# Max Potential Basketball (MPB) System: Complete Architecture & Implementation Plan

## 1. Introduction & Vision

This document outlines a comprehensive architecture and implementation plan for the Max Potential Basketball (MPB) system. The goal is to evolve the foundational concepts from the `mp-basketball` repository into a robust, efficient, and scalable application, `mpb-web-v2`, built on Next.js and Supabase.

The core of this system is the **Development ARC** philosophy:

- **Advancement:** Continuous, measurable progress for individuals and the team.
- **Responsibilities:** Clear roles, ownership, and accountability at all levels.
- **Collective Growth:** Synergistic development where individual improvement contributes to team success and vice-versa.

This plan addresses the 10 critical functional components identified, ensuring they are deeply integrated with the ARC philosophy.

**The 10 Critical Components:**

1. Modular Drill/Action Bank
2. Practice Plan Generator
3. Attendance/Availability Engine
4. PDP (Personal Development Plan) Overlay Engine
5. Constraint Library
6. Review/Approval Workflow
7. Analytics & Feedback Loop
8. Reflection and Media Capture
9. Audit, Traceability, and Permissions
10. UI/UX Principles (Modular, Fast, ARC-Contextualized)

# 2. Core Philosophical Framework: The Development ARC

The Development ARC is not just a feature but the guiding philosophy for the entire system. It influences how data is structured, how plans are generated, and how progress is tracked. The CSV files (`mpbc_pillar_rows.csv`, `mpbc_phase_rows.csv`, `mpbc_practice_themes_rows.csv`, `mpbc_outcome_rows.csv`) provide the vocabulary for this framework.

**ARC Components & Data Mapping:**

- **Pillars (`mpbc_pillar_rows.csv`):** High-level strategic areas of focus (e.g., "HURDLE", "SPRINT", "DISRUPT"). These represent overarching team philosophies.
- **Phases (`mpbc_phase_rows.csv`):** Specific stages of the game or development linked to Pillars (e.g., "Transition Offense" under "SPRINT"). Each phase has an intent, KPI, and description.
- **Practice Themes (`mpbc_practice_themes_rows.csv`):** Concrete themes for individual practice sessions, linking back to broader ARC components (e.g., "Advantage Creation", "Pressure / Disruption").
- **Outcomes (`mpbc_outcome_rows.csv`):** Desired results or behaviors within a Phase, used to measure success and guide drill selection (e.g., "Score before defense sets" for "Transition Offense" phase).

**Proposed ARC-Related Database Tables:**

- `arc_pillar`: Stores pillar definitions.
  - `id` (UUID, PK)
  - `name` (TEXT, e.g., "HURDLE")
  - `description` (TEXT)
- `arc_phase`: Stores phase definitions, linked to pillars.
  - `id` (UUID, PK)
  - `pillar_id` (UUID, FK to `arc_pillar`)
  - `name` (TEXT, e.g., "Transition Offense")
  - `intent` (TEXT)

    - `kpi` (TEXT)
    - `description` (TEXT)
- `arc_theme`: Stores practice theme definitions.
    - `id` (UUID, PK)
    - `name` (TEXT, e.g., "Advantage Creation")
    - `slug` (TEXT, unique)
    - `category` (TEXT, e.g., "Offense", "Defense", "Universal")
    - `description` (TEXT)
    - `color_hex` (TEXT, optional for UI)
- `arc_outcome`: Stores outcome definitions, linked to phases.
    - `id` (UUID, PK)
    - `phase_id` (UUID, FK to `arc_phase`)
    - `description` (TEXT, e.g., "Ball crosses half court in four seconds or less")

These tables will allow the system to tag sessions, drills, and player development plans with ARC components, making the philosophy actionable and measurable.

# 3. System Architecture

## 3.1 Overall Architecture

The MPB system will adopt a modern, scalable architecture:

- **Frontend:** Next.js 15 (`mpb-web-v2`) utilizing the App Router for server and client components, providing a rich, interactive user experience.
- **Backend:** Supabase, providing:
    - **Database:** PostgreSQL for structured data storage, leveraging RLS for security.
    - **Authentication:** Supabase Auth for managing user identities and roles.
    - **Edge Functions:** TypeScript-based serverless functions for business logic (e.g., plan generation, PDP overlays).

    - **Realtime:** For live updates (e.g., attendance changes, collaborative plan editing).
    - **Storage:** For media uploads (videos, images, PDFs related to reflections and drills).
- **Monorepo Structure (Recommended):** Using pnpm workspaces or Turborepo to manage shared packages (UI components, DB types, utils).

```
[Next.js Frontend (apps/web)]
       |
       |--- API Calls (Authenticated) ---> [Supabase Edge
Functions (supabase/functions)]
       |                                          |
       |--- Realtime Subscriptions -------------| |-->
[Supabase PostgreSQL DB (supabase/migrations)]
       |                                          |
(RLS, DB Functions, Triggers)
       |--- Direct DB Queries (RLS-protected) ---|
       |
       |--- Media Uploads/Downloads ----------> [Supabase
Storage (supabase/storage)]
```

## 3.2 Database Schema (Key Tables)

This schema integrates the 10 critical components and the ARC philosophy, drawing from the provided CSVs.

- **User & Team Management:**
    - `person`:
        - `id` (UUID, PK)
        - `auth_user_id` (UUID, FK to `auth.users` ON DELETE CASCADE, UNIQUE)
        - `full_name` (TEXT)
        - `email` (TEXT, UNIQUE)
        - `role` (ENUM: 'coach', 'player', 'admin', 'parent') - *Responsibility*
        - `created_at`, `updated_at`
    - `team`:

        - `id` (UUID, PK)
        - `name` (TEXT)
        - `current_arc_pillar_id` (UUID, FK to `arc_pillar`, nullable) - *Collective Growth*
        - `current_arc_phase_id` (UUID, FK to `arc_phase`, nullable)
        - `created_at`, `updated_at`
    - `person_team_membership`:
        - `person_id` (UUID, FK to `person`)
        - `team_id` (UUID, FK to `team`)
        - PRIMARY KEY (`person_id`, `team_id`)
    - `pod` (Optional small groups within a team):
        - `id` (UUID, PK)
        - `team_id` (UUID, FK to `team`)
        - `name` (TEXT)
        - `created_at`, `updated_at`
    - `person_pod_membership`:
        - `person_id` (UUID, FK to `person`)
        - `pod_id` (UUID, FK to `pod`)
        - PRIMARY KEY (`person_id`, `pod_id`)
- **ARC Framework Tables:** (As defined in Section 2)
    - `arc_pillar`
    - `arc_phase`
    - `arc_theme`
    - `arc_outcome`

- **Drill & Constraint Bank (#1, #5):**
  - `skill_tag`: (Based on `mpbc_skill_tag_rows (1).csv`)
    - `id` (UUID, PK)
    - `name` (TEXT, UNIQUE)
    - `description` (TEXT, nullable)
    - `category` (TEXT, nullable)
    - `subcategory` (TEXT, nullable)
    - `parent_skill_tag_id` (UUID, FK to `skill_tag`, nullable)
    - `synonyms` (JSONB, nullable)
    - `is_active` (BOOLEAN, default true)
    - `created_at`, `updated_at`
  - `constraint_definition`: (Based on `constraint_rows (1).csv`)
    - `id` (UUID, PK)
    - `name` (TEXT, UNIQUE)
    - `description` (TEXT, nullable)
    - `category` (TEXT, e.g., "Offense", "Defense", "Session/Drill", "Compete")
    - `tag_type` (TEXT, default 'constraint')
    - `use_case` (TEXT, nullable)
    - `is_active` (BOOLEAN, default true)
    - `created_at`, `updated_at`
  - `drill_template`:
    - `id` (UUID, PK)

        - `name` (TEXT)
        - `intent` (TEXT) - *What's being worked on*
        - `description` (TEXT, nullable)
        - `supported_formats` (JSONB, e.g., `["1v1", "2v2", "3v3+"]`)
        - `default_duration_minutes` (INTEGER, nullable)
        - `setup_instructions` (TEXT, nullable)
        - `created_by_person_id` (UUID, FK to `person`)
        - `created_at`, `updated_at`
    - `drill_template_skill_tag`: (Many-to-many)
        - `drill_template_id` (UUID, FK to `drill_template`)
        - `skill_tag_id` (UUID, FK to `skill_tag`)
        - PRIMARY KEY (`drill_template_id`, `skill_tag_id`)
    - `drill_template_constraint_definition`: (Many-to-many for suggested/compatible constraints)
        - `drill_template_id` (UUID, FK to `drill_template`)
        - `constraint_definition_id` (UUID, FK to `constraint_definition`)
        - PRIMARY KEY (`drill_template_id`, `constraint_definition_id`)
- **Practice Planning & Execution (#2, #3, #6):**
    - `session`: (Practice Plan Header)
        - `id` (UUID, PK)
        - `team_id` (UUID, FK to `team`)
        - `session_date` (DATE)

        - `start_time` (TIME, nullable)
        - `end_time` (TIME, nullable)
        - `location` (TEXT, nullable)
        - `arc_theme_id` (UUID, FK to `arc_theme`, nullable) - *ARC Context*
        - `status` (ENUM: 'draft', 'finalized', 'active', 'completed', 'archived') - *Review/Approval*
        - `notes_coach` (TEXT, nullable)
        - `created_by_person_id` (UUID, FK to `person`)
        - `finalized_at` (TIMESTAMP, nullable)
        - `created_at`, `updated_at`
    - `attendance`: (#3)
        - `id` (UUID, PK)
        - `session_id` (UUID, FK to `session`)
        - `person_id` (UUID, FK to `person`)
        - `status` (ENUM: 'present', 'absent', 'excused', 'late')
        - `notes` (TEXT, nullable)
        - `check_in_time` (TIMESTAMP, nullable)
        - `created_at`, `updated_at`
        - UNIQUE (`session_id`, `person_id`)
    - `session_block`: (Individual block in a practice plan)
        - `id` (UUID, PK)
        - `session_id` (UUID, FK to `session`)
        - `drill_template_id` (UUID, FK to `drill_template`, nullable, if using a template)

        - `custom_drill_name` (TEXT, nullable, if not using a template)
        - `custom_drill_intent` (TEXT, nullable)
        - `block_order` (INTEGER) - *Sequence*
        - `duration_minutes` (INTEGER)
        - `notes_coach` (TEXT, nullable)
        - `applied_constraints` (JSONB, array of `constraint_definition_id` or custom constraint objects) - *Constraint Overlays*
        - `pdp_overlays` (JSONB, array of objects: `{person_id, pdp_focus_description, skill_tag_id?, constraint_id?}`) - *PDP Overlay Engine (#4)*
        - `created_at`, `updated_at`
- **Player Development (PDP) (#4):**
    - `pdp`: (Personal Development Plan) - *Advancement*
        - `id` (UUID, PK)
        - `person_id` (UUID, FK to `person`) - *Player*
        - `team_id` (UUID, FK to `team`, for context at time of creation)
        - `created_by_person_id` (UUID, FK to `person`) - *Coach/Admin*
        - `start_date` (DATE)
        - `end_date` (DATE, nullable)
        - `overall_focus` (TEXT, nullable)
        - `status` (ENUM: 'active', 'archived', 'completed')
        - `created_at`, `updated_at`
    - `pdp_item`:
        - `id` (UUID, PK)

        - `pdp_id` (UUID, FK to `pdp`)
        - `item_type` (ENUM: 'skill_focus', 'constraint_focus', 'habit', 'goal')
        - `skill_tag_id` (UUID, FK to `skill_tag`, nullable)
        - `constraint_definition_id` (UUID, FK to `constraint_definition`, nullable)
        - `description` (TEXT) - *Specific focus area*
        - `priority` (INTEGER, nullable)
        - `notes` (TEXT, nullable)
        - `target_metric` (TEXT, nullable)
        - `current_progress` (TEXT, nullable)
        - `created_at`, `updated_at`
- **Reflection & Media (#8):**
    - `reflection`:
        - `id` (UUID, PK)
        - `session_id` (UUID, FK to `session`, nullable)
        - `session_block_id` (UUID, FK to `session_block`, nullable)
        - `person_id` (UUID, FK to `person`) - *Author (coach or player)*
        - `reflection_text` (TEXT)
        - `mood_rating` (INTEGER, nullable, e.g., 1-5)
        - `effort_rating` (INTEGER, nullable, e.g., 1-5)
        - `created_at`, `updated_at`
    - `media_attachment`:
        - `id` (UUID, PK)
        - `reflection_id` (UUID, FK to `reflection`, nullable)

        - `session_id` (UUID, FK to `session`, nullable)
        - `session_block_id` (UUID, FK to `session_block`, nullable)
        - `drill_template_id` (UUID, FK to `drill_template`, nullable)
        - `person_id` (UUID, FK to `person`, nullable, e.g. player in video)
        - `uploader_person_id` (UUID, FK to `person`)
        - `storage_path` (TEXT) - *Path in Supabase Storage*
        - `file_name` (TEXT)
        - `mime_type` (TEXT)
        - `description` (TEXT, nullable)
        - `created_at`, `updated_at`
- **Audit & System (#7, #9):**
    - `audit_log`:
        - `id` (BIGSERIAL, PK)
        - `timestamp` (TIMESTAMP WITH TIME ZONE, default NOW())
        - `actor_person_id` (UUID, FK to `person`, nullable)
        - `action` (TEXT, e.g., 'session.create', 'pdp_item.update')
        - `target_table` (TEXT, nullable)
        - `target_record_id` (TEXT, nullable)
        - `details_before` (JSONB, nullable)
        - `details_after` (JSONB, nullable)
        - `ip_address` (INET, nullable)
    - `system_flagged_item`: (For analytics feedback loop - e.g., missed overlays)

        - `id` (UUID, PK)
        - `item_type` (TEXT, e.g., 'block_missing_pdp_overlay', 'player_stale_pdp')
        - `related_session_id` (UUID, FK to `session`, nullable)
        - `related_person_id` (UUID, FK to `person`, nullable)
        - `related_block_id` (UUID, FK to `session_block`, nullable)
        - `description` (TEXT)
        - `status` (ENUM: 'open', 'acknowledged', 'resolved')
        - `created_at`, `updated_at`

### 3.3 Application Layers

- **Frontend (Next.js `apps/web`):**
    - **UI Components (`packages/ui`):** Reusable React components for drills, blocks, player cards, ARC visualizations, forms, etc., styled with Tailwind CSS.
    - **State Management:** React Context, Zustand, or similar for global state (e.g., current user, team). React Query/SWR for server state and caching.
    - **API Interaction:** Typed API client (generated or custom hooks) for calling Supabase Edge Functions and direct DB operations.
    - **Realtime:** Supabase Realtime client for subscribing to changes in `attendance`, `session_block`, etc.
- **API Layer (Supabase Edge Functions `supabase/functions`):**
    - `generate-practice-plan`: Takes `session_id`, fetches attendance, team ARC focus, recent friction (from reflections/analytics), player PDPs, and available `drill_templates`. Returns a structured `session_block` array.
    - `apply-pdp-overlays`: Takes `session_block_id` and `person_id`, fetches PDP data, and suggests/applies overlays.
    - `log-reflection`: Handles creation of reflection entries and media uploads.

    - `crud-operations`: Secure endpoints for complex CUD operations if RLS is insufficient or business logic is needed.
    - All functions will be authenticated and perform necessary authorization checks.
- **Data Layer (Supabase Postgres `supabase/migrations`):**
    - **RLS Policies:** Granular access control for all tables based on `person.role` and memberships.
    - **DB Functions (SQL/PLpgSQL):** For complex queries, data aggregation, or operations best performed close to the data (e.g., calculating player readiness scores).
    - **Triggers:** For `audit_log` entries, updating `updated_at` timestamps.

# 4. Addressing the 10 Critical Components

1. **Modular Drill/Action Bank:**
    - **DB:** `drill_template`, `skill_tag`, `constraint_definition`, `drill_template_skill_tag`, `drill_template_constraint_definition`.
    - **UI:** Interface for coaches to create, browse, search, and tag drills. Visual representation of supported formats and constraints.
    - **ARC:** Drills can be tagged with `arc_theme`s or relevant `skill_tag`s that align with ARC pillars/phases.
2. **Practice Plan Generator:**
    - **Edge Function:** `generate-practice-plan`.
    - **Logic:** Considers `team.current_arc_pillar_id/phase_id`, `session.arc_theme_id`, `attendance` (for formats), `pdp` (player needs), `system_flagged_item` (recent friction), `drill_template` (availability).
    - **Output:** Ordered `session_block`s.
    - **ARC:** Generator prioritizes drills aligning with team's current ARC focus and session theme.
3. **Attendance/Availability Engine:**

- **DB:** `attendance` table.
- **UI:** Realtime interface for coaches to mark attendance for a session. Defaults to all rostered players.
- **Realtime:** Updates propagate to plan generator and UI.

4. **PDP Overlay Engine:**
   - **DB:** `pdp`, `pdp_item`, `session_block.pdp_overlays` (JSONB).
   - **Edge Function:** `apply-pdp-overlays` (can be part of plan generation or a separate call).
   - **Logic:** For each player present in a `session_block`, retrieves their active `pdp_item`s and suggests/annotates the block with relevant focus areas (e.g., "Player X: Focus on 'Weak Hand Dribble' during this drill").
   - **ARC:** Ensures individual *Advancement* within *Collective Growth* activities.
5. **Constraint Library:**
   - **DB:** `constraint_definition` table.
   - **UI:** Palette of available constraints that can be dragged/applied to `session_block`s.
   - **Logic:** `session_block.applied_constraints` stores selected constraints.
6. **Review/Approval Workflow:**
   - **DB:** `session.status` field. RLS policies to lock editing after 'finalized'.
   - **UI:** Coaches can edit 'draft' plans. Finalize button changes status. Admins might have override.
   - **Audit:** `audit_log` tracks all changes to `session` and `session_block`. - *Responsibility*
7. **Analytics & Feedback Loop:**
   - **DB:** Materialized views or DB functions for aggregating data (drill usage, PDP progress, theme effectiveness). `system_flagged_item` table for explicit feedback items.

- **UI:** Dashboards for coaches/admins showing trends, player progress against ARC, areas of friction.
- **ARC:** Analytics track progress towards ARC `kpi`s and `outcome`s. Feedback informs future ARC theme selection and PDP adjustments.

8. **Reflection and Media Capture:**
   - **DB:** `reflection`, `media_attachment`.
   - **UI:** Forms for coaches/players to log reflections per session/block. Upload interface for media, linking to Supabase Storage.
   - **ARC:** Reflections provide qualitative data on ARC effectiveness and player experience.
9. **Audit, Traceability, and Permissions:**
   - **DB:** `audit_log` table (auto-populated by triggers or logged by Edge Functions). RLS on all tables. `person.role` for permissions.
   - **ARC:** Ensures *Responsibility* and transparency.
10. **UI/UX Principles:**
    - **Modular:** Component-based UI (`packages/ui`). Block-based plan editor.
    - **Fast:** Next.js SSR/SSG, Edge Functions, optimized queries, Realtime updates.
    - **ARC-Contextualized:** UI elements display relevant ARC pillars, phases, themes. Player dashboards show progress within the ARC framework.

# 5. Implementation Strategy & Roadmap

A phased approach to build the system incrementally.

**Phase 0: Foundations & Core ARC Setup (2-3 Weeks)**

- **Goal:** Establish monorepo, Supabase project, core ARC tables, basic auth, and user/team management.
- **Tasks:**
  1. Setup pnpm/Turborepo workspace. Migrate `mpb-web-v2` to `apps/web`.
  2. Initialize Supabase project. Apply initial migrations for `person`, `team`, `arc_pillar`, `arc_phase`, `arc_theme`, `arc_outcome`.

3. Implement Supabase Auth (email/password, magic links). Basic login/signup pages.
4. CRUD UI for managing users (admin) and teams.
5. Seed ARC tables with data from CSVs.
6. Generate initial DB types (`packages/db`).

- **ARC Focus:** Establish the philosophical data backbone.

**Phase 1: Drill/Constraint Bank & PDP Foundation (3-4 Weeks)**

- **Goal:** Implement the modular drill and constraint libraries, and basic PDP structure.
- **Tasks:**
  1. Implement `skill_tag`, `constraint_definition`, `drill_template` tables and associated link tables.
  2. CRUD UI for managing skill tags, constraint definitions, and drill templates. Allow tagging drills with skills and suggested constraints.
  3. Implement `pdp` and `pdp_item` tables.
  4. Basic UI for coaches to create PDPs for players, linking to `skill_tag` and `constraint_definition` for focus areas.
- **ARC Focus:** Define the building blocks for *Advancement*.

**Phase 2: Attendance & Practice Plan Generation (MVP) (4-5 Weeks)**

- **Goal:** Implement attendance tracking and a first-pass practice plan generator.
- **Tasks:**
  1. Implement `session` and `attendance` tables.
  2. UI for creating sessions (date, team, select `arc_theme`).
  3. Realtime UI for marking attendance for a session.
  4. Develop `generate-practice-plan` Edge Function (MVP):
     - Input: `session_id`.

        - Logic: Fetch attendance, session theme. Select random/simple drills matching attendance count and theme.
        - Output: Create `session_block` entries for the session.
    5. Basic UI to display generated `session_block`s (read-only initially).
- **ARC Focus:** Link practice themes to ARC, begin *Collective Growth* planning.

**Phase 3: PDP Overlay, Plan Editing & Review Workflow (4-6 Weeks)**

- **Goal:** Enhance plan generator with PDP overlays, enable plan editing, and implement review/approval.
- **Tasks:**
    1. Enhance `generate-practice-plan` Edge Function to include PDP data:
        - Fetch active `pdp_item`s for present players.
        - Populate `session_block.pdp_overlays` with relevant PDP focus.
    2. UI to display PDP overlays on session blocks.
    3. Implement drag-and-drop UI for reordering `session_block`s. Allow editing duration, notes, applied constraints.
    4. Implement `session.status` workflow (draft -> finalized -> active -> completed).
    5. Implement RLS to enforce editing permissions based on status and role.
- **ARC Focus:** Deepen individual *Advancement* within team context, enforce *Responsibility* through review.

**Phase 4: Analytics, Reflection/Media & Audit (4-5 Weeks)**

- **Goal:** Implement core analytics, reflection logging, media uploads, and comprehensive auditing.
- **Tasks:**
    1. Implement `reflection` and `media_attachment` tables.
    2. UI for coaches/players to submit reflections (text) and upload media for sessions/blocks.

3. Implement `audit_log` table and triggers/Edge Function logic to populate it.
4. Develop initial analytics:
    - DB views/functions for drill usage, theme effectiveness, player attendance.
    - Basic dashboard to display these analytics.
5. Implement `system_flagged_item` table and logic to populate it (e.g., a nightly job to find blocks without PDP overlays for highly focused players).

- **ARC Focus:** Measure ARC effectiveness, provide feedback loops for *Advancement* and *Collective Growth*.

**Phase 5: UI Polish, Advanced Features & Scalability (Ongoing)**

- **Goal:** Refine UI/UX, add advanced features, optimize for performance and scale.
- **Tasks:**
    - Comprehensive UI/UX review and polish based on user feedback.
    - Parent dashboards/views.
    - Advanced analytics and reporting.
    - Notifications system (e.g., new PDP items, session finalized).
    - Performance optimization (query tuning, caching).
    - Mobile responsiveness and PWA considerations.
    - Further development of ARC insights and recommendations.

**Testing Strategy:**

- Unit tests for Edge Functions and critical UI components (Jest).
- Integration tests for API endpoints and database interactions.
- End-to-end tests for key user flows (Playwright/Cypress).
- Regular manual testing and user acceptance testing (UAT) at the end of each phase.

# 6. Estimated Timeline

- **Phase 0:** 2-3 Weeks

- **Phase 1:** 3-4 Weeks
- **Phase 2:** 4-5 Weeks
- **Phase 3:** 4-6 Weeks
- **Phase 4:** 4-5 Weeks
- **Total for Core MVP (Phases 0-4):** Approximately 5-6 months.
- **Phase 5:** Ongoing.

This timeline is an estimate and can be adjusted based on team size, experience, and specific priorities.

# 7. UI/UX Principles Revisited

The proposed architecture directly supports the desired UI/UX principles:

- **Modular:**
  - Database schema is normalized, allowing features to be built independently.
  - `drill_template` and `constraint_definition` are inherently modular.
  - `session_block`s are the core modular unit of a practice plan.
  - Component-driven frontend development.
- **Fast:**
  - Next.js provides optimized rendering.
  - Supabase Edge Functions are co-located with the database for low latency.
  - Supabase Realtime minimizes polling and provides instant updates.
  - Efficient PostgreSQL queries and indexing.
- **ARC-Contextualized:**
  - ARC framework tables (`arc_pillar`, `arc_phase`, `arc_theme`) allow tagging of all relevant entities.
  - UI dashboards will prominently display ARC context for teams and players.
  - Practice plan generation and PDPs are driven by ARC principles.

    - Analytics will measure progress and effectiveness in ARC terms.

- **Block-based Editing:** The `session_block` model is designed for easy drag-and-drop reordering, editing of individual blocks, and dynamic application of constraints and PDP overlays.
- **Clean Visual Separation:** The schema allows for distinct querying of team plans (`session`, `session_block`), player-specific overlays (`session_block.pdp_overlays`, `pdp_item`), and session metadata. The UI can then render these with clear visual distinctions.

# 8. Conclusion

This architecture and implementation plan provides a roadmap to build a powerful, efficient, and philosophically grounded Max Potential Basketball system. By leveraging Next.js and Supabase, and by keeping the Development ARC philosophy at the forefront, `mpb-web-v2` can become a transformative tool for basketball development. The phased approach allows for iterative development, feedback, and adaptation, ensuring the final product meets the needs of coaches, players, and organizations.

# Optimal Challenge Point (OCP) Integration into MPB System

## 1. Introduction

This document outlines the strategy for integrating the "Optimal Challenge Point" (OCP) concept into the Max Potential Basketball (MPB) system. The OCP principle, which emphasizes tailoring difficulty to maximize learning for both individual players and the team, will be deeply woven into the Development ARC (Advancement, Responsibilities, Collective Growth) framework.

**Core OCP Tenets:**

* **Player OCP:** The ideal difficulty for an individual, promoting growth without causing boredom (too easy) or frustration (too hard). Determined by their current Advancement (A) and Responsibility (R) levels.

* **Team OCP:** The ideal collective difficulty for the group, based on their current Collective Growth (C) phase.

* **Dynamic Adjustment:** OCP is not static; it evolves as players and teams develop. The system must support this dynamism.

This integration aims to make the MPB system not just a tracking tool but an active optimizer of player and team development by ensuring activities are consistently at the learning edge.

## 2. Database Schema Modifications & Additions

To support OCP, we'll modify existing tables and add new ones:

### 2.1. Modified Tables:

*   **`person` Table:**
    *   Add `current_advancement_level_id` (UUID, FK to a new `advancement_level_definition` table or a simple integer if levels are predefined and static).
    *   Add `current_responsibility_tier_id` (UUID, FK to a new `responsibility_tier_definition` table or a simple integer).
    *   *Purpose:* Store the player's current A & R status.

*   **`team` Table:**
    *   Add `current_collective_growth_phase_id` (UUID, FK to `arc_phase` or a new `collective_growth_phase_definition` table if more granularity is needed beyond `arc_phase`).
    *   *Purpose:* Store the team's current C status.

*   **`drill_template` Table:**
    *   Add `target_advancement_level_min` (INTEGER, e.g., A1-A10 scale).
    *   Add `target_advancement_level_max` (INTEGER, e.g., A1-A10 scale).
    *   Add `target_responsibility_tier_min` (INTEGER, e.g., R1-R6 scale).
    *   Add `target_responsibility_tier_max` (INTEGER, e.g., R1-R6 scale).

    *   Add `target_collective_growth_phase_id` (UUID, FK to `arc_phase` or `collective_growth_phase_definition`).

    *   *Purpose:* Define the intended OCP range for the drill. Allows drills to be suitable for a span of levels.

*   **`session_block` Table:**

    *   Add `coach_assessment_team_ocp` (ENUM: 'below_optimal', 'optimal', 'above_optimal', nullable).

    *   Add `coach_assessment_player_ocp` (JSONB, nullable, structure: `[{ "person_id": "UUID", "assessment": "below_optimal" | "optimal" | "above_optimal" }]`).

    *   *Purpose:* Allow coaches to log their perception of OCP for the team and individual players within a block.

### 2.2. New Tables:

*   **`advancement_level_definition` (Example - can be simplified if levels are just integers):**

    *   `id` (UUID, PK)

    *   `level_value` (INTEGER, e.g., 1, 2, ... 10)

    *   `level_name` (TEXT, e.g., "A1: Foundational Skills", "A10: Elite Scenario Mastery")

    *   `description` (TEXT)

    *   *Purpose:* Define the Advancement (A) scale. (Similar tables for `responsibility_tier_definition` and potentially `collective_growth_phase_definition` if more detail than `arc_phase` is needed).

*   **`challenge_point_log` Table:**

    *   `id` (UUID, PK, default `gen_random_uuid()`)

    *   `timestamp` (TIMESTAMP WITH TIME ZONE, default `NOW()`)

    *   `person_id` (UUID, FK to `person`, nullable) - *For player OCP log*

* `team_id` (UUID, FK to `team`, nullable) - *For team OCP log*

* `session_id` (UUID, FK to `session`, nullable)

* `session_block_id` (UUID, FK to `session_block`, nullable)

* `drill_template_id` (UUID, FK to `drill_template`, nullable)

* `logged_by_person_id` (UUID, FK to `person`) - *Coach who logged it*

* `entity_type` (ENUM: 'player', 'team')

* `assessed_challenge_level` (ENUM: 'too_easy', 'optimal', 'too_hard')

* `context_description` (TEXT, e.g., "Player X during 3v3 half-court drill with 'no dribble' constraint")

* `coach_notes_recommendation` (TEXT, nullable, e.g., "Increase defensive pressure for Player X next time.")

* *Purpose:* Track historical OCP assessments to inform future adjustments and player/team progression.

## 3. Enhanced Algorithms

### 3.1. Practice Plan Generator (`generate-practice-plan` Edge Function)

The generator's logic will be significantly enhanced:

1. **Input:** Session ID, which provides access to `team_id`, `session_date`, `arc_theme_id`, and present players via `attendance`.

2. **Fetch Current Levels:**

* Retrieve `team.current_collective_growth_phase_id`.

* For each present player, retrieve `person.current_advancement_level_id` and `person.current_responsibility_tier_id`.

3. **Filter `drill_template`s:**

    * Initial filter by `arc_theme_id` (if provided for the session) and other existing criteria (e.g., skill tags).

    * **Team OCP Filter:** Prioritize drills where `drill_template.target_collective_growth_phase_id` matches or is slightly above `team.current_collective_growth_phase_id`.

    * **Individual OCP Consideration (Weighted):**

        * For the selected drills, assess suitability for the median or key group of players based on their A & R levels against the drill's `target_A/R_level_min/max`.

        * The goal is to find drills that are generally appropriate for the team's OCP, with the understanding that individual OCPs will be fine-tuned via constraints and PDP overlays.

4. **Constraint Application for OCP:**

    * If a selected drill is slightly below the OCP for a significant portion of the team or key players, the generator can proactively suggest or apply `constraint_definition`s that are known to increase complexity (e.g., "time limit," "limited dribbles," "add defender").

5. **Output:**

    * The list of `session_block`s will now implicitly be more aligned with OCP.

    * The system can also output metadata for each block indicating its OCP alignment score for the team and potentially for key player archetypes.

### 3.2. PDP Overlay Engine

The PDP Overlay Engine's role in OCP:

1. **Context:** When a `session_block` is being planned or executed.

2. **Logic:**

    * For each player in the block, retrieve their active `pdp_item`s.

    * Compare the player's `current_advancement_level` with the `session_block`'s (derived from `drill_template`) `target_advancement_level`.

    * **If block is AT player's OCP:** The PDP overlay focuses on specific nuances within that challenge (e.g., "Player X: Focus on left-hand finishes during this finishing drill").

    *   **If block is BELOW player's OCP:** The PDP overlay might suggest a specific constraint *for that player* to increase the challenge (e.g., "Player Y: Attempt this passing drill with eyes up, using only non-verbal cues"). This personalized constraint is stored in `session_block.pdp_overlays`.

    *   **If block is ABOVE player's OCP:** The PDP overlay might suggest a simplification or a focus on a foundational element for that player to make the drill more accessible.

## 4. UI/UX Considerations

### 4.1. Planning Phase:

*   **Drill Library (`apps/web/src/app/drills`):**

    *   Display `target_A/R/C_level` for each `drill_template` using clear visual tags or icons.

    *   Allow filtering drills by these target levels.

*   **Player Profiles (`apps/web/src/app/players/[id]`):**

    *   Clearly display `current_advancement_level` and `current_responsibility_tier`.

    *   Show a history of OCP assessments from `challenge_point_log`.

*   **Team Profiles (`apps/web/src/app/teams/[id]`):**

    *   Display `current_collective_growth_phase`.

    *   Show team-level OCP history.

*   **Session Planning UI (`apps/web/src/app/sessions/[id]/plan`):**

    *   When adding/viewing a `session_block`:

        *   Visually indicate if the block is generally Below/At/Above the **team's OCP** (e.g., color-coding, icon).

        *   For each player rostered for the block, provide a subtle indicator of their **individual OCP alignment** with that block (e.g., a small dot: green for optimal, yellow for slightly off, red for significantly off).

        *   Allow coaches to manually override or confirm the system's OCP assessment for the block.

### 4.2. Live Session / Post-Session:

*   **Live Adaptation Tools (Practice Execution View):**
    *   Buttons like "Increase Challenge" / "Decrease Challenge" for the current `session_block`.
        *   Clicking these could open a modal suggesting relevant `constraint_definition`s (from the library) or rule modifications.
        *   Applying these would update `session_block.applied_constraints`.
*   **Quick OCP Logging:**
    *   Simple interface (e.g., thumbs up/down/sideways or a 3-point scale) for coaches to quickly log `coach_assessment_team_ocp` for the current block.
    *   Similar quick logging for `coach_assessment_player_ocp` for specific players observed during the block. This data populates `challenge_point_log`.
*   **Reflection Prompts:**
    *   Reflection prompts can be dynamically generated based on OCP assessments (e.g., "The team found block X too easy. What adjustments could be made next time?").

## 5. Analytics & Feedback Loop

The OCP data provides a rich source for analytics:

1. **OCP Alignment Reports:**
    *   Percentage of session time players/teams spend at, below, or above their OCP.
    *   Trends over time for individual players and the team.
2. **Correlation Analysis:**
    *   Correlate OCP alignment with `advancement_level` progression for players.
    *   Correlate team OCP alignment with `collective_growth_phase` progression.

* Identify if consistent "optimal" challenge leads to faster skill acquisition or team development.

3. **Drill Effectiveness from OCP Perspective:**

* Identify `drill_template`s that are frequently rated "too easy" or "too hard" for their tagged `target_A/R/C_level`. This can trigger a review/re-tagging of the drill.

4. **Automated System Refinement:**

* The `challenge_point_log` data becomes a feedback mechanism for the `Practice Plan Generator`.

* If the system consistently recommends drills that coaches mark as "too easy" for a player/team at a certain level, the system can learn to adjust its definition of "optimal" for that profile or suggest slightly harder drills.

5. **Identifying "Stuck" Points:**

* Flag players or teams that remain in the "too hard" zone for extended periods despite adjustments, indicating a need for more foundational work or a different approach.

* Flag players or teams that are too often in the "too easy" zone, indicating they are ready for more significant advancement.

## 6. Implementation Steps (Summary)

1. **Schema Update:** Implement DB changes outlined in Section 2. Create seed data for new definition tables.

2. **Core Data Entry:** Update UI for `person`, `team`, and `drill_template` to include and manage new OCP-related fields (current levels, target levels).

3. **Algorithm Enhancement:** Refactor `generate-practice-plan` and PDP overlay logic.

4. **UI Development (Planning):** Implement OCP indicators in drill library, player/team profiles, and session planning views.

5. **UI Development (Live/Post-Session):** Implement quick OCP logging and live adaptation suggestion tools.

6. **Logging:** Ensure `challenge_point_log` is populated from coach inputs.

7. **Analytics MVP:** Develop initial dashboards for OCP alignment and progression.

8. **Iterate:** Collect feedback and refine OCP tagging, generator logic, and UI based on usage.

By systematically integrating the Optimal Challenge Point concept, the MPB system will evolve into a highly adaptive and effective platform for fostering both individual talent and team synergy, truly embodying the Development ARC philosophy.