Below is a **Supabase + MemberPress** version of the integration map, ending with a concrete pattern for "parent → child" accounts that you can drop straight into a React component.

1 . High-level Architecture

Layer	Primary Service	Purpose
Payments &	**WordPress + MemberPress**	Takes the money, owns SKU logic, fires webhooks.
Auth & Data	**Supabase** (Postgres + Auth + Edge Functions +	Single source of truth for users, clubs, teams, workouts, points, parent/child links.
Front-end	**React (mobile-first)**	Pulls everything through the Supabase JS SDK; embeds WordPress pages or pulls WP REST for

⚡ WordPress only needs to *push* data to Supabase after a transaction. All normal reads/writes in-app go straight to Supabase, keeping the UI snappy.

2 . Supabase Schema (minimum viable tables)

```
public.users              -- Supabase Auth user_id (uuid PK), role (player/coach/director/parent)
public.clubs              -- id, name, tier, director_id (FK -> users)
public.teams              -- id, club_id, season, age_group, coach_id (FK)
public.players            -- id (FK -> users), team_id, jersey_number
public.parent_child       -- parent_id, child_id  (both FK → users)
public.workouts           -- id, title, skill_type, video_url, point_value
public.workout_logs       -- id, player_id, workout_id, completed_at
public.points_ledger      -- id, user_id, delta, reason, created_at
```

Enable Row-Level Security so each role sees only what it should.
Example RLS for `parent_child`:

```
-- parents can read/create links where they are the parent
create policy parent_owns_child on parent_child
```

```
  for all using ( auth.uid() = parent_id );
```

3 . Connecting MemberPress → Supabase

1. **Install MemberPress Developer Tools** (gives REST API + Webhooks).
2. Create an **Edge Function** in Supabase (e.g. `sync_membership`) that:
    - upserts the user (email → Supabase Auth with service key)
    - writes/updates `clubs.tier` **or** grants a `role` in `users`
3. In WordPress add a webhook on `mepr-event-transaction-completed`:

```
add_action('mepr-event-transaction-completed',
function($txn) {
  $member = $txn->user();
  $body = [
    'email'      => $member->user_email,
    'product_id' => $txn->product()->ID,
    'txn_id'     => $txn->id
  ];
  wp_remote_post('https://<project>.functions.supabase.co/
sync_membership', [
    'method'  => 'POST',
    'body'    => wp_json_encode($body),
    'headers' => ['Content-Type' => 'application/json',
                  'apikey'       =>
'<supabase_service_key>']
  ]);
});
```

4. The Edge Function looks up the SKU → tier map, inserts or updates the **club**, spins up default **teams**, and sets `users.role = 'director'`.
5. Send the director a Supabase **magic-link** email so they can finish setting a password inside React.

4 . Automating “Spawn Club OS / Team HQ / Skills Academy”

Trigger	Action (Edge Function)
Club OS purchase	`INSERT clubs`, loop tiers → `INSERT teams` with starter season data, grant Skills Academy quota (rows in `parent_child` still empty).
Add-Team button in	Call a `create_team` RPC (see code below) that checks `role='coach' OR 'director'`, inserts into `teams`, returns `team_id`.
Workout completion	Front-end inserts into `workout_logs`; a Postgres TRIGGER awards points into `points ledger` and broadcasts via Supabase Realtime.

5 . Parent → Child Account Pattern

A. Supabase stored procedure

```
create or replace function create_child(
  _parent uuid,
  _child_email text,
  _child_name text
) returns uuid
language plpgsql
security definer
as $$
declare
  new_user uuid;
begin
  -- 1. create Auth user
  new_user := auth.sign_up(
    email          := _child_email,
    password       := null,          -- will use magic link
    email_redirect_to := 'https://app.powlax.com/magic'
  ).user.id;

  -- 2. insert into users
  insert into public.users(id, role, full_name)
  values (new_user, 'player', _child_name);

  -- 3. link to parent
  insert into parent_child(parent_id, child_id)
  values (_parent, new_user);

  return new_user;
```

```
end;
$$;
```
RLS on `users` lets parents **only** call this function for themselves (`_parent = auth.uid()`).

B. React Hook (simplified)

```
import { supabase } from '@/supabaseClient';

export async function addChild(email: string, name: string)
{
  const { data, error } = await
supabase.rpc('create_child', {
    _parent:  (await
supabase.auth.getUser()).data.user?.id,
    _child_email: email,
    _child_name: name
  });
  if (error) throw error;
  return data; // child user_id
}
```

C. UI Flow

1. **Parent** logs in → `ParentDashboard`.
2. Click **"Add Child Player"** → Modal collects `name` + optional `email`.
3. On success, toast *"Invite sent to <email>. Your child will appear once they accept the link."*
4. Child uses the magic-link, lands in `SkillsAcademyHome` with role `player` and is *already* mapped to the parent and (via `team_id`) to the correct Team HQ.

6 . Points & Leaderboards without Gamipress

- **workout_logs** trigger:

```
create or replace function award_points() returns trigger
as $$
begin
  insert into points_ledger(user_id, delta, reason)
```

```
  values (new.player_id,
          (select point_value from workouts where id =
new.workout_id),
          'workout');
  return new;
end;
$$ language plpgsql;

create trigger trg_award after insert on workout_logs
for each row execute procedure award_points();
```

- **Leaderboard view**:

```
create view v_team_leaderboard as
  select p.team_id,
         u.full_name,
         sum(l.delta) as points
  from points_ledger l
  join users u on u.id = l.user_id
  join players p on p.id = u.id
  group by 1,2;
```

React subscribes with

```
supabase
  .from('v_team_leaderboard')
  .select('*')
  .eq('team_id', currentTeam)
  .order('points', { ascending: false });
```

Because it's a **view**, you get instant reactivity without extra code.

7 . Checklist to Get Moving

1. **Create Supabase project** → set up tables & RLS from Section 2.
2. Build **Edge Functions**: `sync_membership`, `create_child`, `create_team`.
3. Wire **MemberPress webhooks** to `sync_membership`.
4. Scaffold React auth with `@supabase/auth-ui-react`.
5. Implement the **Add Child** modal using `addChild()` hook above.

6. Migrate one LearnDash workout into `workouts` and test the points trigger.

7. Start replacing Team HQ pages with React views fed purely by Supabase.

Follow this flow and you'll keep WordPress & MemberPress right where they shine (sales and gating) while running the day-to-day, realtime lacrosse experience on Supabase—with seamless parent/child hierarchies baked in.