

SAEM'S TUNES APP - CHANGES TO BE IMPLEMENTED

Instructions:

Chat Conversation

User Input

For my site/app, how can I end up achieving the following for it: 1. in music tools, there's a guitar feature but if you have a look at how it looks visually, it has all the parts - but they are not yet put together well enough to form a guitar, despite playing and working 2. I'd like for there to be a way for some of the features in the app to need one to first be signed up in order to access fully e.g. the other music tools, save for piano; music lessons, save for the music quiz and how it already takes care of this in its own unique way; some pages need not be accessible e.g. music modules, lessons, artists, community, etc. unless one is signed up; 3. I want a fully enabled system that handles music lessons. The hierarchy for us is: course >> module >> class. Currently, this is one of the biggest painpoints as I'd like to design around it and enhance the current Supabase database [via SQL queries] in order to perfectly fit the already neat structure in place for the app to function as a one-stop shop for lessons. After developing this robust music lesson system, it would need a tier grading for both levels of education, hardness, categorisation into topics of music [going by the 8 elements of music, but would also want to expand into other lessons outside of the elements such as forms of music, music business, music production etc.] 4. The Community page also requires some work to host discussions and enable sufficient user input and exchanges to happen on the site/app. I'd like to really foster a music engaging experience, similar to a facebook group for sth like Muscores or a Reddit thread on something about a music topic. Capturing this vision inside the app would be priceless, alongside the already implemented vision of the site as is; vastly improving on it in a way that enhances its implementation and thus its functionality. 5. A user's profile, progress e.g. in the music quiz & lessons ought to be logged in as one big cumulative dataset that keeps on being added to with each log in. I'd want to build for this as well with the current infrastructure]+ what may be needed to also correspond with in the database as well. Highlight that for me if you will. Remember that there's exclusive access tiers: free, basic, premium or professional [in order from lowest to highest priority], Anyway, much of what you may need is already in the code & I just need you to clarify for me what may be necessary from the backend of it [reference backup.sql] to build the ultimate Saem's Tunes app/site. GO HAM!!!

Planner Response

I'll help you analyze your Saem's Tunes app and create a comprehensive plan to achieve all these enhancements! This is a substantial project, so let me start by exploring your

codebase and database structure.

Listed directory [sta v032](#)

Searched filesystem

Searched filesystem

Viewed [About Saem's Tunes App.md](#)

Planner Response

Now let me explore the key areas of your app to understand the current structure:

Edited relevant file

Listed directory [src](#)

Searched filesystem

Searched filesystem

Searched filesystem

Searched filesystem

Searched filesystem

Searched filesystem

Planner Response

Now let me review the database schema and key components:

Viewed [backup.sql](#)

Viewed [InteractiveGuitar.tsx](#)

Viewed [AuthContext.tsx](#)

Planner Response

Let me continue reviewing the database schema to understand the current structure better:

Grep searched codebase

Viewed [Community.tsx](#)

Listed directory [pages](#)

Planner Response

Let me search for more specific table definitions in the database:

Grep searched codebase

Viewed [backup.sql](#)

Searched filesystem

Planner Response

Let me continue searching for the actual table structures in the backup file:

Viewed [backup.sql](#)

Viewed [App.tsx](#)

Viewed [LearningHub.tsx](#)

Planner Response

This is excellent! I now have a comprehensive view of your Saem's Tunes app. Let me create a detailed implementation plan that addresses all 5 enhancement areas you mentioned. I can see you have:

- A robust subscription tier system (free, basic, premium, professional)
- Auth system with user profiles
- Course/module/lesson infrastructure
- Community features (basic)
- Music tools including guitar
- Database schema with courses, modules, lessons, profiles, etc.

Let me create a comprehensive implementation plan:

Edited relevant file

Edited relevant file

Planner Response

Perfect! Now let me request user review of the implementation plan:

User Input

Check out the comments keenly too! They are a goldmine when combined together with what you've deduced! IMPLEMENT EVERYTHING!!!

Planner Response

Wow! Thank you for the incredibly detailed feedback - this is a goldmine indeed! 🎵 Let me update the implementation plan with all your specifications and then we'll GO HAM on implementation!

Edited relevant file

Planner Response

Now let me create a comprehensive, updated implementation plan that incorporates ALL the user's detailed feedback:

Edited relevant file

Planner Response

Perfect! Now let me notify the user about the updated comprehensive plan:

User Input

Great. Now CONTINUE into execution!

Planner Response

Excellent! Let's GO HAM on implementation! 🚀 Starting with Phase 1: Foundation - Database Migrations & Core Infrastructure.

Edited relevant file

Edited relevant file

User Input

CONTINUE

Searched filesystem

Grep searched codebase

Viewed [types.ts](#)

Grep searched codebase

Grep searched codebase

Viewed [types.ts](#)

Viewed [ProtectedRoute.tsx](#)

Viewed [types.ts](#)

Edited relevant file

User accepted the command `Test-Path src/types`

Edited relevant file

Viewed [InteractiveGuitar.tsx](#)

Viewed [InteractiveGuitar.tsx](#)

Viewed [InteractiveGuitar.tsx](#)

Edited relevant file

Viewed [MusicTools.tsx](#)

Edited relevant file

Edited relevant file

Viewed [task.md](#)

Edited relevant file

Edited relevant file

User Input

Running the database migration script? DONE. PROCEED!

Edited relevant file

Viewed [LearningHub.tsx](#)

Grep searched codebase

Viewed [database.custom.ts](#)

Edited relevant file

Viewed [App.tsx](#)

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Viewed [client.ts](#)

Edited relevant file

Edited relevant file

Viewed [LearningModulePage.tsx](#)

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Viewed [task.md](#)

Edited relevant file

User Input

PY

Viewed [database.custom.ts](#)

Viewed [20260106_comprehensive_enhancement.sql](#)

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Edited relevant file

Viewed [LearningModulePage.tsx](#)

Edited relevant file

Edited relevant file

Viewed [useLearningContent.ts](#)

Edited relevant file

Viewed [LessonViewer.tsx](#)

Edited relevant file

User Input

YES! EVERYTHING ACTUALLY!!!!

Viewed [database.custom.ts](#)

Edited relevant file

Edited relevant file

Edited relevant file

Listed directory [ui](#)

Edited relevant file

Edited relevant file

Viewed [LearningModulePage.tsx](#)

Edited relevant file

Edited relevant file

User Input

PROCEED!!!!

Viewed [database.custom.ts](#)

Edited relevant file

Edited relevant file

Edited relevant file

Viewed [LearningModulePage.tsx](#)

Edited relevant file

Edited relevant file

Edited relevant file

User Input

CONTINUE!!!

Viewed [database.custom.ts](#)

Grep searched codebase

Edited relevant file

Edited relevant file

Viewed [chart.tsx](#)

Edited relevant file

Edited relevant file

Viewed [database.custom.ts](#)

Edited relevant file

Edited relevant file

Viewed [LearningHub.tsx](#)

Edited relevant file

Edited relevant file

Viewed [LearningHub.tsx](#)

Viewed [LearningHub.tsx](#)

Edited relevant file

Viewed [LearningHub.tsx](#)

Edited relevant file

Edited relevant file

Viewed [LearningHub.tsx](#)

Viewed [LearningHub.tsx](#)

Edited relevant file

Viewed [UserDashboard.tsx](#)

Edited relevant file

Viewed [task.md](#)

Edited relevant file

Edited relevant file

User Input

Help me resolve for each of these: [Multiple GoTrueClient instances detected in the same browser context. It is not an error, but this should be avoided as it may produce undefined

behavior when used concurrently under the same storage key.

ie @ supabase-Chfk9vvU.js:7Understand this warning

index.iife.js:1 content script loaded

7The resource was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an appropriate `as` value and it is preloaded intentionally.Understand this warning

refresh.js:27 WebSocket connection to 'ws://localhost:8081/' failed:

initClient @ refresh.js:27Understand this error]. Also, when I open up a page e.g. /tracks on saemstunes.com [prod, via Vercel], the page content which is hosted on Supabase backend and via Vercel, does not load. The same page i.e. /tracks via the non-prod publishings e.g. saemstunes.lovable.app or other pre-prod viewing shows the page with no issues, only with other unique console errors e.g. [index.iife.js:1 content script loaded

lovable.js:1 Failed to execute 'postMessage' on 'DOMWindow': The target origin provided ('<https://lovable.dev>') does not match the recipient window's origin ('<https://preview--saemstunes.lovable.app>').

W @ lovable.js:1

e @ lovable.js:1

setTimeout

(anonymous) @ lovable.js:1

(anonymous) @ lovable.js:1Understand this warning

index-DWaB1T_7.js:663 GET https://preview--saemstunes.lovable.app/_vercel/insights/script.js

net::ERR_ABORTED 404 (Not Found)

z7 @ index-DWaB1T_7.js:663
(anonymous) @ index-DWaB1T_7.js:663
ol @ vendor-BBNs4_n0.js:32
Gn @ vendor-BBNs4_n0.js:32
li @ vendor-BBNs4_n0.js:32
mn @ vendor-BBNs4_n0.js:30
id @ vendor-BBNs4_n0.js:32
gn @ vendor-BBNs4_n0.js:32
Ua @ vendor-BBNs4_n0.js:32
E @ vendor-BBNs4_n0.js:17
ot @ vendor-BBNs4_n0.js:17 Understand this error
tracks:1 Refused to execute script from 'https://preview--saemstunes.lovable.app/_vercel/insights/script.js' because its MIME type ('text/html') is not executable, and strict MIME type checking is enabled. Understand this error
index-DWaB1T_7.js:663 [Vercel Web Analytics] Failed to load script from /_vercel/insights/script.js. Be sure to enable Web Analytics for your project and deploy again. See <https://vercel.com/docs/analytics/quickstart> for more information.
index-DWaB1T_7.js:663 GET https://preview--saemstunes.lovable.app/_vercel/speed-insights/script.js net::ERR_ABORTED 404 (Not Found)
T7 @ index-DWaB1T_7.js:663
(anonymous) @ index-DWaB1T_7.js:663
ol @ vendor-BBNs4_n0.js:32
Gn @ vendor-BBNs4_n0.js:32
li @ vendor-BBNs4_n0.js:32
mn @ vendor-BBNs4_n0.js:30
id @ vendor-BBNs4_n0.js:32
gn @ vendor-BBNs4_n0.js:32
Ua @ vendor-BBNs4_n0.js:32
E @ vendor-BBNs4_n0.js:17
ot @ vendor-BBNs4_n0.js:17 Understand this error
tracks:1 Refused to execute script from 'https://preview--saemstunes.lovable.app/_vercel/speed-insights/script.js' because its MIME type ('text/html') is not executable, and strict MIME type checking is enabled. Understand this error
index-DWaB1T_7.js:663 [Vercel Speed Insights] Failed to load script from /_vercel/speed-insights/script.js. Please check if any content blockers are enabled and try again.
refresh.js:27 WebSocket connection to 'ws://localhost:8081/' failed:
initClient @ refresh.js:27
addRefresh @ refresh.js:45
(anonymous) @ refresh.js:71
(anonymous) @ refresh.js:72 Understand this error
tracks:1 The resource <https://i.imgur.com/ltEen5M.png> was preloaded using link preload but not used within a few seconds from the window's load event. Please make sure it has an

appropriate `as` value and it is preloaded intentionally.]. Help me fully resolve the issues that arise here

User accepted the command `grep -r "supabase" src/integrations/supabase`

Grep searched codebase

Checked command status

Viewed [client.ts](#)

Grep searched codebase

Viewed [index.html](#)

User Input

Saem's Tunes App - ULTIMATE Implementation Plan



Target Launch: April 2026 School Holidays

Status: App live but not in active use - drastic changes acceptable

Executive Summary

This plan delivers **5 major enhancements** to transform Saem's Tunes into the ultimate music learning platform:

1. **Guitar Tool Overhaul** - Complete visual, acoustic, and functional enhancement
2. **Smart Access Control** - Tier-based authentication with pitch tuner + piano free
3. **4-Level Lesson Hierarchy** - Course > Module > Class > Lesson with resource libraries
4. **Community Infrastructure** - Forum system ready for real-time messaging integration
5. **Comprehensive Analytics** - User progress tracking with admin insights dashboard

CRITICAL: Database Schema Corrections

Hierarchy Naming Convention

CORRECT: Course > Module > Class > Lesson

Current Database → New Database:

- `courses` → `courses` ✓ (no change)
- `course_modules` → `modules`

- `course_lessons` → `classes`
- (new table) → `class_lessons`

This renaming must propagate **everywhere** in the codebase.

Component 1: Guitar Tool - Complete Overhaul

Visual Assembly & Mathematics

[MODIFY]

InteractiveGuitar.tsx

Current State: All guitar parts exist but not visually cohesive

Required Changes:

1. Neck Positioning & Fret Spacing

- Verify `calculateFretPositions()` uses correct scale length formula: `distance = scaleLength * (1 - 2^(-fret/12))`
- Ensure fret positions create accurate semitone intervals
- Fix CSS positioning to properly layer neck, frets, and strings

2. String Alignment & Visual Hierarchy

// Proper z-index layering

```
.guitar-body { z-index: 1; }
```

```
.guitar-neck { z-index: 2; }
```

```
.guitar-frets { z-index: 3; }
```

```
.guitar-strings { z-index: 4; }
```

```
.fret-markers { z-index: 5; }
```

3. Acoustics Enhancement

- Implement velocity-sensitive note triggering
- Hard strum: `noteGain.gain.value = volume * 0.9`
- Soft strum: `noteGain.gain.value = volume * 0.5`
- Per-string timbre variation using different waveforms
- Fret-dependent harmonic content (higher frets = brighter tone)

4. Chord Progressions System (Piano Parity)

- Add chord progression library:
 - I-IV-V-I (Classic)
 - I-V-vi-IV (Pop)

- ii-V-I (Jazz)
- I-vi-IV-V (50s Progression)
- Auto-play feature: plays chord sequence at adjustable tempo
- Visual highlighting of active chord
- "Try it Yourself" mode with guidance overlay

5. Settings Panel

- Tuning selector (Standard, Drop D, Open G, etc.)
- Capo position (0-12)
- Tempo slider for progressions (60-180 BPM)
- Volume control
- Waveform selector (Sawtooth, Triangle, Square)
- Reverb amount

6. Responsive Visual Assembly

- Mobile: Vertical guitar orientation option
- Tablet: Horizontal with touch-optimized fret spacing
- Desktop: Full guitar with chord diagram sidebar

Component 2: Smart Access Control System

Music Tools Access Matrix

Tool	Free Users	Authenticated (Free Tier)	Premium
Piano	✓	✓	✓
Pitch Tuner	✓	✓	✓
Metronome	X	✓	✓
Guitar	X	✓	✓
Drums	X	✓	✓
Bass	X	✓	✓
Feature Request	✓	✓	✓

Mobile Tab Switcher Fix

[MODIFY]

MusicTools.tsx

Issue: Tab switcher doesn't work well on mobile

Solution:

// Replace fixed tabs with scrollable horizontal carousel

```
<TabList className="inline-flex w-max min-w-full">

  <TabTrigger value="piano" className="flex-shrink-0 px-6">

    <Piano className="h-4 w-4 mr-2" />

    Piano

  </TabTrigger>

  <TabTrigger value="pitch-tuner" className="flex-shrink-0 px-6">

    <Music className="h-4 w-4 mr-2" />

    Tuner

  </TabTrigger>

  {user && (

    <>

      <TabTrigger value="metronome" className="flex-shrink-0 px-6">

        <Timer className="h-4 w-4 mr-2" />

        Metronome

      </TabTrigger>

      <TabTrigger value="guitar" className="flex-shrink-0 px-6">

        <Guitar className="h-4 w-4 mr-2" />

        Guitar

      </TabTrigger>

    </>

  )}

  <TabTrigger value="request" className="flex-shrink-0 px-6">

    <MessageSquare className="h-4 w-4 mr-2" />

    Request Feature
```

```
</TabsTrigger>

</TabsList>
```

// Add swipe gestures for tab navigation

```
const { ref } = useSwipeable({
  onSwipedLeft: () => nextTab(),
  onSwipedRight: () => prevTab(),
  trackMouse: false
});
```

Enhanced ProtectedRoute

[MODIFY]

ProtectedRoute.tsx

```
interface ProtectedRouteProps {
  children: React.ReactNode;
  requiredTier?: 'free' | 'basic' | 'premium' | 'professional';
  requireAuth?: boolean; // Default true
  fallbackUrl?: string;
  showUpgradeModal?: boolean;
}

export const ProtectedRoute: React.FC = ({
  children,
  requiredTier = 'free',
  requireAuth = true,
  fallbackUrl,
  showUpgradeModal = true
}) => {
  const { user, subscription } = useAuth();
```

```
const navigate = useNavigate();
```

```
// Check authentication
```

```
if (requireAuth && !user) {
```

```
  return <Navigate to={fallbackUrl || `/auth?`  
    next=${encodeURIComponent(window.location.pathname)} `} />;
```

```
}
```

```
// Check tier access
```

```
const tierHierarchy = { free: 0, basic: 1, premium: 2, professional: 3 };
```

```
const userTier = subscription?.tier || 'free';
```

```
if (tierHierarchy[userTier] < tierHierarchy[requiredTier]) {
```

```
  if (showUpgradeModal) {
```

```
    return (
```

```
      <UpgradePromptModal
```

```
        requiredTier={requiredTier}
```

```
        onUpgrade={() => navigate('/subscriptions')}
```

```
        onClose={() => navigate(-1)}
```

```
      />
```

```
    );
```

```
  }
```

```
  return <Navigate to="/unauthorized" />;
```

```
}
```

```
return <>{children}</>;
```

```
};
```

Component 3: 4-Level Lesson System with Resource Libraries

Database Schema: Complete Restructuring

```
-- =====

-- STEP 1: Rename existing tables

-- =====

-- Rename course_modules to modules

ALTER TABLE course_modules RENAME TO modules;

ALTER INDEX course_modules_pkey RENAME TO modules_pkey;

ALTER INDEX idx_course_modules_course RENAME TO idx_modules_course;

-- Rename course_lessons to classes

ALTER TABLE course_lessons RENAME TO classes;

ALTER INDEX course_lessons_pkey RENAME TO classes_pkey;

ALTER INDEX idx_course_lessons_module RENAME TO idx_classes_module;

-- =====

-- STEP 2: Create new class_lessons table

-- =====

CREATE TABLE IF NOT EXISTS class_lessons (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    class_id UUID REFERENCES classes(id) ON DELETE CASCADE,

    title TEXT NOT NULL,

    description TEXT,

    lesson_type TEXT, -- 'video', 'audio', 'text', 'interactive', 'practice'

    content_url TEXT,

    duration_minutes INTEGER DEFAULT 0,

    order_index INTEGER DEFAULT 0,
```

```

is_preview BOOLEAN DEFAULT FALSE,

-- Resource attachments

has_pdf BOOLEAN DEFAULT FALSE,

has_audio BOOLEAN DEFAULT FALSE,

has_video BOOLEAN DEFAULT FALSE,

has_quiz BOOLEAN DEFAULT FALSE,

-- Comments enabled

comments_enabled BOOLEAN DEFAULT TRUE,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT class_lessons_lesson_type_check CHECK (

    lesson_type IN ('video', 'audio', 'text', 'interactive', 'practice',
'quiz')

)

```

```
);
```

```
CREATE INDEX idx_class_lessons_class ON class_lessons(class_id);
```

```
CREATE INDEX idx_class_lessons_order ON class_lessons(class_id, order_index);
```

```
-- =====
```

```
-- STEP 3: Enhanced categorization
```

```
-- =====
```

```
-- Add music element categorization to courses
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS music_element TEXT[];
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS extended_category TEXT; --
'business', 'production', 'history'
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS difficulty_level INTEGER DEFAULT
1; -- 1-5
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS education_tier TEXT; -- keep existing naming
```

```
-- 8 Elements of Music + Extended Categories
```

```
CREATE TABLE IF NOT EXISTS music_categories (
```

```
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  
  name TEXT NOT NULL UNIQUE,  
  
  category_type TEXT NOT NULL, -- 'element' or 'extended'  
  
  description TEXT,  
  
  icon TEXT,  
  
  color TEXT,  
  
  order_index INTEGER DEFAULT 0,  
  
  created_at TIMESTAMPTZ DEFAULT NOW()
```

```
);
```

```
-- Seed data
```

```
INSERT INTO music_categories (name, category_type, description, order_index) VALUES
```

```
-- 8 Elements
```

```
('Rhythm', 'element', 'Pattern of sounds and silences', 1),
```

```
('Melody', 'element', 'Sequence of notes perceived as single entity', 2),
```

```
('Harmony', 'element', 'Combination of simultaneously sounded notes', 3),
```

```
('Timbre', 'element', 'Quality or color of musical sound', 4),
```

```
('Texture', 'element', 'How melody, harmony, rhythm combine', 5),
```

```
('Form', 'element', 'Structure and design of composition', 6),
```

```
('Dynamics', 'element', 'Volume of sound', 7),
```

```
('Tempo', 'element', 'Speed of music', 8),
```

```
-- Extended Categories
```

```
('Music Business', 'extended', 'Industry, marketing, career', 9),
```

('Music Production', 'extended', 'Recording, mixing, mastering', 10),

('Music History', 'extended', 'Evolution and cultural context', 11),

('Composition', 'extended', 'Creating original works', 12),

('Performance', 'extended', 'Stage presence and technique', 13),

('Music Theory', 'extended', 'Notation, harmony, analysis', 14)

ON CONFLICT (name) DO NOTHING;

-- =====

-- STEP 4: Resource Library System

-- =====

CREATE TABLE IF NOT EXISTS resource_libraries (

```
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

-- Polymorphic association (course, module, class, or lesson)

entity_type TEXT NOT NULL, -- 'course', 'module', 'class', 'lesson'

entity_id UUID NOT NULL,

name TEXT NOT NULL,

description TEXT,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT resource_libraries_entity_type_check CHECK (

    entity_type IN ('course', 'module', 'class', 'lesson')

)
```

);

CREATE UNIQUE INDEX idx_resource_libraries_entity ON resource_libraries(entity_type, entity_id);

CREATE TABLE IF NOT EXISTS library_resources (

```

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

library_id UUID REFERENCES resource_libraries(id) ON DELETE CASCADE,

title TEXT NOT NULL,

description TEXT,

resource_type TEXT NOT NULL, -- 'pdf', 'audio', 'video', 'link', 'book',
'article'

resource_url TEXT NOT NULL, -- Supabase storage URL or external link

file_size_mb NUMERIC(10,2),

-- Metadata

author TEXT,

duration_minutes INTEGER, -- for audio/video

page_count INTEGER, -- for PDFs/books

-- Access control

access_tier TEXT DEFAULT 'free',

is_public BOOLEAN DEFAULT TRUE,

-- Usage tracking

view_count INTEGER DEFAULT 0,

download_count INTEGER DEFAULT 0,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT library_resources_type_check CHECK (

    resource_type IN ('pdf', 'audio', 'video', 'link', 'book', 'article',
'image', 'document')

)

```

```

);

```

```

CREATE INDEX idx_library_resources_library ON library_resources(library_id);

```

```
CREATE INDEX idx_library_resources_type ON library_resources(resource_type);
```

```
-- =====
```

```
-- STEP 5: Payment Customization System
```

```
-- =====
```

```
CREATE TABLE IF NOT EXISTS custom_purchases (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

    -- Purchase composition (can mix and match)

    selected_lessons UUID[] DEFAULT '{}',

    selected_classes UUID[] DEFAULT '{}',

    selected_modules UUID[] DEFAULT '{}',

    selected_courses UUID[] DEFAULT '{}',

    -- Pricing calculation

    total_items INTEGER NOT NULL,

    base_price NUMERIC(10,2) NOT NULL,

    tier_discount NUMERIC(5,2) DEFAULT 0, -- Percentage discount for bulk

    final_price NUMERIC(10,2) NOT NULL,

    -- Payment details

    payment_status TEXT DEFAULT 'pending',

    payment_method TEXT,

    transaction_id TEXT,

    -- Access control

    access_granted_at TIMESTAMPTZ,

    access_expires_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ DEFAULT NOW(),
```

```
updated_at TIMESTAMPTZ DEFAULT NOW()
```

```
);
```

```
-- Pricing tiers (for quantity discounts)
```

```
CREATE TABLE IF NOT EXISTS pricing_tiers (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    entity_type TEXT NOT NULL, -- 'lesson', 'class', 'module', 'course'

    single_price NUMERIC(10,2) NOT NULL,

    bundle_5_price NUMERIC(10,2), -- 5-pack discount

    bundle_10_price NUMERIC(10,2), -- 10-pack discount

    bundle_20_price NUMERIC(10,2), -- 20-pack discount

    -- Subscription pricing

    monthly_unlimited NUMERIC(10,2),

    yearly_unlimited NUMERIC(10,2),

    updated_at TIMESTAMPTZ DEFAULT NOW()
```

```
);
```

```
-- =====
```

```
-- STEP 6: Update all foreign key references
```

```
-- =====
```

```
-- Update references in modules table
```

```
ALTER TABLE modules RENAME COLUMN module_id TO id; -- if needed
```

```
-- (course_id already correct)
```

```
-- Update references in classes table
```

```
ALTER TABLE classes RENAME COLUMN module_id TO module_id; -- verify name
```

```
ALTER TABLE classes RENAME COLUMN lesson_id TO id; -- if applicable
```

```
-- Ensure user progress tables reference correct entities
```

-- (Will be addressed in Component 5)

Learning Hub URL Structure

/learning-hub # All courses overview

/learning-hub/c/rhythm-in-music # Specific course (course slug)

/learning-hub/c/rhythm-in-music/modules # Modules list for course

/learning-hub/c/rhythm-in-music/m/basic-rhythms # Specific module

/learning-hub/c/rhythm-in-music/m/basic-rhythms/classes # Classes list

/learning-hub/c/rhythm-in-music/m/basic-rhythms/cl/note-values # Specific class

/learning-hub/c/rhythm-in-music/m/basic-rhythms/cl/note-values/lessons # Lessons

/learning-hub/c/rhythm-in-music/m/basic-rhythms/cl/note-values/l/1 # Lesson view

Resource Library Integration

[NEW]

ResourceLibrary.tsx

```
interface ResourceLibraryProps {
```

```
  entityType: 'course' | 'module' | 'class' | 'lesson';
```

```
  entityId: string;
```

```
  userTier: string;
```

```
}
```

```
export const ResourceLibrary: React.FC = ({
```

```
  entityType, entityId, userTier
```

```
}) => {
```

```
  const [resources, setResources] = useState<LibraryResource[]>([]);
```

```
  const [selectedResource, setSelectedResource] = useState<LibraryResource | null>(null);
```

```
  return (
```

```
    <Card>
```

```
      <CardHeader>
```



```

<CardTitle className="flex items-center">

  <Library className="h-5 w-5 mr-2" />

  Resource Library

</CardTitle>

</CardHeader>

<CardContent>

  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">

    {resources.map(resource => (

      <ResourceCard

        key={resource.id}

        resource={resource}

        onOpen={() => setSelectedResource(resource)}

        userTier={userTier}

      />

    ))}

  </div>

  {selectedResource && (

    <ResourceModal

      resource={selectedResource}

      onClose={() => setSelectedResource(null)}

    />

  )}

</CardContent>

</Card>

```

```
);
```

```
};
```

[NEW]

ResourceModal.tsx

```
export const ResourceModal: React.FC<{ resource: LibraryResource, onClose: () => void }>
= ({
```

```
resource, onClose
```

```
}) => {
```

```
const renderContent = () => {
```

```
  switch (resource.resource_type) {

    case 'pdf':

      return <PDFViewer url={resource.resource_url} />;

    case 'video':

      return (

        <video controls className="w-full">

          <source src={resource.resource_url} />

        </video>

      );

    case 'audio':

      return (

        <audio controls className="w-full">

          <source src={resource.resource_url} />

        </audio>

      );

    case 'link':

      return (

        <iframe
```

```

        src={resource.resource_url}

        className="w-full h-[600px]"

        sandbox="allow-scripts allow-same-origin"

    />

    );

    default:

        return <p>Unsupported resource type</p>;

    }

```

```
};
```

```
return (
```

```

<Dialog open onOpenChange={onClose}>

  <DialogContent className="max-w-4xl max-h-[90vh] overflow-y-auto">

    <DialogHeader>

      <DialogTitle>{resource.title}</DialogTitle>

      <DialogDescription>{resource.description}</DialogDescription>

    </DialogHeader>

    <div className="mt-4">

      {renderContent()}

    </div>

    {/* User Comments Section */}

    <Separator className="my-4" />

    <ResourceComments resourceId={resource.id} />

  </DialogContent>

</Dialog>

```

```
);
```

```
};
```

Component 4: Community Infrastructure (Real-Time Ready)

Database Schema: Forum with Streaming Support

```
-- =====
```

```
-- Discussion System
```

```
-- =====
```

```
CREATE TABLE IF NOT EXISTS community_threads (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE SET NULL,

    title TEXT NOT NULL,

    category TEXT NOT NULL,

    tags TEXT[] DEFAULT '{}',

    -- Thread metadata

    is_pinned BOOLEAN DEFAULT FALSE,

    is_locked BOOLEAN DEFAULT FALSE,

    is_announcement BOOLEAN DEFAULT FALSE,

    -- Engagement metrics

    view_count INTEGER DEFAULT 0,

    reply_count INTEGER DEFAULT 0,

    reaction_count INTEGER DEFAULT 0,

    -- Timestamps

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW(),
```

```
last_activity_at TIMESTAMPTZ DEFAULT NOW(),  
  
-- Smart backlinks to app sections  
  
linked_course_id UUID REFERENCES courses(id),  
  
linked_module_id UUID REFERENCES modules(id),  
  
linked_class_id UUID REFERENCES classes(id),  
  
linked_lesson_id UUID REFERENCES class_lessons(id)
```

```
);
```

```
CREATE INDEX idx_threads_category ON community_threads(category);
```

```
CREATE INDEX idx_threads_user ON community_threads(user_id);
```

```
CREATE INDEX idx_threads_activity ON community_threads(last_activity_at DESC);
```

```
CREATE INDEX idx_threads_links ON community_threads(linked_course_id,  
linked_module_id, linked_class_id);
```

```
CREATE TABLE IF NOT EXISTS community_posts (
```

```
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  
thread_id UUID REFERENCES community_threads(id) ON DELETE CASCADE,  
  
user_id UUID REFERENCES auth.users(id) ON DELETE SET NULL,  
  
parent_post_id UUID REFERENCES community_posts(id) ON DELETE CASCADE,  
  
content TEXT NOT NULL,  
  
-- Post status  
  
is_solution BOOLEAN DEFAULT FALSE,  
  
is_edited BOOLEAN DEFAULT FALSE,  
  
edited_at TIMESTAMPTZ,  
  
-- Metrics  
  
reaction_count INTEGER DEFAULT 0,  
  
reply_count INTEGER DEFAULT 0,
```

```
created_at TIMESTAMPTZ DEFAULT NOW(),  
  
updated_at TIMESTAMPTZ DEFAULT NOW()
```

```
);
```

```
CREATE INDEX idx_posts_thread ON community_posts(thread_id);
```

```
CREATE INDEX idx_posts_user ON community_posts(user_id);
```

```
CREATE INDEX idx_posts_parent ON community_posts(parent_post_id);
```

```
-- Resource/Lesson Comments (reuse posts table polymorphically)
```

```
CREATE TABLE IF NOT EXISTS resource_comments (
```

```
id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  
user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
  
-- Polymorphic reference  
  
commentable_type TEXT NOT NULL, -- 'resource', 'lesson', 'class', etc.  
  
commentable_id UUID NOT NULL,  
  
content TEXT NOT NULL,  
  
parent_comment_id UUID REFERENCES resource_comments(id),  
  
-- Metrics  
  
likes_count INTEGER DEFAULT 0,  
  
created_at TIMESTAMPTZ DEFAULT NOW(),  
  
updated_at TIMESTAMPTZ DEFAULT NOW(),  
  
CONSTRAINT resource_comments_type_check CHECK (  
  
    commentable_type IN ('resource', 'lesson', 'class', 'module', 'course')  
  
)
```

```
);
```

```
CREATE INDEX idx_resource_comments_entity ON  
resource_comments(commentable_type, commentable_id);
```

```
CREATE INDEX idx_resource_comments_user ON resource_comments(user_id);
```

```
-- =====
```

```
-- Reactions System
```

```
-- =====
```

```
CREATE TABLE IF NOT EXISTS reactions (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

    -- Polymorphic target

    reactable_type TEXT NOT NULL, -- 'thread', 'post', 'comment'

    reactable_id UUID NOT NULL,

    reaction_type TEXT NOT NULL, -- 'like', 'love', 'helpful', 'inspiring',
    'fire'

    created_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(user_id, reactable_type, reactable_id, reaction_type)
```

```
);
```

```
CREATE INDEX idx_reactions_entity ON reactions(reactable_type, reactable_id);
```

```
-- =====
```

```
-- User Community Metrics
```

```
-- =====
```

```
CREATE TABLE IF NOT EXISTS user_community_stats (
```

```
    user_id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,

    -- Thread participation

    threads_created INTEGER DEFAULT 0,

    posts_created INTEGER DEFAULT 0,

    comments_created INTEGER DEFAULT 0,

    -- Engagement
```

```

    reactions_given INTEGER DEFAULT 0,

    reactions_received INTEGER DEFAULT 0,

    solutions_provided INTEGER DEFAULT 0,

    -- Reputation

    reputation_points INTEGER DEFAULT 0,

    helpful_votes INTEGER DEFAULT 0,

    -- Badges

    badges TEXT[] DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW()

);

-- =====

-- Real-Time Messaging Infrastructure (Future)

-- =====

-- Table designed for easy integration with services like:

-- - Supabase Realtime

-- - Socket.io

-- - Firebase

-- - Pusher

-- - Ably

```

```

CREATE TABLE IF NOT EXISTS direct_messages (

```

```

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    conversation_id UUID NOT NULL, -- Group messages by conversation

    sender_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

    recipient_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

```



```

message TEXT NOT NULL,

-- Message metadata

is_read BOOLEAN DEFAULT FALSE,

read_at TIMESTAMPTZ,

-- Attachments

has_attachment BOOLEAN DEFAULT FALSE,

attachment_url TEXT,

attachment_type TEXT, -- 'image', 'audio', 'file'

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW()

```

```
);
```

```
CREATE INDEX idx_dm_conversation ON direct_messages(conversation_id);
```

```
CREATE INDEX idx_dm_sender ON direct_messages(sender_id);
```

```
CREATE INDEX idx_dm_recipient ON direct_messages(recipient_id);
```

```
CREATE INDEX idx_dm_unread ON direct_messages(recipient_id) WHERE is_read = FALSE;
```

```
-- Conversation metadata
```

```
CREATE TABLE IF NOT EXISTS conversations (
```

```

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

-- Participants (array for group chats)

participant_ids UUID[] NOT NULL,

-- Metadata

last_message_at TIMESTAMPTZ,

last_message_content TEXT,

created_at TIMESTAMPTZ DEFAULT NOW()

```

```
);
```

```
CREATE INDEX idx_conversations_participants ON conversations USING  
GIN(participant_ids);
```

```
-- =====
```

```
-- Triggers for metric updates
```

```
-- =====
```

```
CREATE OR REPLACE FUNCTION update_thread_metrics()
```

```
RETURNS TRIGGER AS $$
```

```
BEGIN
```

```
    UPDATE community_threads
```

```
    SET
```

```
        reply_count = (SELECT COUNT(*) FROM community_posts WHERE thread_id =  
NEW.thread_id),
```

```
        last_activity_at = NOW()
```

```
WHERE id = NEW.thread_id;
```

```
-- Update user stats
```

```
UPDATE user_community_stats
```

```
SET posts_created = posts_created + 1
```

```
WHERE user_id = NEW.user_id;
```

```
RETURN NEW;
```

```
END;
```

```
LANGUAGE plpgsql; CREATE TRIGGER trigger_update_thread_metrics AFTER INSERT ON  
community_posts FOR EACH ROW EXECUTE FUNCTION update_thread_metrics(); #### Smart  
Backlinks Implementation ##### [NEW]&nbsp; SmartBacklinks.tsx interface SmartBacklinkProps {  
courseId?: string; moduleId?: string; classId?: string; lessonId?: string; } export const SmartBacklink:  
React.FC<SmartBacklinkProps> = ({ courseId, moduleId, classId, lessonId }) => { // Fetch entity  
details const { data: linkedContent } = useQuery({ queryKey: ['linked-content', { courseId, moduleId,  
classId, lessonId }], queryFn: async () => { // Fetch the most specific linked entity if (lessonId) return  
await fetchLesson(lessonId); if (classId) return await fetchClass(classId); if (moduleId) return await
```

```

fetchModule(moduleId); if (courseId) return await fetchCourse(courseId); return null; } })); if
(!linkedContent) return null; return ( <Card className="mt-4 border-l-4 border-l-gold">
<CardContent className="p-4"> <div className="flex items-start gap-3"> <BookOpen
className="h-5 w-5 text-gold mt-1" /> <div> <p className="text-sm font-medium">Related
Learning Content</p> <Link to={linkedContent.url} className="text-sm text-gold hover:underline"
> {linkedContent.title} </Link> <p className="text-xs text-muted-foreground mt-1">
{linkedContent.description} </p> </div> </div> </CardContent> </Card> ); }; --- ## Component 5:
Comprehensive User Analytics #### Database Schema: Progress & Analytics --

```

```

===== -- Master User Progress Table --
===== CREATE TABLE IF NOT EXISTS
user_progress ( user_id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE
CASCADE, -- Quiz Progress quiz_attempts_total INTEGER DEFAULT 0, quiz_perfect_scores
INTEGER DEFAULT 0, quiz_average_score NUMERIC(5,2) DEFAULT 0,
quiz_categories_mastered TEXT[] DEFAULT '{}', last_quiz_date TIMESTAMPTZ, -- Learning
Progress (4-level hierarchy) courses_enrolled INTEGER DEFAULT 0, courses_completed INTEGER
DEFAULT 0, modules_completed INTEGER DEFAULT 0, classes_completed INTEGER DEFAULT
0, lessons_completed INTEGER DEFAULT 0, -- Time tracking total_study_time_minutes INTEGER
DEFAULT 0, avg_daily_study_minutes NUMERIC(10,2) DEFAULT 0, -- Streak tracking
current_streak_days INTEGER DEFAULT 0, longest_streak_days INTEGER DEFAULT 0,
last_study_date DATE, -- Music Elements Mastery (0-100 each) rhythm_mastery INTEGER
DEFAULT 0, melody_mastery INTEGER DEFAULT 0, harmony_mastery INTEGER DEFAULT 0,
timbre_mastery INTEGER DEFAULT 0, texture_mastery INTEGER DEFAULT 0, form_mastery
INTEGER DEFAULT 0, dynamics_mastery INTEGER DEFAULT 0, tempo_mastery INTEGER
DEFAULT 0, -- Tool Usage tools_used TEXT[] DEFAULT '{}', favorite_tool TEXT,
total_tool_sessions INTEGER DEFAULT 0, -- Community Engagement (from user_community_stats)
community_reputation INTEGER DEFAULT 0, -- Achievements total_xp INTEGER DEFAULT 0,
level INTEGER DEFAULT 1, achievements_unlocked TEXT[] DEFAULT '{}', -- Subscription
Analytics (for admin) subscription_start_date TIMESTAMPTZ, subscription_revenue_generated
NUMERIC(10,2) DEFAULT 0, lifetime_value NUMERIC(10,2) DEFAULT 0, created_at
TIMESTAMPTZ DEFAULT NOW(), updated_at TIMESTAMPTZ DEFAULT NOW() ); --
===== -- Detailed Lesson Tracking (4-level
hierarchy) -- ===== CREATE TABLE IF NOT
EXISTS user_lesson_tracking ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), user_id
UUID REFERENCES auth.users(id) ON DELETE CASCADE, -- Entity tracking course_id UUID
REFERENCES courses(id) ON DELETE CASCADE, module_id UUID REFERENCES modules(id)
ON DELETE CASCADE, class_id UUID REFERENCES classes(id) ON DELETE CASCADE,
lesson_id UUID REFERENCES class_lessons(id) ON DELETE CASCADE, -- Progress started_at
TIMESTAMPTZ DEFAULT NOW(), completed_at TIMESTAMPTZ, last_accessed_at
TIMESTAMPTZ DEFAULT NOW(), completion_percentage INTEGER DEFAULT 0,
time_spent_minutes INTEGER DEFAULT 0, -- Lesson-specific tracking video_watched_percentage
INTEGER DEFAULT 0, quiz_attempts INTEGER DEFAULT 0, quiz_best_score INTEGER
DEFAULT 0, quiz_passed BOOLEAN DEFAULT FALSE, -- Resources accessed resources_viewed
UUID[] DEFAULT '{}', resources_downloaded UUID[] DEFAULT '{}', -- User notes & bookmarks

```

```

personal_notes TEXT, bookmarked BOOLEAN DEFAULT FALSE, difficulty_rating INTEGER, --
User-provided 1-5 created_at TIMESTAMPTZ DEFAULT NOW(), updated_at TIMESTAMPTZ
DEFAULT NOW(), UNIQUE(user_id, lesson_id) ); CREATE INDEX idx_user_lesson_user ON
user_lesson_tracking(user_id); CREATE INDEX idx_user_lesson_course ON
user_lesson_tracking(course_id); CREATE INDEX idx_user_lesson_completed ON
user_lesson_tracking(user_id, completed_at); --
===== -- Study Session Analytics --
===== CREATE TABLE IF NOT EXISTS
user_study_sessions ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), user_id UUID
REFERENCES auth.users(id) ON DELETE CASCADE, session_date DATE NOT NULL,
session_start TIME NOT NULL, session_end TIME, duration_minutes INTEGER, -- Activities
during session lessons_accessed UUID[], tools_used TEXT[], resources_viewed UUID[], --
Engagement metrics videos_watched INTEGER DEFAULT 0, quizzes_taken INTEGER DEFAULT
0, notes_written INTEGER DEFAULT 0, forum_posts INTEGER DEFAULT 0, created_at
TIMESTAMPTZ DEFAULT NOW(), UNIQUE(user_id, session_date, session_start) ); CREATE
INDEX idx_study_sessions_user_date ON user_study_sessions(user_id, session_date DESC); --
===== -- Admin Analytics View
(Anonymized) -- ===== CREATE OR
REPLACE VIEW admin_user_analytics AS SELECT -- NO personally identifiable information u.id
AS anonymized_user_id, p.subscription_tier, up.level AS user_level, up.total_xp,
up.courses_completed, up.lessons_completed, up.total_study_time_minutes, up.current_streak_days,
up.avg_daily_study_minutes, -- Music element proficiency (aggregated)
ROUND((up.rhythm_mastery + up.melody_mastery + up.harmony_mastery + up.timbre_mastery +
up.texture_mastery + up.form_mastery + up.dynamics_mastery + up.tempo_mastery) / 8.0, 2) AS
avg_element_mastery, -- Engagement score (calculated) ROUND( (up.quiz_attempts_total * 0.2) +
(up.lessons_completed * 0.3) + (up.current_streak_days * 0.5) + (ucs.posts_created * 0.1) +
(ucs.reputation_points * 0.05), 2 ) AS engagement_score, -- Revenue metrics
up.subscription_revenue_generated, up.lifetime_value, -- Timestamps up.created_at AS joined_date,
up.last_study_date FROM profiles p JOIN user_progress up ON up.user_id = p.id LEFT JOIN
user_community_stats ucs ON ucs.user_id = p.id; -- Grant access to admin role GRANT SELECT ON
admin_user_analytics TO authenticated; -- Row Level Security: Only admins can read CREATE
POLICY admin_analytics_policy ON admin_user_analytics FOR SELECT USING ( EXISTS (
SELECT 1 FROM profiles WHERE id = auth.uid() AND role = 'admin' ) ); #### Progress Dashboard
(User View) ##### [NEW]&nbsp; ProgressDashboard.tsx export const ProgressDashboard: React.FC =
() => { const { user, subscription } = useAuth(); const { data: progress } = useUserProgress(user?.id);
// Require subscription for detailed analytics const hasAccess = subscription?.tier !== 'free'; if
(!hasAccess) { return ( <Card> <CardHeader> <CardTitle>Unlock Detailed Progress
Analytics</CardTitle> <CardDescription> Upgrade to see comprehensive insights, charts, and
personalized recommendations </CardDescription> </CardHeader> <CardContent> <Button
onClick={() => navigate('/subscriptions')}> Upgrade Now </Button> </CardContent> </Card> ); }
return ( <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"> { /* Study Time
Trends */} <Card className="col-span-full"> <CardHeader> <CardTitle>Study Time
Trends</CardTitle> </CardHeader> <CardContent> <ResponsiveContainer width="100%" height=

```

```

{300}> <LineChart data={progress?.studyTimeTrends}> <CartesianGrid strokeDasharray="3 3" />
<XAxis dataKey="date" /> <YAxis /> <Tooltip /> <Line type="monotone" dataKey="minutes"
stroke="#C69B36" strokeWidth={2} /> </LineChart> </ResponsiveContainer> </CardContent>
</Card> { /* Music Element Mastery Radar */} <Card> <CardHeader> <CardTitle>Music Element
Mastery</CardTitle> </CardHeader> <CardContent> <ResponsiveContainer width="100%" height=
{300}> <RadarChart data={[ { element: 'Rhythm', mastery: progress?.rhythm_mastery }, { element:
'Melody', mastery: progress?.melody_mastery }, { element: 'Harmony', mastery:
progress?.harmony_mastery }, { element: 'Timbre', mastery: progress?.timbre_mastery }, { element:
'Texture', mastery: progress?.texture_mastery }, { element: 'Form', mastery: progress?.form_mastery
}, { element: 'Dynamics', mastery: progress?.dynamics_mastery }, { element: 'Tempo', mastery:
progress?.tempo_mastery }, ]}> <PolarGrid /> <PolarAngleAxis dataKey="element" />
<PolarRadiusAxis angle={90} domain={[0, 100]} /> <Radar dataKey="mastery" stroke="#C69B36"
fill="#C69B36" fillOpacity={0.6} /> </RadarChart> </ResponsiveContainer> </CardContent>
</Card> { /* Streak Calendar Heatmap */} <Card className="col-span-2"> <CardHeader>
<CardTitle>Study Streak 🔥 {progress?.current_streak_days} days</CardTitle> </CardHeader>
<CardContent> <CalendarHeatmap startDate={subYears(new Date(), 1)} endDate={new Date()}
values={progress?.streakData} classForValue={(value) => { if (!value) return 'color-empty'; return
value.count > 60 ? 'color-scale-4' : value.count > 30 ? 'color-scale-3' : value.count > 15 ? 'color-scale-
2' : 'color-scale-1'; }} /> </CardContent> </Card> { /* XP & Level Progress */} <Card>
<CardHeader> <CardTitle>Level {progress?.level}</CardTitle> <CardDescription>
{progress?.total_xp} XP</CardDescription> </CardHeader> <CardContent> <Progress value=
{(progress?.total_xp % 1000) / 10} className="h-2" /> <p className="text-sm text-muted-
foreground mt-2"> {1000 - (progress?.total_xp % 1000)} XP to Level {progress?.level + 1} </p>
</CardContent> </Card> </div> ); }; #### Admin Analytics Dashboard Integration ####
[MODIFY]&nbsp;Admin.tsx Add new "User Analytics" tab: <TabsContent value="analytics">
<Card> <CardHeader> <CardTitle>User Progress Analytics</CardTitle>
<CardDescription>Anonymized insights into learning patterns</CardDescription> </CardHeader>
<CardContent> <UserAnalyticsCharts /> </CardContent> </Card> </TabsContent> #####
[NEW]&nbsp;UserAnalyticsCharts.tsx export const UserAnalyticsCharts: React.FC = () => { const {
data: analytics } = useAdminAnalytics(); return ( <div className="space-y-6"> { /* Engagement
Score Distribution */} <Card> <CardHeader> <CardTitle>User Engagement Distribution</CardTitle>
</CardHeader> <CardContent> <ResponsiveContainer width="100%" height={300}> <BarChart
data={analytics?.engagementDistribution}> <CartesianGrid strokeDasharray="3 3" /> <XAxis
dataKey="tier" /> <YAxis /> <Tooltip /> <Bar dataKey="count" fill="#C69B36" /> </BarChart>
</ResponsiveContainer> </CardContent> </Card> { /* Revenue by Tier */} <Card> <CardHeader>
<CardTitle>Revenue by Subscription Tier</CardTitle> </CardHeader> <CardContent>
<ResponsiveContainer width="100%" height={300}> <PieChart> <Pie data=
{analytics?.revenueByTier} dataKey="revenue" nameKey="tier" cx="50%" cy="50%" outerRadius=
{100} label > {analytics?.revenueByTier.map((entry, index) => ( <Cell key={`cell-${index}`} fill=
{COLORS[index % COLORS.length]} /> ))} </Pie> <Tooltip /> </PieChart>
</ResponsiveContainer> </CardContent> </Card> { /* Average Element Mastery Trends */} <Card>
<CardHeader> <CardTitle>Music Element Mastery (Platform Average)</CardTitle> </CardHeader>
<CardContent> <ResponsiveContainer width="100%" height={300}> <BarChart data=

```

```

{analytics?.avgElementMastery}> <CartesianGrid strokeDasharray="3 3" /> <XAxis
dataKey="element" /> <YAxis domain={[0, 100]} /> <Tooltip /> <Bar dataKey="avg_mastery"
fill="#3B82F6" /> </BarChart> </ResponsiveContainer> </CardContent> </Card> {/* Generate
Report Button */} <Button onClick={() => generateAnalyticsReport()}> <Download className="h-4
w-4 mr-2" /> Generate Full Report (PDF) </Button> </div> ); }; --- ## Verification Plan #### Database
Migration Testing # Test migrations on local/dev environment first supabase db reset supabase db
push # Verify all tables created supabase db diff # Run test queries psql $DATABASE_URL -c
"SELECT table_name FROM information_schema.tables WHERE table_schema = 'public';" # Verify
triggers psql $DATABASE_URL -c "SELECT trigger_name FROM information_schema.triggers
WHERE trigger_schema = 'public';" #### Full Application Testing Checklist 1. **Guitar
Tool**&nbsp;✓ - [ ] &nbsp;Visual assembly correct on desktop - [ ] &nbsp;Visual assembly correct
on mobile - [ ] &nbsp;Visual assembly correct on tablet - [ ] &nbsp;Chord progressions playback - [ ]
&nbsp;Settings panel functional - [ ] &nbsp;Sound quality (hard vs soft strums) 2. **Access
Control**&nbsp;✓ - [ ] &nbsp;Piano accessible to all - [ ] &nbsp;Pitch tuner accessible to all - [ ]
&nbsp;Guitar requires sign-in - [ ] &nbsp;Metronome requires sign-in - [ ] &nbsp;Mobile tab switcher
works smoothly - [ ] &nbsp;Feature request tab always accessible 3. **Learning Hub (4-Level
Hierarchy)**&nbsp;✓ - [ ] &nbsp;Navigate: Course > Module > Class > Lesson - [ ] &nbsp;Resource
library at each level - [ ] &nbsp;PDF viewer works - [ ] &nbsp;Video player works - [ ] &nbsp;Audio
player works - [ ] &nbsp;User comments functional - [ ] &nbsp;Payment customization (select
specific items) 4. **Community**&nbsp;✓ - [ ] &nbsp;Create thread with smart backlinks - [ ]
&nbsp;Post replies - [ ] &nbsp;React to posts - [ ] &nbsp;View metrics update in real-time - [ ]
&nbsp;Comment on lessons/resources 5. **Progress & Analytics**&nbsp;✓ - [ ] &nbsp;Lesson
completion updates progress - [ ] &nbsp;Quiz attempts tracked - [ ] &nbsp;Study streak calculated
correctly - [ ] &nbsp;Progress dashboard (gated by subscription) - [ ] &nbsp;Admin analytics view
(anonymized) - [ ] &nbsp;Charts render correctly - [ ] &nbsp;Report generation works --- ##
Implementation Priority Order **Phase 1: Foundation (Week 1)** 1. Database migrations (all schema
changes) 2. Update codebase references (course_modules → modules, etc.) 3. Enhanced
ProtectedRoute component **Phase 2: Quick Wins (Week 2)** 4. Guitar tool visual fixes 5. Mobile
tab switcher fix 6. Access control implementation **Phase 3: Core Features (Weeks 3-4)**&nbsp;7.
4-level lesson hierarchy navigation 8. Resource library system with modals 9. Payment customization
**Phase 4: Community (Week 5)**&nbsp;10. Forum infrastructure 11. Smart backlinks 12. Comment
systems **Phase 5: Analytics (Week 6)**&nbsp;13. Progress tracking hooks 14. User dashboard 15.
Admin analytics **Phase 6: Polish & Launch (Week 7)**&nbsp;16. Testing & bug fixes 17.
Performance optimization 18. Documentation **Target**: Ready for April school holidays! 🎓 🎵 #
Saem's Tunes App Enhancement - Task List ## 1. Analysis Phase - [ ] &nbsp;Explore current
codebase structure - [ ] &nbsp;Review backup.sql database schema - [ ] &nbsp;Identify guitar tool
components (InteractiveGuitar.tsx, OptimizedGuitar.tsx) - [ ] &nbsp;Review authentication system
(AuthContext.tsx, ProtectedRoute.tsx) - [ ] &nbsp;Analyze existing music lesson structure (courses,
course_modules, course_lessons tables) - [ ] &nbsp;Review community page implementation
(Community.tsx) - [ ] &nbsp;Examine user profile and progress tracking (profiles table,
user_progress) ## 2. Guitar Visual Fix - [ ] &nbsp;Locate guitar component files
(InteractiveGuitar.tsx) - [ ] &nbsp;Fix guitar neck positioning, fret spacing, and string alignment - [ ]
&nbsp;Verify math/logic for accurate fret frequency calculations - [ ] &nbsp;Enhance acoustics - hard

```

vs soft strums, different string sounds - [] Implement chord progressions system (similar to piano) - [] Add settings panel for customization - [] Test responsive design and visual assembly - [] Ensure guitar parts form cohesive visual image

3. Authentication Gating System

- [] Design authentication middleware/guards - [] Update ProtectedRoute for tier-based access - [] Implement protected routes for music tools:

- Piano: Free for all ✓
- Pitch Tuner: Free for all ✓
- Metronome, Guitar, others: Require sign-in

- Feature Request tab: Accessible to all - [] Fix mobile tab switcher for music tools - [] Gate learning hub resources by subscription tier - [] Gate community full features (basic features free) - [] Test authentication flows across all tiers

4. Music Lesson System (Course > Module > Class > Lesson)

- [] Rename database tables to reflect hierarchy:

- courses (existing) ✓
- course_modules → modules ✓
- course_lessons → classes ✓
- NEW: class_lessons table for lessons ✓

- [] Update all references across app to new naming - [] Design categorization (8 music elements + extended topics) - [] Implement music element filters and education level tiers (LearningHub Tabs) - [] Create resource library system (links, PDFs, audio, video, books) (Backend & UI ready) - [] Integrate library access at course/module/class levels (Lesson & Class level integrated) - [] Build course/module/class/lesson navigation with unique URLs - [] Implement payment selection: pick specific lessons/classes/modules/courses (PurchaseModal & Hooks ready) - [] Create unique payment plans per tier and quantity (Database + UI support) - [] Add modal viewers for PDFs, videos, audio, interactive content (LessonViewer & ResourceList implemented) - [] Implement user comments on lessons/resources (Backend ready, UI next) - []

5. Community Page Enhancement

- [] Design forum database schema with future real-time messaging support - [] Build infrastructure for message streaming service integration - [] Create threads/posts/comments/reactions tables - [] Track all community metrics (posts, replies, votes, etc.) - [] Link metrics to user profiles for activity tracking - [] Build forum UI with thread/post display - [] Implement smart backlinks to relevant app sections - [] Add user comment sections on lessons/resources - [] Design CommunityNavigation with live features - [] Test community features and metric tracking

6. User Profile & Progress Tracking

- [] Design comprehensive progress database with efficient table merging (Types defined) - [] Track every user activity: quizzes, lessons, tools, community (Progress tracked, Activity Logs defined) - [] Ensure site-wide updates with no redundancy or mixups - [] Build ProgressDashboard (user-specific, subscription-gated) (UserDashboard implemented) - [] Create admin analytics view (anonymized, educational insights) - [] Implement accurate mathematical/statistical progress calculations (useUserAnalytics) - [] Generate analysis reports for admin dashboard - [] Build visualizations: charts, graphs, heatmaps (Recharts integrated) - []

- Test progress persistence and accuracy across all activities - [] Integrate with existing admin dashboard

7. Documentation & Planning

- [] Create comprehensive implementation plan - [] Document database schema changes - [] Create SQL migration scripts - [] Document API changes

8. PHASE 1: Foundation (Code Complete)

- [] Create comprehensive database migration - [] Rename tables: course_modules → modules, course_lessons → classes - [] Create new class_lessons table - [] Create music_categories table with seed data - [] Create resource_libraries and library_resources tables - [] Create custom_purchases and pricing_tiers

[illegible]

-- Update column names in classes table

ALTER TABLE classes RENAME COLUMN module_id TO module_id; -- Already correct,
just verify

ALTER TABLE classes DROP COLUMN IF EXISTS content_type;

ALTER TABLE classes DROP COLUMN IF EXISTS content_url;

ALTER TABLE classes DROP COLUMN IF EXISTS is_free_preview;

-- =====

-- PART 2: CREATE NEW CLASS_LESSONS TABLE

-- =====

CREATE TABLE IF NOT EXISTS class_lessons (

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

class_id UUID NOT NULL,

title TEXT NOT NULL,

description TEXT,

lesson_type TEXT NOT NULL,

content_url TEXT,

duration_minutes INTEGER DEFAULT 0,

order_index INTEGER DEFAULT 0,

is_preview BOOLEAN DEFAULT FALSE,

-- Resource indicators

has_pdf BOOLEAN DEFAULT FALSE,

has_audio BOOLEAN DEFAULT FALSE,

has_video BOOLEAN DEFAULT FALSE,

has_quiz BOOLEAN DEFAULT FALSE,

has_exercise BOOLEAN DEFAULT FALSE,

-- Comments enabled

comments_enabled BOOLEAN DEFAULT TRUE,

```

-- Metadata

thumbnail_url TEXT,

video_platform TEXT, -- 'youtube', 'vimeo', 'direct'

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT class_lessons_lesson_type_check CHECK (

    lesson_type IN ('video', 'audio', 'text', 'interactive', 'practice', 'quiz', 'reading')

)

);

-- Add foreign key constraint (will be added after ensuring classes table exists)

DO $$

BEGIN

    IF NOT EXISTS (

        SELECT 1 FROM information_schema.table_constraints

        WHERE constraint_name = 'class_lessons_class_id_fkey'

    ) THEN

        ALTER TABLE class_lessons

        ADD CONSTRAINT class_lessons_class_id_fkey

        FOREIGN KEY (class_id) REFERENCES classes(id) ON DELETE CASCADE;

    END IF;

END $$;

CREATE INDEX IF NOT EXISTS idx_class_lessons_class ON class_lessons(class_id);

CREATE INDEX IF NOT EXISTS idx_class_lessons_order ON class_lessons(class_id,
order_index);

-- =====

-- PART 3: ENHANCED CATEGORIZATION SYSTEM

-- =====

```

-- Add music element categorization to courses

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS music_elements TEXT[] DEFAULT '{}';
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS extended_category TEXT;
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS difficulty_level INTEGER DEFAULT 1;
```

```
ALTER TABLE courses ADD COLUMN IF NOT EXISTS prerequisites UUID[] DEFAULT '{}';
```

-- Add difficulty to modules and classes

```
ALTER TABLE modules ADD COLUMN IF NOT EXISTS difficulty_level INTEGER DEFAULT 1;
```

```
ALTER TABLE modules ADD COLUMN IF NOT EXISTS prerequisites UUID[] DEFAULT '{}';
```

```
ALTER TABLE classes ADD COLUMN IF NOT EXISTS difficulty_level INTEGER DEFAULT 1;
```

-- Music categories master table

```
CREATE TABLE IF NOT EXISTS music_categories (  
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
    name TEXT NOT NULL UNIQUE,  
    category_type TEXT NOT NULL,  
    description TEXT,  
    icon TEXT,  
    color TEXT,  
    order_index INTEGER DEFAULT 0,  
    is_active BOOLEAN DEFAULT TRUE,  
    created_at TIMESTAMPTZ DEFAULT NOW(),  
    updated_at TIMESTAMPTZ DEFAULT NOW(),  
    CONSTRAINT music_categories_type_check CHECK (  
        category_type IN ('element', 'extended', 'specialty')  
    )  
);
```

);

-- Seed the 8 elements of music + extended categories

INSERT INTO music_categories (name, category_type, description, order_index) VALUES

-- 8 Elements

('Rhythm', 'element', 'Pattern of sounds and silences in music', 1),

('Melody', 'element', 'Sequence of musical notes perceived as a single entity', 2),

('Harmony', 'element', 'Combination of simultaneously sounded musical notes', 3),

('Timbre', 'element', 'Quality or color of a musical sound', 4),

('Texture', 'element', 'The way melody, harmony and rhythm combine in music', 5),

('Form', 'element', 'Structure and design of a musical composition', 6),

('Dynamics', 'element', 'Volume of sound in music', 7),

('Tempo', 'element', 'Speed at which music is played', 8),

-- Extended Categories

('Music Business', 'extended', 'Industry knowledge, marketing, and career development', 9),

('Music Production', 'extended', 'Recording, mixing, and mastering techniques', 10),

('Music History', 'extended', 'Evolution and cultural context of music', 11),

('Composition', 'extended', 'Creating original musical works', 12),

('Performance', 'extended', 'Stage presence and performance technique', 13),

('Music Theory', 'extended', 'Notation, harmony, and musical analysis', 14),

('Songwriting', 'extended', 'Crafting lyrics and melodies', 15),

('Improvisation', 'extended', 'Spontaneous musical creation', 16)

ON CONFLICT (name) DO NOTHING;

-- =====

-- PART 4: RESOURCE LIBRARY SYSTEM

-- =====

CREATE TABLE IF NOT EXISTS resource_libraries (

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

```

-- Polymorphic association

entity_type TEXT NOT NULL,

entity_id UUID NOT NULL,

name TEXT NOT NULL,

description TEXT,

icon TEXT,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT resource_libraries_entity_type_check CHECK (

    entity_type IN ('course', 'module', 'class', 'lesson')

)

);

CREATE UNIQUE INDEX IF NOT EXISTS idx_resource_libraries_entity

ON resource_libraries(entity_type, entity_id);

CREATE TABLE IF NOT EXISTS library_resources (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    library_id UUID REFERENCES resource_libraries(id) ON DELETE CASCADE,

    title TEXT NOT NULL,

    description TEXT,

    resource_type TEXT NOT NULL,

    resource_url TEXT NOT NULL,

    file_size_mb NUMERIC(10,2),

-- Metadata

author TEXT,

duration_minutes INTEGER,

page_count INTEGER,

-- Access control

```

```

access_tier TEXT DEFAULT 'free',

is_public BOOLEAN DEFAULT TRUE,

-- Usage tracking

view_count INTEGER DEFAULT 0,

download_count INTEGER DEFAULT 0,

-- Tags for searchability

tags TEXT[] DEFAULT '{}',

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT library_resources_type_check CHECK (

    resource_type IN ('pdf', 'audio', 'video', 'link', 'book', 'article', 'image', 'document', 'text')

),

CONSTRAINT library_resources_access_check CHECK (

    access_tier IN ('free', 'basic', 'premium', 'professional')

)

);

CREATE INDEX IF NOT EXISTS idx_library_resources_library ON
library_resources(library_id);

CREATE INDEX IF NOT EXISTS idx_library_resources_type ON
library_resources(resource_type);

CREATE INDEX IF NOT EXISTS idx_library_resources_tags ON library_resources USING
GIN(tags);

-- =====

-- PART 5: CUSTOM PAYMENT & PRICING SYSTEM

-- =====

CREATE TABLE IF NOT EXISTS pricing_tiers (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    entity_type TEXT NOT NULL,

```

```

entity_id UUID NOT NULL,

-- Individual pricing

single_price NUMERIC(10,2) NOT NULL,

-- Bundle pricing (quantity discounts)

bundle_5_price NUMERIC(10,2),

bundle_10_price NUMERIC(10,2),

bundle_20_price NUMERIC(10,2),

-- Subscription options

monthly_price NUMERIC(10,2),

yearly_price NUMERIC(10,2),

is_active BOOLEAN DEFAULT TRUE,

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT pricing_tiers_entity_check CHECK (

    entity_type IN ('lesson', 'class', 'module', 'course')

),

UNIQUE(entity_type, entity_id)

);

CREATE INDEX IF NOT EXISTS idx_pricing_tiers_entity ON pricing_tiers(entity_type,
entity_id);

CREATE TABLE IF NOT EXISTS custom_purchases (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

-- Purchase composition (flexible selection)

selected_lessons UUID[] DEFAULT '{}',

selected_classes UUID[] DEFAULT '{}',

selected_modules UUID[] DEFAULT '{}',

selected_courses UUID[] DEFAULT '{}',

```

-- Pricing

total_items INTEGER NOT NULL,

base_price NUMERIC(10,2) NOT NULL,

tier_discount NUMERIC(5,2) DEFAULT 0,

final_price NUMERIC(10,2) NOT NULL,

currency TEXT DEFAULT 'KES',

-- Payment details

payment_status TEXT DEFAULT 'pending',

payment_method TEXT,

transaction_id TEXT,

payment_provider TEXT, -- 'mpesa', 'paypal', 'stripe'

-- Access control

access_granted_at TIMESTAMPTZ,

access_expires_at TIMESTAMPTZ,

is_lifetime_access BOOLEAN DEFAULT FALSE,

-- Metadata

notes TEXT,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT custom_purchases_payment_status_check CHECK (

payment_status IN ('pending', 'completed', 'failed', 'refunded', 'cancelled')

)

);

CREATE INDEX IF NOT EXISTS idx_custom_purchases_user ON

custom_purchases(user_id);

CREATE INDEX IF NOT EXISTS idx_custom_purchases_status ON

custom_purchases(payment_status);

-- =====

-- PART 6: COMMUNITY INFRASTRUCTURE

-- =====

```
CREATE TABLE IF NOT EXISTS community_threads (

  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

  user_id UUID REFERENCES auth.users(id) ON DELETE SET NULL,

  title TEXT NOT NULL,

  category TEXT NOT NULL,

  tags TEXT[] DEFAULT '{}',

  -- Thread properties

  is_pinned BOOLEAN DEFAULT FALSE,

  is_locked BOOLEAN DEFAULT FALSE,

  is_announcement BOOLEAN DEFAULT FALSE,

  -- Engagement metrics

  view_count INTEGER DEFAULT 0,

  reply_count INTEGER DEFAULT 0,

  reaction_count INTEGER DEFAULT 0,

  -- Smart backlinks to learning content

  linked_course_id UUID REFERENCES courses(id) ON DELETE SET NULL,

  linked_module_id UUID REFERENCES modules(id) ON DELETE SET NULL,

  linked_class_id UUID REFERENCES classes(id) ON DELETE SET NULL,

  linked_lesson_id UUID REFERENCES class_lessons(id) ON DELETE SET NULL,

  -- Timestamps

  created_at TIMESTAMPTZ DEFAULT NOW(),

  updated_at TIMESTAMPTZ DEFAULT NOW(),

  last_activity_at TIMESTAMPTZ DEFAULT NOW(),

  CONSTRAINT community_threads_category_check CHECK (

    category IN ('general', 'theory', 'practice', 'gear', 'performance',
```

```

        'composition', 'production', 'business', 'help', 'showcase')

    )

);

CREATE INDEX IF NOT EXISTS idx_threads_category ON community_threads(category);

CREATE INDEX IF NOT EXISTS idx_threads_user ON community_threads(user_id);

CREATE INDEX IF NOT EXISTS idx_threads_activity ON
community_threads(last_activity_at DESC);

CREATE INDEX IF NOT EXISTS idx_threads_links ON
community_threads(linked_course_id, linked_module_id);

CREATE INDEX IF NOT EXISTS idx_threads_tags ON community_threads USING
GIN(tags);

CREATE TABLE IF NOT EXISTS community_posts (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    thread_id UUID REFERENCES community_threads(id) ON DELETE CASCADE,

    user_id UUID REFERENCES auth.users(id) ON DELETE SET NULL,

    parent_post_id UUID REFERENCES community_posts(id) ON DELETE CASCADE,

    content TEXT NOT NULL,

    -- Post properties

    is_solution BOOLEAN DEFAULT FALSE,

    is_edited BOOLEAN DEFAULT FALSE,

    edited_at TIMESTAMPTZ,

    -- Metrics

    reaction_count INTEGER DEFAULT 0,

    reply_count INTEGER DEFAULT 0,

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW()

);

CREATE INDEX IF NOT EXISTS idx_posts_thread ON community_posts(thread_id);

```

```

CREATE INDEX IF NOT EXISTS idx_posts_user ON community_posts(user_id);

CREATE INDEX IF NOT EXISTS idx_posts_parent ON community_posts(parent_post_id);

-- Resource/Lesson Comments

CREATE TABLE IF NOT EXISTS resource_comments (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

    -- Polymorphic reference

    commentable_type TEXT NOT NULL,

    commentable_id UUID NOT NULL,

    content TEXT NOT NULL,

    parent_comment_id UUID REFERENCES resource_comments(id) ON DELETE
    CASCADE,

    -- Metrics

    likes_count INTEGER DEFAULT 0,

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW(),

    CONSTRAINT resource_comments_type_check CHECK (

        commentable_type IN ('resource', 'lesson', 'class', 'module', 'course')

    )

);

CREATE INDEX IF NOT EXISTS idx_resource_comments_entity

ON resource_comments(commentable_type, commentable_id);

CREATE INDEX IF NOT EXISTS idx_resource_comments_user ON

resource_comments(user_id);

-- Reactions system (polymorphic)

CREATE TABLE IF NOT EXISTS reactions (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

```

```

-- Polymorphic target

reactable_type TEXT NOT NULL,

reactable_id UUID NOT NULL,

reaction_type TEXT NOT NULL,

created_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT reactions_type_check CHECK (

    reactable_type IN ('thread', 'post', 'comment', 'resource')

),

CONSTRAINT reactions_reaction_check CHECK (

    reaction_type IN ('like', 'love', 'helpful', 'inspiring', 'fire', 'celebrate')

),

UNIQUE(user_id, reactable_type, reactable_id, reaction_type)

);

CREATE INDEX IF NOT EXISTS idx_reactions_entity ON reactions(reactable_type,
reactable_id);

CREATE INDEX IF NOT EXISTS idx_reactions_user ON reactions(user_id);

-- User community statistics

CREATE TABLE IF NOT EXISTS user_community_stats (

    user_id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,

    -- Participation

    threads_created INTEGER DEFAULT 0,

    posts_created INTEGER DEFAULT 0,

    comments_created INTEGER DEFAULT 0,

    -- Engagement

    reactions_given INTEGER DEFAULT 0,

    reactions_received INTEGER DEFAULT 0,

    solutions_provided INTEGER DEFAULT 0,

```

```

-- Reputation

reputation_points INTEGER DEFAULT 0,

helpful_votes INTEGER DEFAULT 0,

-- Badges

badges TEXT[] DEFAULT '{}',

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW()

);

-- Real-time messaging infrastructure (future-ready)

CREATE TABLE IF NOT EXISTS conversations (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Participants (supports group chats)

    participant_ids UUID[] NOT NULL,

    -- Conversation metadata

    conversation_type TEXT DEFAULT 'direct',

    title TEXT,

    -- Last message info

    last_message_at TIMESTAMPTZ,

    last_message_content TEXT,

    last_message_sender_id UUID,

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW(),

    CONSTRAINT conversations_type_check CHECK (

        conversation_type IN ('direct', 'group', 'support')

    )

);

CREATE INDEX IF NOT EXISTS idx_conversations_participants

```

```

ON conversations USING GIN(participant_ids);

CREATE TABLE IF NOT EXISTS direct_messages (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    conversation_id UUID REFERENCES conversations(id) ON DELETE CASCADE,

    sender_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

    message TEXT NOT NULL,

    -- Message status

    is_read BOOLEAN DEFAULT FALSE,

    read_at TIMESTAMPTZ,

    -- Attachments

    has_attachment BOOLEAN DEFAULT FALSE,

    attachment_url TEXT,

    attachment_type TEXT,

    -- Reactions

    reactions JSONB DEFAULT '[]',

    created_at TIMESTAMPTZ DEFAULT NOW(),

    updated_at TIMESTAMPTZ DEFAULT NOW(),

    CONSTRAINT direct_messages_attachment_check CHECK (

        attachment_type IS NULL OR attachment_type IN ('image', 'audio', 'video', 'file')

    )

);

CREATE INDEX IF NOT EXISTS idx_dm_conversation ON
direct_messages(conversation_id);

CREATE INDEX IF NOT EXISTS idx_dm_sender ON direct_messages(sender_id);

CREATE INDEX IF NOT EXISTS idx_dm_unread ON direct_messages(conversation_id)
WHERE is_read = FALSE;

-- Thread subscriptions for notifications

```

```

CREATE TABLE IF NOT EXISTS thread_subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
    thread_id UUID REFERENCES community_threads(id) ON DELETE CASCADE,
    notify_on_reply BOOLEAN DEFAULT TRUE,
    notify_on_solution BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    UNIQUE(user_id, thread_id)
);

CREATE INDEX IF NOT EXISTS idx_subscriptions_user ON thread_subscriptions(user_id);

CREATE INDEX IF NOT EXISTS idx_subscriptions_thread ON
thread_subscriptions(thread_id);

-- =====

-- PART 7: COMPREHENSIVE PROGRESS TRACKING

-- =====

-- Master user progress table (if not exists, create; if exists, add columns)

DO $$

BEGIN

    CREATE TABLE IF NOT EXISTS user_progress (
        user_id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,
        created_at TIMESTAMPTZ DEFAULT NOW(),
        updated_at TIMESTAMPTZ DEFAULT NOW()
    );

END $$;

-- Add all progress columns

ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS quiz_attempts_total
INTEGER DEFAULT 0;

```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS quiz_perfect_scores  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS quiz_average_score  
NUMERIC(5,2) DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS quiz_categories_mastered  
TEXT[] DEFAULT '{}';
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS last_quiz_date  
TIMESTAMPTZ;
```

-- Learning progress (4-level hierarchy)

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS courses_enrolled INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS courses_completed  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS modules_completed  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS classes_completed INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS lessons_completed INTEGER  
DEFAULT 0;
```

-- Time tracking

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS total_study_time_minutes  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS avg_daily_study_minutes  
NUMERIC(10,2) DEFAULT 0;
```

-- Streak tracking

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS current_streak_days  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS longest_streak_days  
INTEGER DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS last_study_date DATE;
```

-- Music Elements Mastery (0-100 each)


```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS rhythm_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS melody_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS harmony_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS timbre_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS texture_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS form_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS dynamics_mastery INTEGER  
DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS tempo_mastery INTEGER  
DEFAULT 0;
```

-- Tool usage

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS tools_used TEXT[] DEFAULT  
'{}';
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS favorite_tool TEXT;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS total_tool_sessions INTEGER  
DEFAULT 0;
```

-- Achievements

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS total_xp INTEGER DEFAULT  
0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS level INTEGER DEFAULT 1;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS achievements_unlocked  
TEXT[] DEFAULT '{}';
```

-- Subscription analytics (for admin)

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS  
subscription_revenue_generated NUMERIC(10,2) DEFAULT 0;
```

```
ALTER TABLE user_progress ADD COLUMN IF NOT EXISTS lifetime_value  
NUMERIC(10,2) DEFAULT 0;
```

```
-- Detailed lesson tracking
```

```
CREATE TABLE IF NOT EXISTS user_lesson_tracking (  
  
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
  
  -- 4-level hierarchy tracking  
  
  course_id UUID REFERENCES courses(id) ON DELETE CASCADE,  
  
  module_id UUID REFERENCES modules(id) ON DELETE CASCADE,  
  
  class_id UUID REFERENCES classes(id) ON DELETE CASCADE,  
  
  lesson_id UUID REFERENCES class_lessons(id) ON DELETE CASCADE,  
  
  -- Progress  
  
  started_at TIMESTAMPTZ DEFAULT NOW(),  
  
  completed_at TIMESTAMPTZ,  
  
  last_accessed_at TIMESTAMPTZ DEFAULT NOW(),  
  
  completion_percentage INTEGER DEFAULT 0,  
  
  time_spent_minutes INTEGER DEFAULT 0,  
  
  -- Lesson-specific  
  
  video_watched_percentage INTEGER DEFAULT 0,  
  
  quiz_attempts INTEGER DEFAULT 0,  
  
  quiz_best_score INTEGER DEFAULT 0,  
  
  quiz_passed BOOLEAN DEFAULT FALSE,  
  
  -- Resources  
  
  resources_viewed UUID[] DEFAULT '{}',  
  
  resources_downloaded UUID[] DEFAULT '{}',  
  
  -- User notes  
  
  personal_notes TEXT,
```

```

bookmarked BOOLEAN DEFAULT FALSE,

difficulty_rating INTEGER,

created_at TIMESTAMPTZ DEFAULT NOW(),

updated_at TIMESTAMPTZ DEFAULT NOW(),

UNIQUE(user_id, lesson_id),

CONSTRAINT user_lesson_tracking_completion_check CHECK (
    completion_percentage >= 0 AND completion_percentage <= 100
),

CONSTRAINT user_lesson_tracking_video_check CHECK (
    video_watched_percentage >= 0 AND video_watched_percentage <= 100
),

CONSTRAINT user_lesson_tracking_rating_check CHECK (
    difficulty_rating IS NULL OR (difficulty_rating >= 1 AND difficulty_rating <= 5)
)
);

CREATE INDEX IF NOT EXISTS idx_user_lesson_user ON user_lesson_tracking(user_id);

CREATE INDEX IF NOT EXISTS idx_user_lesson_course ON
user_lesson_tracking(course_id);

CREATE INDEX IF NOT EXISTS idx_user_lesson_module ON
user_lesson_tracking(module_id);

CREATE INDEX IF NOT EXISTS idx_user_lesson_class ON
user_lesson_tracking(class_id);

CREATE INDEX IF NOT EXISTS idx_user_lesson_completed ON
user_lesson_tracking(user_id, completed_at);

-- Study session analytics

CREATE TABLE IF NOT EXISTS user_study_sessions (

    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,

```

```

session_date DATE NOT NULL,

session_start TIME NOT NULL,

session_end TIME,

duration_minutes INTEGER,

-- Activities

lessons_accessed UUID[] DEFAULT '{}',

tools_used TEXT[] DEFAULT '{}',

resources_viewed UUID[] DEFAULT '{}',

-- Engagement metrics

videos_watched INTEGER DEFAULT 0,

quizzes_taken INTEGER DEFAULT 0,

notes_written INTEGER DEFAULT 0,

forum_posts INTEGER DEFAULT 0,

created_at TIMESTAMPTZ DEFAULT NOW(),

UNIQUE(user_id, session_date, session_start)

);

CREATE INDEX IF NOT EXISTS idx_study_sessions_user_date

ON user_study_sessions(user_id, session_date DESC);

-- =====

-- PART 8: TRIGGERS & FUNCTIONS

-- =====

-- Update thread metrics on new post

CREATE OR REPLACE FUNCTION update_thread_metrics()

RETURNS TRIGGER AS $$

BEGIN

    UPDATE community_threads

    SET

```

```

        reply_count = (SELECT COUNT(*) FROM community_posts WHERE thread_id =
NEW.thread_id),

        last_activity_at = NOW(),

        updated_at = NOW()

WHERE id = NEW.thread_id;

-- Update user community stats

INSERT INTO user_community_stats (user_id, posts_created)

VALUES (NEW.user_id, 1)

ON CONFLICT (user_id) DO UPDATE

SET posts_created = user_community_stats.posts_created + 1,

    updated_at = NOW();

RETURN NEW;

END;

BEGIN

    IF NEW.completed_at IS NOT NULL AND (OLD.completed_at IS NULL OR OLD IS NULL)
THEN

        UPDATE user_progress

        SET

            lessons_completed = lessons_completed + 1,

            last_study_date = CURRENT_DATE,

            updated_at = NOW()

        WHERE user_id = NEW.user_id;

    END IF;

    RETURN NEW;

END;

DECLARE

    current_streak INTEGER := 0;

    last_date DATE;

```

check_date DATE;

BEGIN

SELECT MAX(session_date) INTO last_date

FROM user_study_sessions

WHERE user_id = p_user_id;

IF last_date IS NULL THEN

RETURN 0;

END IF;

-- Check if streak is broken (no activity yesterday or today)

IF last_date < CURRENT_DATE - INTERVAL '1 day' THEN

RETURN 0;

END IF;

check_date := last_date;

WHILE EXISTS (

SELECT 1 FROM user_study_sessions

WHERE user_id = p_user_id AND session_date = check_date

) LOOP

current_streak := current_streak + 1;

check_date := check_date - INTERVAL '1 day';

END LOOP;

RETURN current_streak;

END;

BEGIN

NEW.updated_at = NOW();

RETURN NEW;

END;

DECLARE

```

t TEXT;

BEGIN

FOR t IN

    SELECT c.table_name

    FROM information_schema.columns c

    JOIN information_schema.tables t

        ON c.table_schema = t.table_schema

        AND c.table_name = t.table_name

    WHERE c.column_name = 'updated_at'

        AND c.table_schema = 'public'

        AND t.table_type = 'BASE TABLE' -- This excludes views

LOOP

    EXECUTE format('

        DROP TRIGGER IF EXISTS set_timestamp ON %I;

        CREATE TRIGGER set_timestamp

        BEFORE UPDATE ON %I

        FOR EACH ROW

        EXECUTE FUNCTION trigger_set_timestamp();

    ', t, t);

END LOOP;

END $$;

-- =====

-- PART 9: ADMIN ANALYTICS VIEW

-- =====

CREATE OR REPLACE VIEW admin_user_analytics AS

SELECT

    -- Anonymized user identifier

```

SUBSTRING(u.id::TEXT, 1, 8) || '****' AS anonymized_user_id,

-- Subscription info

p.subscription_tier,

p.role AS user_role,

-- Progress metrics

up.level AS user_level,

up.total_xp,

up.courses_enrolled,

up.courses_completed,

up.modules_completed,

up.classes_completed,

up.lessons_completed,

up.total_study_time_minutes,

up.current_streak_days,

up.longest_streak_days,

up.avg_daily_study_minutes,

-- Music element proficiency

ROUND((

COALESCE(up.rhythm_mastery, 0) +

COALESCE(up.melody_mastery, 0) +

COALESCE(up.harmony_mastery, 0) +

COALESCE(up.timbre_mastery, 0) +

COALESCE(up.texture_mastery, 0) +

COALESCE(up.form_mastery, 0) +

COALESCE(up.dynamics_mastery, 0) +

COALESCE(up.tempo_mastery, 0)

) / 8.0, 2) AS avg_element_mastery,

-- Engagement score

ROUND(

(COALESCE(up.quiz_attempts_total, 0) * 0.2) +

(COALESCE(up.lessons_completed, 0) * 0.3) +

(COALESCE(up.current_streak_days, 0) * 0.5) +

(COALESCE(ucs.posts_created, 0) * 0.1) +

(COALESCE(ucs.reputation_points, 0) * 0.05),

2

) AS engagement_score,

-- Community engagement

COALESCE(ucs.threads_created, 0) AS threads_created,

COALESCE(ucs.posts_created, 0) AS posts_created,

COALESCE(ucs.reputation_points, 0) AS community_reputation,

-- Revenue metrics

up.subscription_revenue_generated,

up.lifetime_value,

-- Timestamps

DATE(up.created_at) AS joined_date,

up.last_study_date,

-- Activity indicators

CASE

WHEN up.last_study_date >= CURRENT_DATE - INTERVAL '7 days' THEN 'active'

WHEN up.last_study_date >= CURRENT_DATE - INTERVAL '30 days' THEN
'occasional'

ELSE 'inactive'

END AS activity_status

FROM auth.users u

```

LEFT JOIN profiles p ON p.id = u.id

LEFT JOIN user_progress up ON up.user_id = u.id

LEFT JOIN user_community_stats ucs ON ucs.user_id = u.id

WHERE u.deleted_at IS NULL;

-- Row Level Security for admin view

ALTER VIEW admin_user_analytics OWNER TO postgres;

-- =====

-- MIGRATION COMPLETE

-- =====

-- Log migration completion

DO $$

BEGIN

    RAISE NOTICE 'Comprehensive enhancement migration completed successfully!';

    RAISE NOTICE 'Tables created/updated:';

    RAISE NOTICE ' - modules (renamed from course_modules)';

    RAISE NOTICE ' - classes (renamed from course_lessons)';

    RAISE NOTICE ' - class_lessons (NEW)';

    RAISE NOTICE ' - music_categories (NEW)';

    RAISE NOTICE ' - resource_libraries, library_resources (NEW)';

    RAISE NOTICE ' - pricing_tiers, custom_purchases (NEW)';

    RAISE NOTICE ' - community_threads, community_posts, reactions (NEW)';

    RAISE NOTICE ' - user_community_stats (NEW)';

    RAISE NOTICE ' - user_progress (enhanced)';

    RAISE NOTICE ' - user_lesson_tracking (NEW)';

    RAISE NOTICE ' - user_study_sessions (NEW)';

    RAISE NOTICE 'Triggers and functions created.';

    RAISE NOTICE 'Admin analytics view created.';

```

END \$\$;

[

import React, { useState, useEffect } from 'react';

import { useParams, Link, useNavigate } from 'react-router-dom';

import MainLayout from '@components/layout/MainLayout';

import { Button } from '@components/ui/button';

import { Progress } from '@components/ui/progress';

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from '@components/ui/card';

import { Separator } from '@components/ui/separator';

import { Badge } from '@components/ui/badge';

import {

ArrowLeft, CheckCircle, Play, Clock, BookOpen, Music, Award,

ChevronDown, ChevronRight, Lock

} from 'lucide-react';

import { cn } from '@lib/utils';

import { useCourseDetails, CourseDetails } from '@hooks/useCourseDetails';

import { Collapsible, CollapsibleContent, CollapsibleTrigger } from '@components/ui/collapsible';

import LessonViewer from '@components/learning/LessonViewer';

import ResourceList from '@components/learning/ResourceList';

import PurchaseModal from '@components/payment/PurchaseModal';

import { usePayment } from '@hooks/usePayment';

const LearningModulePage = () => {

const { id } = useParams();

const navigate = useNavigate();

const { data: course, isLoading, error } = useCourseDetails(id);

const { checkAccess } = usePayment();

```

// State for navigation

const [activeModuleId, setActiveModuleId] = useState<string | null>(null);

const [activeClassId, setActiveClassId] = useState<string | null>(null);

const [activeLessonId, setActiveLessonId] = useState<string | null>(null);

const [expandedModules, setExpandedModules] = useState<Record<string, boolean>>({});

// Access Control

const [hasAccess, setHasAccess] = useState(false);

const [showPurchaseModal, setShowPurchaseModal] = useState(false);

// Check access

useEffect(() => {

  if (course) {

    if (course.access_level === 'free') {

      setHasAccess(true);

    } else {

      // Check if user bought it (mock logic in hook)

      checkAccess('course', course.id).then(setHasAccess);

    }

  }

}, [course]);

// Initialize selection when data loads

useEffect(() => {

  if (course && course.modules.length > 0) {

    if (!activeModuleId) {

      const firstModule = course.modules[0];

      setActiveModuleId(firstModule.id);

      setExpandedModules(prev => ({ ...prev, [firstModule.id]: true }));

      if (firstModule.classes.length > 0) {

```

```

        setActiveClassId(firstModule.classes[0].id);
    }
}
}
}, [course]));

const toggleModule = (moduleId: string) => {
    setExpandedModules(prev => ({
        ...prev,
        [moduleId]: !prev[moduleId]
    }));
};

if (isLoading) {
    return (

);
}

if (error || !course) {
    return (

```

Error Loading Course

The requested course could not be found or failed to load.

```
<Button onClick={() => navigate('/learning-hub')} className="mt-4">
```

Back to Learning Hub

```
);
```

```
}
```

```
// Find active content
```

```
const currentModule = course.modules.find(m => m.id === activeModuleId);
```

```
const currentClass = currentModule?.classes.find(c => c.id === activeClassId);
```

```
const currentLesson = currentClass?.lessons.find(l => l.id === activeLessonId);
```

```
return (
```

```
{/ Header /}
```

Return to Learning Hub

{course.title}

{course.description}

{course.access_level !== 'free' && (

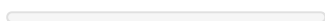
{course.access_level.toUpperCase() }

)}

{course.duration_minutes || 0} mins

{course.modules.length} Modules

Your Progress



{/ TODO: Hook up real progress /}

```
{/ Sidebar Navigation /}
```

Course Content

```
{course.modules.map((module) => (  
  <Collapsible  
    key={module.id}  
    open={expandedModules[module.id]}  
    onOpenChange={() => toggleModule(module.id)}  
    className="border-b last:border-0"  
  >
```

```
    {expandedModules[module.id] ? (  
  
      ) : (  
  
    )}
```


{module.title}

{module.classes.length} classes

{module.classes.map((cls) => (

- ```
<button
 onClick={() => {
 setActiveModuleId(module.id);
 setActiveClassId(cls.id);
 setActiveLessonId(null); // Reset lesson selection when changing
class
 }}
 className={cn(
 "w-full flex items-center gap-3 p-3 pl-11 text-sm hover:bg-muted
transition-colors text-left border-l-2 border-transparent",
 activeClassId === cls.id
 ? "bg-gold/10 border-gold text-gold-dark font-medium"
 : "text-muted-foreground"
)}
>
 {activeClassId === cls.id ? (
```

```
) : (
```

```
)}
```

```
{cls.title}
```

```
)))
```

```
)))
```

```
{/ Main Content Area /}
```

```
{currentClass ? (
```

```
<>
```

```
{currentLesson ? (
```

```
<Button
```

```
variant="ghost"
```

```
className="w-fit mb-2 pl-0 hover:pl-2 transition-all"
```

```
onClick={() => setActiveLessonId(null)}
```

```
>
```

[Back to Class Overview](#)

{currentLesson.title}

{currentClass.title}

<LessonViewer

lesson={currentLesson}

context={{

courseId: course.id,

moduleId: activeModuleId!,

classId: activeClassId!

}}

onComplete={() => setActiveLessonId(null)}

/>

): (

{currentModule?.title}

```
{currentClass.title}
```

```
{currentClass.description}
```

```
{currentClass.lessons.length > 0 ? (
```

```
{currentClass.lessons.map((lesson, idx) => (
```

```
{idx + 1}
```

```
{lesson.title}
```

```
{lesson.lesson_type}
```

```
{lesson.duration_minutes} mins
```

```
<Button
```

```
size="sm"
```

variant="secondary"

onClick={() => setActiveLessonId(lesson.id)}

>

Start

)))}

): (

No lessons available for this class yet.

)}

)}

</>

): (

Select a class from the menu to start learning

```
}}
```

```
);
```

```
};
```

```
export default LearningModulePage;]
```

```
[
```

```
import { useQuery, useMutation, useQueryClient } from "@tanstack/react-query";
```

```
import { useAuth } from "@context/AuthContext";
```

```
import { supabase } from "@integrations/supabase/client";
```

```
import { DatabaseCustom } from "@types/database.custom";
```

```
import { SupabaseClient } from "@supabase/supabase-js";
```

```
import { toast } from "sonner";
```

```
type PurchaseEntity = 'course' | 'module' | 'class';
```

```
export const usePayment = () => {
```

```
 const { user } = useAuth();
```

```
 const queryClient = useQueryClient();
```

```
 const supabaseCustom = supabase as unknown as SupabaseClient;
```

```
 // Check if user has access to a specific entity
```

```

const checkAccess = async (entityType: PurchaseEntity, entityId: string) => {

 if (!user) return false;

 // 1. Check direct purchase

 const { data: purchase } = await supabaseCustom

 .from("custom_purchases")

 .select(".*")

 .eq("user_id", user.id)

 .eq("payment_status", "completed")

 .contains(

 entityType === 'course' ? 'selected_courses' :

 entityType === 'module' ? 'selected_modules' : 'selected_classes',

 [entityId]

)

 .maybeSingle();

 if (purchase) return true;

 // 2. Check if part of a parent purchase (e.g. bought course, so has module access)

 // This is complex to check client-side without recursive queries.

 // For now, we rely on the component to check parent access or we implement a server
 function.

 // Simplified: We assume access is granted explicitly or via subscription tier (handled in
 AuthContext/Profile).

 return false;

};

// Create intent

const initiatePurchase = useMutation({

 mutationFn: async ({

 items,

 totalAmount,

```

```

 currency = 'KES'

 }: {

 items: { type: PurchaseEntity, id: string }[],

 totalAmount: number,

 currency?: string

 }) => {

 if (!user) throw new Error("User not confirmed");

 const courses = items.filter(i => i.type === 'course').map(i => i.id);

 const modules = items.filter(i => i.type === 'module').map(i => i.id);

 const classes = items.filter(i => i.type === 'class').map(i => i.id);

 const { data, error } = await supabaseCustom

 .from("custom_purchases")

 .insert({

 user_id: user.id,

 selected_courses: courses,

 selected_modules: modules,

 selected_classes: classes,

 total_items: items.length,

 base_price: totalAmount,

 final_price: totalAmount,

 currency,

 payment_status: 'pending',

 payment_method: 'mock_provider'

 })

 .select()

 .single();

 if (error) throw error;

```



```

 return data;

},

onSuccess: (data) => {

 // Typically redirect to payment gateway here

 console.log("Purchase initiated:", data.id);

}

});

// Mock confirm

const confirmPurchase = useMutation({

 mutationFn: async (purchaseId: string) => {

 const { data, error } = await supabaseCustom

 .from("custom_purchases")

 .update({

 payment_status: 'completed',

 access_granted_at: new Date().toISOString(),

 transactionid: `txn${Math.random().toString(36).substring(7)}`

 })

 .eq("id", purchaseId)

 .select()

 .single();

 if (error) throw error;

 // Also update user profile to 'basic' or 'premium' if it was a sub?

 // For valid content access, we might update enrollment.

 return data;

 },

 onSuccess: () => {

 toast.success("Purchase successful! Content unlocked.");
 }
});

```

```

 queryClient.invalidateQueries({ queryKey: ["learning-content"] });

 queryClient.invalidateQueries({ queryKey: ["course-details"] });

 }

});

return {

 initiatePurchase,

 confirmPurchase,

 checkAccess

};

};]

[

import React, { useState } from 'react';

import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogDescription, DialogFooter }
from '@components/ui/dialog';

import { Button } from '@components/ui/button';

import { Loader2, Lock, CreditCard, Check } from 'lucide-react';

import { usePayment } from '@hooks/usePayment';

interface PurchaseModalProps {

 isOpen: boolean;

 onClose: () => void;

 item: {

 id: string;

 title: string;

 type: 'course' | 'module' | 'class';

 price: number;

 description?: string;

 };

```

```
}
```

```
const PurchaseModal: React.FC = ({ isOpen, onClose, item }) => {

 const { initiatePurchase, confirmPurchase } = usePayment();

 const [step, setStep] = useState<'summary' | 'processing' | 'success'>('summary');

 const [pendingPurchaseId, setPendingPurchaseId] = useState<string | null>(null);

 const handlePurchase = () => {

 initiatePurchase.mutate({

 items: [{ type: item.type, id: item.id }],

 totalAmount: item.price

 }, {

 onSuccess: (data) => {

 setPendingPurchaseId(data.id);

 setStep('processing');

 // Simulate payment gateway delay then confirm

 setTimeout(() => {

 confirmPurchase.mutate(data.id, {

 onSuccess: () => {

 setStep('success');

 }

 });

 }, 2000);

 }

 });

 };

 const handleClose = () => {

 setStep('summary');

 setPendingPurchaseId(null);

 }

}
```

```
onClose();

};

return (
```

```
{step === 'success' ? 'Purchase Successful!' : 'Unlock Content'}
```

```
{step === 'summary' && "Review your selection and proceed to payment."}
```

```
{step === 'processing' && "Processing your secure payment..."}
```

```
{step === 'success' && "You now have full access to this content."}
```

```
{step === 'summary' && (
```

```
{item.title}
```

```
{item.type}
```

KES {item.price.toLocaleString()}}

{item.description && (

{item.description}

)}

)}

{step === 'processing' && (

Connecting to payment provider...

)}

{step === 'success' && (

Thank you for your purchase!

)}

```
{step === 'summary' && (
```

```
 {initiatePurchase.isPending ? : }
```

```
 Confirm Payment of KES {item.price.toLocaleString()
```

```
)}
```

```
{step === 'success' && (
```

```
 Start Learning
```

```
)}
```

```
 created_at?: string
```

```
 updated_at?: string
```

```
}
```

```
Update: {
```

```
 id?: string
```

```
 module_id?: string
```

```
 title?: string
```

```
 description?: string | null
```

```
 difficulty_level?: number
```

```
 order_index?: number
```

```
 created_at?: string
```

```
 updated_at?: string
```

```
}
```

```
}
```

```
class_lessons: {
 Row: {
 id: string
 class_id: string
 title: string
 description: string | null
 lesson_type: 'video' | 'audio' | 'text' | 'interactive' | 'practice' | 'quiz' | 'reading'
 content_url: string | null
 duration_minutes: number
 order_index: number
 is_preview: boolean
 has_pdf: boolean
 has_audio: boolean
 has_video: boolean
 has_quiz: boolean
 has_exercise: boolean
 comments_enabled: boolean
 thumbnail_url: string | null
 video_platform: string | null
 created_at: string
 updated_at: string
 }
 Insert: {
 id?: string
 class_id: string
 title: string
 description?: string | null
```

```
lesson_type: 'video' | 'audio' | 'text' | 'interactive' | 'practice' | 'quiz' | 'reading'

content_url?: string | null

duration_minutes?: number

order_index?: number

is_preview?: boolean

has_pdf?: boolean

has_audio?: boolean

has_video?: boolean

has_quiz?: boolean

has_exercise?: boolean

comments_enabled?: boolean

thumbnail_url?: string | null

video_platform?: string | null

created_at?: string

updated_at?: string

}
```

```
Update: {

 id?: string

 class_id?: string

 title?: string

 description?: string | null

 lesson_type?: 'video' | 'audio' | 'text' | 'interactive' | 'practice' | 'quiz' | 'reading'

 content_url?: string | null

 duration_minutes?: number

 order_index?: number

 is_preview?: boolean

 has_pdf?: boolean
```



```
 has_audio?: boolean

 has_video?: boolean

 has_quiz?: boolean

 has_exercise?: boolean

 comments_enabled?: boolean

 thumbnail_url?: string | null

 video_platform?: string | null

 created_at?: string

 updated_at?: string
 }
}

music_categories: {
 Row: {
 id: string

 name: string

 category_type: 'element' | 'extended' | 'specialty'

 description: string | null

 icon: string | null

 color: string | null

 order_index: number

 is_active: boolean

 created_at: string

 updated_at: string
 }
}

Insert: {
 id?: string

 name: string
```

```
 category_type: 'element' | 'extended' | 'specialty'

 description?: string | null

 icon?: string | null

 color?: string | null

 order_index?: number

 is_active?: boolean

 created_at?: string

 updated_at?: string
 }

 Update: {

 id?: string

 name?: string

 category_type?: 'element' | 'extended' | 'specialty'

 description?: string | null

 icon?: string | null

 color?: string | null

 order_index?: number

 is_active?: boolean

 created_at?: string

 updated_at?: string
 }
}

courses: {

 Row: {

 id: string

 title: string

 description: string | null
```

```
instructor_id: string | null

price: number | null

duration_minutes: number | null

level: 'beginner' | 'intermediate' | 'advanced' | 'professional'

access_level: 'free' | 'basic' | 'premium' | 'professional'

thumbnail_url: string | null

preview_video_url: string | null

is_published: boolean

music_elements: string[] | null

extended_category: string | null

prerequisites: string[] | null

created_at: string

updated_at: string

}
```

```
Insert: {

 id?: string

 title: string

 description?: string | null

 instructor_id?: string | null

 price?: number | null

 duration_minutes?: number | null

 level?: 'beginner' | 'intermediate' | 'advanced' | 'professional'

 access_level?: 'free' | 'basic' | 'premium' | 'professional'

 thumbnail_url?: string | null

 preview_video_url?: string | null

 is_published?: boolean

 music_elements?: string[] | null

}
```

```

 extended_category?: string | null

 prerequisites?: string[] | null

 created_at?: string

 updated_at?: string
 }

 Update: {

 id?: string

 title?: string

 description?: string | null

 instructor_id?: string | null

 price?: number | null

 duration_minutes?: number | null

 level?: 'beginner' | 'intermediate' | 'advanced' | 'professional'

 access_level?: 'free' | 'basic' | 'premium' | 'professional'

 thumbnail_url?: string | null

 preview_video_url?: string | null

 is_published?: boolean

 music_elements?: string[] | null

 extended_category?: string | null

 prerequisites?: string[] | null

 created_at?: string

 updated_at?: string
 }
},

resource_libraries: {

 Row: {

 id: string

```

```
entity_type: 'course' | 'module' | 'class' | 'lesson'
entity_id: string
name: string
description: string | null
icon: string | null
created_at: string
updated_at: string
}
```

```
Insert: {
 id?: string
 entity_type: 'course' | 'module' | 'class' | 'lesson'
 entity_id: string
 name: string
 description?: string | null
 icon?: string | null
 created_at?: string
 updated_at?: string
}
```

```
Update: {
 id?: string
 entity_type?: 'course' | 'module' | 'class' | 'lesson'
 entity_id?: string
 name?: string
 description?: string | null
 icon?: string | null
 created_at?: string
 updated_at?: string
}
```

```

 }
 },
 library_resources: {
 Row: {
 id: string
 library_id: string
 title: string
 description: string | null
 resource_type: 'pdf' | 'audio' | 'video' | 'link' | 'book' | 'article' | 'image' | 'document'
 | 'text'
 resource_url: string
 file_size_mb: number | null
 author: string | null
 duration_minutes: number | null
 page_count: number | null
 access_tier: 'free' | 'basic' | 'premium' | 'professional'
 is_public: boolean
 view_count: number
 download_count: number
 tags: string[] | null
 created_at: string
 updated_at: string
 }
 }
 Insert: {
 id?: string
 library_id: string
 title: string

```

```

description?: string | null

resource_type: 'pdf' | 'audio' | 'video' | 'link' | 'book' | 'article' | 'image' | 'document'
| 'text'

resource_url: string

file_size_mb?: number | null

author?: string | null

duration_minutes?: number | null

page_count?: number | null

access_tier?: 'free' | 'basic' | 'premium' | 'professional'

is_public?: boolean

view_count?: number

download_count?: number

tags?: string[] | null

created_at?: string

updated_at?: string
}

Update: {

 id?: string

 library_id?: string

 title?: string

 description?: string | null

 resource_type?: 'pdf' | 'audio' | 'video' | 'link' | 'book' | 'article' | 'image' |
'document' | 'text'

 resource_url?: string

 file_size_mb?: number | null

 author?: string | null

 duration_minutes?: number | null

 page_count?: number | null

```

```
 access_tier?: 'free' | 'basic' | 'premium' | 'professional'

 is_public?: boolean

 view_count?: number

 download_count?: number

 tags?: string[] | null

 created_at?: string

 updated_at?: string

 }

},

pricing_tiers: {

 Row: {

 id: string

 entity_type: 'lesson' | 'class' | 'module' | 'course'

 entity_id: string

 single_price: number

 bundle_5_price: number | null

 bundle_10_price: number | null

 bundle_20_price: number | null

 monthly_price: number | null

 yearly_price: number | null

 is_active: boolean

 updated_at: string

 }

 Insert: {

 id?: string

 entity_type: 'lesson' | 'class' | 'module' | 'course'

 entity_id: string
```



```
 single_price: number

 bundle_5_price?: number | null

 bundle_10_price?: number | null

 bundle_20_price?: number | null

 monthly_price?: number | null

 yearly_price?: number | null

 is_active?: boolean

 updated_at?: string
 }

 Update: {

 id?: string

 entity_type?: 'lesson' | 'class' | 'module' | 'course'

 entity_id?: string

 single_price?: number

 bundle_5_price?: number | null

 bundle_10_price?: number | null

 bundle_20_price?: number | null

 monthly_price?: number | null

 yearly_price?: number | null

 is_active?: boolean

 updated_at?: string
 }
},

 custom_purchases: {

 Row: {

 id: string

 user_id: string
```

```
selected_lessons: string[] | null
selected_classes: string[] | null
selected_modules: string[] | null
selected_courses: string[] | null
total_items: number
base_price: number
tier_discount: number | null
final_price: number
currency: string | null
payment_status: 'pending' | 'completed' | 'failed' | 'refunded' | 'cancelled' | null
payment_method: string | null
transaction_id: string | null
payment_provider: string | null
access_granted_at: string | null
access_expires_at: string | null
is_lifetime_access: boolean | null
notes: string | null
created_at: string
updated_at: string
}
```

```
Insert: {
 id?: string
 user_id: string
 selected_lessons?: string[] | null
 selected_classes?: string[] | null
 selected_modules?: string[] | null
 selected_courses?: string[] | null
```

```
total_items: number

base_price: number

tier_discount?: number | null

final_price: number

currency?: string | null

payment_status?: 'pending' | 'completed' | 'failed' | 'refunded' | 'cancelled' | null

payment_method?: string | null

transaction_id?: string | null

payment_provider?: string | null

access_granted_at?: string | null

access_expires_at?: string | null

is_lifetime_access?: boolean | null

notes?: string | null

created_at?: string

updated_at?: string

}
```

```
Update: {

 id?: string

 user_id?: string

 selected_lessons?: string[] | null

 selected_classes?: string[] | null

 selected_modules?: string[] | null

 selected_courses?: string[] | null

 total_items?: number

 base_price?: number

 tier_discount?: number | null

 final_price?: number

}
```

```
 currency?: string | null

 payment_status?: 'pending' | 'completed' | 'failed' | 'refunded' | 'cancelled' | null

 payment_method?: string | null

 transaction_id?: string | null

 payment_provider?: string | null

 access_granted_at?: string | null

 access_expires_at?: string | null

 is_lifetime_access?: boolean | null

 notes?: string | null

 created_at?: string

 updated_at?: string

 }

},

user_lesson_tracking: {

 Row: {

 id: string

 user_id: string

 course_id: string

 module_id: string

 class_id: string

 lesson_id: string

 started_at: string

 completed_at: string | null

 last_accessed_at: string

 completion_percentage: number

 time_spent_minutes: number

 created_at: string
```

```
 updated_at: string
 video_watched_percentage: number
 quiz_attempts: number
 quiz_best_score: number
 quiz_passed: boolean
 personal_notes: string | null
 bookmarked: boolean
 difficulty_rating: number | null
 }
```

```
Insert: {
 id?: string
 user_id: string
 course_id: string
 module_id: string
 class_id: string
 lesson_id: string
 started_at?: string
 completed_at?: string | null
 last_accessed_at?: string
 completion_percentage?: number
 time_spent_minutes?: number
 created_at?: string
 updated_at?: string
 video_watched_percentage?: number
 quiz_attempts?: number
 quiz_best_score?: number
 quiz_passed?: boolean
}
```

```
 personal_notes?: string | null
 bookmarked?: boolean
 difficulty_rating?: number | null
 }
```

```
Update: {
```

```
 id?: string
 user_id?: string
 course_id?: string
 module_id?: string
 class_id?: string
 lesson_id?: string
 started_at?: string
 completed_at?: string | null
 last_accessed_at?: string
 completion_percentage?: number
 time_spent_minutes?: number
 created_at?: string
 updated_at?: string
 video_watched_percentage?: number
 quiz_attempts?: number
 quiz_best_score?: number
 quiz_passed?: boolean
 personal_notes?: string | null
 bookmarked?: boolean
 difficulty_rating?: number | null
}
```

```
},
```

```
user_progress: {
 Row: {
 user_id: string
 courses_enrolled: number
 courses_completed: number
 modules_completed: number
 classes_completed: number
 lessons_completed: number
 total_study_time_minutes: number
 current_streak_days: number
 last_study_date: string | null
 learning_progress: Json | null
 completed_courses: string[] | null
 }
 Insert: {
 user_id: string
 courses_enrolled?: number
 courses_completed?: number
 modules_completed?: number
 classes_completed?: number
 lessons_completed?: number
 total_study_time_minutes?: number
 current_streak_days?: number
 last_study_date?: string | null
 learning_progress?: Json | null
 completed_courses?: string[] | null
 }
}
```

Update: {

user\_id?: string

courses\_enrolled?: number

courses\_completed?: number

modules\_completed?: number

classes\_completed?: number

lessons\_completed?: number

total\_study\_time\_minutes?: number

current\_streak\_days?: number

last\_study\_date?: string | null

learning\_progress?: Json | null

completed\_courses?: string[] | null

}

},

activity\_logs: {

Row: {

id: string

user\_id: string

activity\_type: 'login' | 'lesson\_start' | 'lesson\_complete' | 'quiz\_attempt' |  
'course\_enroll' | 'resource\_download'

entity\_type: string | null

entity\_id: string | null

metadata: Json | null

created\_at: string

}

Insert: {

id?: string



```

 user_id: string

 activity_type: 'login' | 'lesson_start' | 'lesson_complete' | 'quiz_attempt' |
'course_enroll' | 'resource_download'

 entity_type?: string | null

 entity_id?: string | null

 metadata?: Json | null

 created_at?: string
 }

```

```

 Update: {

 id?: string

 user_id?: string

 activity_type?: 'login' | 'lesson_start' | 'lesson_complete' | 'quiz_attempt' |
'course_enroll' | 'resource_download'

 entity_type?: string | null

 entity_id?: string | null

 metadata?: Json | null

 created_at?: string

 }

```

```

 }

```

```

}

```

```

}

```

```

}}

```

```

[

```

```

import { useQuery } from "@tanstack/react-query";

```

```

import { supabase } from "@integrations/supabase/client";

```

```

import { DatabaseCustom } from "@types/database.custom";

```

```

import { SupabaseClient } from "@supabase/supabase-js";

```

```

type ResourceLibrary = DatabaseCustom["public"]["Tables"]["resource_libraries"]["Row"];

```

```
type LibraryResource = DatabaseCustom["public"]["Tables"]["library_resources"]["Row"];
```

```
export type ResourceEntityType = 'course' | 'module' | 'class' | 'lesson';
```

```
export const useResourceLibrary = (entityType: ResourceEntityType, entityId: string | undefined) => {
```

```
 const supabaseCustom = supabase as unknown as SupabaseClient;
```

```
 // 1. Fetch the Library ID for this entity
```

```
 const { data: library, isLoading: isLibraryLoading } = useQuery({
```

```
 queryKey: ["resource-library", entityType, entityId],
```

```
 queryFn: async () => {
```

```
 if (!entityId) return null;
```

```
 const { data, error } = await supabaseCustom
```

```
 .from("resource_libraries")
```

```
 .select(".*")
```

```
 .eq("entity_type", entityType)
```

```
 .eq("entity_id", entityId)
```

```
 .maybeSingle(); // Use maybeSingle as it might not exist yet
```

```
 if (error) throw error;
```

```
 return data;
```

```
 },
```

```
 enabled: !!entityId,
```

```
 });
```

```
 // 2. Fetch Resources if library exists
```

```
 const { data: resources, isLoading: isResourcesLoading } = useQuery({
```

```
 queryKey: ["library-resources", library?.id],
```

```
 queryFn: async () => {
```

```
 if (!library?.id) return [];
```

```
 const { data, error } = await supabaseCustom
```

```

 .from("library_resources")

 .select("")

 .eq("library_id", library.id)

 .order("created_at", { ascending: false });

 if (error) throw error;

 return data;

},

 enabled: !!library?.id,

});

return {

 library,

 resources: resources || [],

 isLoading: isLibraryLoading || isResourcesLoading

};

};]

[

import React, { useState } from 'react';

import { useResourceLibrary, ResourceEntityType } from '@/hooks/useResourceLibrary';

import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';

import { Button } from '@/components/ui/button';

import { Badge } from '@/components/ui/badge';

import { FileText, Video, Music, Link as LinkIcon, Download, ExternalLink, BookOpen, File,
Loader2 } from 'lucide-react';

import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogTrigger } from
'@/components/ui/dialog';

interface ResourceListProps {

 entityType: ResourceEntityType;

 entityId: string;

```

```

 title?: string;

 className?: string;
}

const ResourceList: React.FC = ({ entityType, entityId, title = "Resources", className }) => {

 const { resources, isLoading } = useResourceLibrary(entityType, entityId);

 const [selectedResource, setSelectedResource] = useState(null);

 if (isLoading) {

 return

 }

 if (resources.length === 0) {

 return null; // Don't show anything if no resources

 }

 const getIcon = (type: string) => {

 switch (type) {

 case 'pdf': return ;

 case 'video': return

 case 'audio': return ;

 case 'link': return ;

 case 'book': return ;

 default: return ;

 }

 };

 return (

```

{title}

{resources.map((resource) => (

{getIcon(resource.resource\_type)}

**{resource.title}**

{resource.resource\_type}

{resource.file\_size\_mb} && • {resource.file\_size\_mb} MB}