description (string), IpAddress (string, nullable), CreatedDate.

---

## PHẦN 3: YÊU CẦU API & FRONTEND

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

**Yêu cầu bắt buộc:**

- **Authentication & Authorization:**
    - Sử dụng JWT (JSON Web Token) cho authentication.
    - API endpoints phải có authorization dựa trên Role (Admin, Treasurer, Referee, Member).
    - Endpoint đăng nhập: `POST /api/auth/login` - Trả về JWT token.
    - Endpoint đăng ký: `POST /api/auth/register` (nếu cần).
    - Endpoint lấy thông tin user hiện tại: `GET /api/auth/me`.

- **API Endpoints cần có (tối thiểu):**
  - **Members:** `GET /api/members`, `GET /api/members/{id}`, `PUT /api/members/{id}`, `GET /api/members/top-ranking`
  - **News:** `GET /api/news`, `GET /api/news/{id}`, `POST /api/news`, `PUT /api/news/{id}`, `DELETE /api/news/{id}`
  - **Courts:** `GET /api/courts`, `POST /api/courts`, `PUT /api/courts/{id}`, `DELETE /api/courts/{id}`
  - **Bookings:** `GET /api/bookings`, `POST /api/bookings`, `PUT /api/bookings/{id}`, `DELETE /api/bookings/{id}`, `GET /api/bookings/available-slots`
  - **Challenges:** `GET /api/challenges`, `GET /api/challenges/{id}`, `POST /api/challenges`, `PUT /api/challenges/{id}`, `POST /api/challenges/{id}/join`, `POST /api/challenges/{id}/auto-divide-teams`
  - **Matches:** `GET /api/matches`, `GET /api/matches/{id}`, `POST /api/matches`, `PUT /api/matches/{id}`
  - **Transactions:** `GET /api/transactions`, `POST /api/transactions`, `GET /api/transactions/summary`
  - **TransactionCategories:** `GET /api/transaction-categories`, `POST /api/transaction-categories`, `PUT /api/transaction-categories/{id}`, `DELETE /api/transaction-categories/{id}`
- **Response Format:**
  - Sử dụng chuẩn RESTful API.
  - Response JSON với format nhất quán (có thể dùng wrapper class hoặc theo chuẩn).
  - Xử lý lỗi trả về status code phù hợp (400, 401, 403, 404, 500) và message rõ ràng.
- **CORS Configuration:**
  - Cấu hình CORS để cho phép Vue.js frontend gọi API (có thể cấu hình cho localhost:5173 hoặc port Vue dev server).
- **Swagger/OpenAPI:**
  - **Khuyến nghị:** Cấu hình Swagger UI để document API (truy cập `/swagger`).

## 2. Frontend (Vue.js)

**Yêu cầu bắt buộc:**

- **Cấu trúc Project:**
  - Sử dụng Vue 3 với Composition API (khuyến nghị) hoặc Options API.
  - Sử dụng Vue Router cho routing.
  - Sử dụng Pinia hoặc Vuex cho state management (khuyến nghị Pinia).
  - Sử dụng Axios hoặc Fetch API để gọi backend API.
- **Authentication:**
  - Lưu JWT token trong localStorage hoặc sessionStorage.
  - Tự động thêm token vào header `Authorization: Bearer {token}` cho mọi API request.

    - Xử lý token hết hạn: Redirect về trang login khi nhận 401 Unauthorized.
    - Tạo interceptor (Axios) để tự động refresh token nếu cần.

- **Các trang/component cần có:**

    - **Login/Register:** Form đăng nhập, gọi API `/api/auth/login`.
    - **Dashboard:** Hiển thị thống kê, tin tức ghim, cảnh báo quỹ âm (nếu có quyền).
    - **Members:** Danh sách members, chi tiết member, chỉnh sửa (chỉ member của chính mình).
    - **News:** Danh sách news, chi tiết, CRUD (nếu có quyền).
    - **Courts:** Danh sách courts, CRUD (Admin only).
    - **Bookings:** Lịch đặt sân, form đặt sân, danh sách booking của mình.
    - **Challenges:** Danh sách challenges, chi tiết challenge, tạo challenge, tham gia challenge.
    - **Matches:** Danh sách matches, form tạo/nhập kết quả match (Referee/Admin only).
    - **Transactions:** Danh sách transactions, form tạo transaction, báo cáo tài chính (Admin/Treasurer only).
    - **TransactionCategories:** CRUD categories (Admin only).

- **UI/UX:**

    - Sử dụng CSS framework (Bootstrap, Vuetify, Tailwind CSS, hoặc tự thiết kế).
    - Giao diện responsive (mobile-friendly).
    - Loading states khi gọi API (spinner, skeleton).
    - Toast notifications cho success/error (có thể dùng thư viện như vue-toastification).
    - Form validation (có thể dùng VeeValidate hoặc tự validate).

- **State Management:**

    - Lưu thông tin user hiện tại (sau khi login).
    - Lưu token, roles.
    - Cache dữ liệu thường dùng (members, courts, categories).

## 3. Dashboard & News

- Frontend gọi API `GET /api/news?isPinned=true` để lấy tin ghim.
- Frontend gọi API `GET /api/transactions/summary` để lấy số dư quỹ, hiển thị cảnh báo nếu < 0 (chỉ Admin/Treasurer).
- **Khuyến nghị:** Gọi API `GET /api/members/top-ranking?limit=5` để hiển thị BXH Top 5.

## 4. Quản lý Tài chính Động (Dynamic Treasury)

- Frontend gọi API CRUD TransactionCategories và Transactions.
- API `GET /api/transactions/summary` trả về: `{ totalIncome, totalExpense, balance }`.
- Khi Challenge kết thúc: **Khuyến nghị** Backend tự động tạo Transaction ghi nhận chi trả thưởng (trong service layer).

## 5. Quản lý Sân & Đặt sân (Booking)

- **CRUD Courts:** Frontend gọi API CRUD Courts (chỉ Admin).

- **Đặt sân:**

    - Frontend gọi API `GET /api/bookings/available-slots?courtId={id}&date={date}` để lấy slot trống.
    - Frontend gọi API `POST /api/bookings` với body: `{ courtId, startTime, endTime, notes }`.
    - Backend validation: Kiểm tra trùng lịch, StartTime ≥ hiện tại, EndTime > StartTime.

- **Khuyến nghị:**
    - API trả về danh sách slot available theo ngày/tuần.
    - Frontend hiển thị Calendar view với màu phân biệt.

## 6. Module Trọng tài & Ghi nhận Trận đấu (Referee)

**Giao diện Frontend (Vue Component):**

- **Bước 1: Chọn Thể thức (Format)**
    - ☐ Đánh Đôi (2vs2).
    - Sử dụng `v-model` để bind giá trị, `v-if`/`v-show` để hiển thị/ẩn các field Player 2.
- **Bước 2: Chọn Người chơi**
    - Dropdown (select) load danh sách members từ API `GET /api/members`.
    - Validation: Không được chọn cùng một người hai lần (check trong `computed` hoặc `watch`).
- **Bước 3: Chọn Kết quả & Cấu hình**
    - Radio: Team 1 Thắng / Team 2 Thắng.
    - Checkbox: IsRanked.
    - Dropdown: Chọn Challenge (load từ API `GET /api/challenges?status=Open,Ongoing`).
- **Submit:** Gọi API `POST /api/matches` với body chứa đầy đủ thông tin.

**Logic Backend (bắt buộc):**

- **Nếu IsRanked = true:**
    - Cập nhật Rank cho người thắng và người thua.
    - Có thể dùng công thức đơn giản: Thắng +0.1, Thua -0.1.
    - **Khuyến nghị:** Công thức kiểu Elo/DUPR.
- **Nếu trận thuộc Challenge (ChallengeId != null):**
    - Nếu GameMode = TeamBattle: Cập nhật CurrentScore_TeamA hoặc CurrentScore_TeamB (+1).
        - Khi một team đạt Config_TargetWins → đánh dấu Challenge.Status = Finished.
    - Nếu GameMode = RoundRobin: Tích điểm cá nhân.
    - **Khuyến nghị:** Khi Challenge Finished: Tự động phân phối thưởng, lock Match, tạo notifications.

## 7. Sàn đấu Kèo (Challenge Board)

- Frontend gọi API `GET /api/challenges` để lấy danh sách.
- Frontend gọi API `POST /api/challenges` để tạo challenge mới.

- Frontend gọi API `POST /api/challenges/{id}/join` để tham gia challenge.
- **Khuyến nghị:** Frontend gọi API `POST /api/challenges/{id}/auto-divide-teams` để phân chia Team A/B tự động (nếu TeamBattle).

### 8. Phân quyền & Bảo mật

- **Backend:**
  - Sử dụng `[Authorize(Roles = "Admin")]` trên các Controller/Action.
  - Kiểm tra quyền trong service layer (VD: Member chỉ sửa được thông tin của chính mình).
- **Frontend:**
  - Sử dụng Vue Router guards (`beforeEach`) để kiểm tra authentication.
  - Ẩn/hiện UI elements dựa trên role (VD: Chỉ Admin mới thấy nút "Quản lý Courts").
  - Redirect về login nếu chưa đăng nhập hoặc không có quyền.

---

## PHẦN 4: YÊU CẦU KỸ THUẬT (MIGRATION & SEEDING)

**Yêu cầu bắt buộc:** Sinh viên thực hiện Data Seeding (trong `OnModelCreating` hoặc `DbInitializer`) để khi chạy `Update-Database`, hệ thống có sẵn dữ liệu sau:

1. **Identity:**
   - 1 Admin (email: admin@pcm.com, password: tùy chọn, VD: "Admin@123").
   - 6–8 Member mẫu (tạo user Identity tương ứng).
   - **Members** phải có **UserId** liên kết với user Identity (lấy UserId từ AspNetUsers sau khi tạo user).
2. **Courts:**
   - Ít nhất 2 sân mẫu (VD: "Sân 1", "Sân 2", IsActive = true).
3. **Tài chính:**
   - Categories mẫu: "Tiền sân" (Thu), "Quỹ tháng" (Thu), "Nước" (Chi), "Phạt" (Chi).
   - Transactions mẫu: Tạo vài giao dịch Thu và Chi sao cho **Quỹ đang dương** (tổng Thu > tổng Chi).
   - Ví dụ: Thu 1.000.000đ (Quỹ tháng), Chi 200.000đ (Nước) → Số dư = 800.000đ.
4. **Hoạt động:**
   - 1 Kèo Mini-game (Team Battle) **Status = Ongoing**, có:
     - EntryFee (VD: 50.000đ).
     - PrizePool (VD: 500.000đ = 10 người × 50.000đ).
     - **Participants:** 10-12 người, chia đều Team A / Team B (Team = TeamA hoặc TeamB), EntryFeePaid = true, EntryFeeAmount = EntryFee, Status = Confirmed.
   - Lịch sử **Matches:** 2–3 trận đã diễn ra:
     - 1-2 trận Đơn (Singles) với IsRanked = true, có kết quả (WinningSide = Team1 hoặc Team2).
     - 1 trận Đôi (Doubles) với IsRanked = true, có kết quả.
     - Có thể gắn với Challenge (ChallengeId) hoặc để null (giao hữu).

---

# PHẦN 5: YÊU CẦU GIAO DIỆN & TRẢI NGHIỆM (UI/UX)

- **Frontend:** Sử dụng **Vue.js 3** với Composition API (khuyến nghị) hoặc Options API.
- **UI Framework:** Sử dụng CSS framework (Bootstrap, Vuetify, Tailwind CSS, hoặc tự thiết kế).
- **Dashboard:**
  - Hiển thị các chỉ số quan trọng: Số dư quỹ, Số kèo đang mở, BXH Top 5 (gọi API để lấy dữ liệu).
  - **Khuyến nghị:** Widget Top 5 Ranking; thống kê nhanh (số trận hôm nay, booking sắp tới).
- **Sàn đấu:**
  - Thiết kế dạng thẻ (Card), làm nổi bật **PrizePool**, **Entry Fee** và phần thưởng Mini-game.
  - Hiển thị rõ Status (Open/Ongoing/Finished), danh sách participants.
- **Khuyến nghị:**
  - Giao diện responsive (mobile-friendly).
  - Pagination cho danh sách dài (có thể dùng API pagination: `?page=1&pageSize=20`).
  - Loading spinner/skeleton khi gọi API.
  - Toast notifications cho success/error (vue-toastification hoặc tương tự).

---

# PHẦN 6: ĐIỂM CỘNG (BONUS FEATURES)

1. **Báo cáo tài chính (Export):**
   - Backend API endpoint `GET /api/transactions/export?format=excel|pdf` trả về file.
   - Frontend download file hoặc hiển thị preview.
2. **Top Ranking:**
   - Widget Top 5 cao thủ (Rank cao nhất) trên Dashboard, có avatar, trend up/down.
3. **Validate Tuổi:**
   - Frontend tính tuổi từ DateOfBirth, hiện cảnh báo nếu < 10 hoặc > 80: "Cẩn thận chấn thương nhé!".
4. **DUPR nâng cao:**
   - Backend tính Rank theo công thức kiểu Elo/DUPR thay vì +0.1/-0.1 đơn giản.
5. **Calendar view:**
   - Frontend hiển thị lịch đặt sân theo tuần/tháng (có thể dùng thư viện như FullCalendar, Vue Calendar), màu phân biệt đã đặt / còn trống / của mình.
6. **Notifications / ActivityLogs:**
   - Backend API: `GET /api/notifications`, `GET /api/activity-logs`.
   - Frontend hiển thị notifications với badge số lượng chưa đọc, real-time update (có thể dùng SignalR hoặc polling).
7. **Tìm kiếm & lọc:**
   - Frontend có search box và filters, gọi API với query params: `GET /api/members?search=keyword&rankMin=4.0`.

- Backend hỗ trợ filtering, searching, sorting.

8. **Real-time updates:**
   - Sử dụng SignalR để real-time notifications, cập nhật Challenge scores, v.v.

---

# PHẦN 7: QUY ĐỊNH NỘP BÀI

**Cách thức nộp bài:**

1. **Bước 1:** Đẩy toàn bộ Source Code lên **GitHub** (Public).
   - **Backend:** Đảm bảo `.gitignore` loại trừ `bin`, `obj`, `appsettings.json` (nếu có thông tin nhạy cảm).
   - **Frontend:** Đảm bảo `.gitignore` loại trừ `node_modules`, `.env.local` (nếu có).
   - Có thể tách 2 repository riêng (backend và frontend) hoặc monorepo (1 repository với 2 folders).
2. **Bước 2:** Copy đường link Repository (hoặc 2 links nếu tách riêng).
3. **Bước 3:** Gửi link vào nhóm Zalo môn học với cú pháp:

   ```
   NỘP BÀI THI: [MSSV] - [HỌ VÀ TÊN] - [LINK GITHUB BACKEND] - [LINK GITHUB FRONTEND]
   ```

   Hoặc nếu monorepo:

   ```
   NỘP BÀI THI: [MSSV] - [HỌ VÀ TÊN] - [LINK GITHUB]
   ```

   Ví dụ: `NỘP BÀI THI: 123456 - Nguyễn Văn A - https://github.com/nguyenvana/pcm-api - https://github.com/nguyenvana/pcm-frontend`

**Lưu ý:**

- Đảm bảo repository có README.md mô tả cách chạy project (cả backend và frontend).
- Backend: Đảm bảo có file migration hoặc script tạo database.
- Frontend: Đảm bảo có file `package.json` và hướng dẫn cài đặt dependencies (`npm install`).
- Giảng viên sẽ:
  - Clone repo backend, chạy `Update-Database` để kiểm tra seeding data.
  - Chạy backend API (VD: `dotnet run` hoặc F5 trong Visual Studio).
  - Clone repo frontend, chạy `npm install` và `npm run dev`.
  - Kiểm tra frontend kết nối với backend API.

Chúc các em làm bài tốt!