Problem Statement

Education

The "Talent"

In the modern education landscape (aligned with NEP 2020), a student's value is defined not just by grades, but by their holistic skills (Sports, Arts, Tech). However, these achievements remain unorganized and unverified. Create a professional platform that allows students to discover skills, build a verifiable portfolio of their talent, and track their dedication.

Problem Statement

In the modern education landscape (aligned with NEP 2020), a student's value is defined not just by grades, but by their holistic skills (Sports, Arts, Tech). However, these achievements remain unorganized and unverified. Create a professional platform that allows students to discover skills, build a verifiable portfolio of their talent, and track their dedication.

Required Features

a. The "Skill Discovery" & Mentor Gateway : The Directory Interface: The landing page must be a structured "Activity Catalog" where students select a domain to pursue (e.g., Performing Arts > Classical Dance > Kathak or Music and other vocational talent). Mentor Matching Logic: Once a specific skill is selected, the system must display a list of Professional Mentors/Coaches. b. The "Portfolio Feed" & Showcase Wall : The Digital Portfolio: A centralized "Feed Wall" where students or mentors upload their milestones (e.g., A video of a Piano Recital, a photo of a District Medal). The solution should include a comment and like system for each post c. Role-Based Registration & Authentication: Sign-Up Logic: The application must start with a secure login/sign-up page. The Roles: It must support two distinct User Roles: Student: Mentor: Profile Management: Users should be able to set a profile picture and relavant details. d. [Open-Ended Innovation 1]: Participants must create a unique feature for this app e. [Open-Ended Innovation 2]: Participants must create a unique feature for this app

Production Plan 1

Technology Stack
Frontend: React with TypeScript, Tailwind CSS for modern UI
Backend: Supabase (PostgreSQL database, Authentication, Storage, Edge Functions)
Database: PostgreSQL via Supabase with Row Level Security (RLS)
Authentication: Supabase Auth (email/password, OAuth support)
File Storage: Supabase Storage for image/video uploads
State Management: React Context API with useReducer
API: Supabase REST API and Edge Functions for custom logic
Real-time: Supabase Realtime subscriptions

AI Services:
Google Gemini API for skill recommendations and mentor matching
Google Vision API for image/video analysis and auto-tagging
OpenAI API for content generation and student feedback

Core Features Implementation
1. Skill Discovery & Mentor Gateway
Activity Catalog (Landing Page)

Hierarchical skill structure:
Domain (e.g., Performing Arts, Sports, Technology)
Category (e.g., Classical Dance, Music, Programming)
Specific Skill (e.g., Kathak, Piano, Python)
Implementation:
client/src/pages/ActivityCatalog.tsx - Main catalog page with expandable tree structure
supabase/migrations/xxxx_create_skills.sql - Skills table with hierarchy support (parent_id for tree structure)
client/src/services/skills.ts - Supabase client queries for skills
Catalog displays as cards/tiles with search and filter functionality
AI Feature:
supabase/functions/ai-services/skill-recommendations/index.ts - Edge Function for AI skill recommendations
client/src/services/aiRecommendations.ts - Call Edge Function for personalized skill suggestions
AI analyzes student's existing skills, interests, and posts to suggest complementary skills
Mentor Matching

Implementation:
supabase/migrations/xxxx_create_mentors.sql - Mentors table with skills relationship (junction table)
supabase/migrations/xxxx_create_mentor_skills.sql - Junction table for mentor-skill relationships
client/src/pages/MentorList.tsx - Displays mentors filtered by selected skill

client/src/services/mentors.ts - Supabase queries: .from('mentors').select('*, mentor_skills(skills(*))').eq('skill_id', skillId)
supabase/functions/ai-services/mentor-matching/index.ts - Edge Function for AI-powered mentor recommendations
client/src/services/aiRecommendations.ts - Call Edge Function for AI-recommended mentors
Mentor cards show: Name, profile picture, expertise level, ratings, bio, contact options
AI Feature: Gemini API analyzes student interests, learning style, and portfolio to suggest best-matched mentors

2. Portfolio Feed & Showcase Wall
Digital Portfolio Feed
Implementation:
supabase/migrations/xxxx_create_posts.sql - Posts table with media URLs, captions, user_id, created_at
supabase/migrations/xxxx_create_post_skills.sql - Junction table for post-skill tags
client/src/pages/PortfolioFeed.tsx - Infinite scroll feed displaying posts with Supabase real-time subscriptions
client/src/components/PostCard.tsx - Individual post component with like/comment UI
Media upload: supabase.storage.from('portfolio-media').upload() for image/video uploads
Posts linked to student profiles and skills via foreign keys
AI Features:
supabase/functions/ai-services/vision-analysis/index.ts - Edge Function for Google Vision API integration
Auto-tagging: Vision API analyzes uploaded images/videos to automatically detect and suggest skill tags
Content verification: AI validates achievement claims (e.g., detecting medals, certificates, instruments in photos)
Content moderation: Vision API flags inappropriate content before publishing
supabase/functions/ai-services/caption-generation/index.ts - Edge Function for OpenAI caption suggestions
Social Interactions

Implementation:
supabase/migrations/xxxx_create_likes.sql - Likes table (user_id, post_id, created_at, unique constraint)
supabase/migrations/xxxx_create_comments.sql - Comments table (user_id, post_id, text, created_at)
client/src/services/posts.ts - Supabase queries for CRUD operations on posts
client/src/services/interactions.ts - Supabase queries: .from('likes').insert() and .from('comments').insert()
Real-time updates: supabase.channel('post-updates').on('postgres_changes', {...}) for instant like/comment feedback
RLS policies ensure users can only like/comment on visible posts
3. Role-Based Registration & Authentication

Authentication System

Implementation:
client/src/pages/Login.tsx and client/src/pages/Register.tsx - Auth UI
client/src/services/supabase.ts - Supabase client initialization with auth
Supabase Auth handles: supabase.auth.signUp(), supabase.auth.signInWithPassword(), supabase.auth.signOut()
supabase/migrations/xxxx_create_profiles.sql - Profiles table with role enum (student, mentor) linked to auth.users
Authentication tokens automatically managed by Supabase client
client/src/contexts/AuthContext.tsx - React context to access current user: supabase.auth.getUser()
Protected routes: Check user from AuthContext before rendering
Profile Management

Implementation:
client/src/pages/Profile.tsx - Profile editing page
supabase/migrations/xxxx_create_profiles.sql - Profiles table with:
id (references auth.users.id)
role (student | mentor)
avatar_url (Supabase Storage URL)
bio, location, contact_info (JSONB)
student_data (JSONB): grade, school, interests
mentor_data (JSONB): qualifications, experience, hourly_rate, availability
client/src/services/profiles.ts - Supabase queries: .from('profiles').select().eq('id', userId)
Profile picture upload: supabase.storage.from('avatars').upload() then update profile with URL
RLS policies ensure users can only update their own profile
4. Innovation Feature 1: Achievement Badge System
Skill Progress Tracking & Verification

Implementation:
supabase/migrations/xxxx_create_achievements.sql - Achievements table with:
skill_id (foreign key), user_id (foreign key)
level (beginner | intermediate | advanced | expert)
verification_status (self_verified | mentor_verified | organization_verified | ai_verified)
badge_url (Supabase Storage URL)
milestone_data (JSONB)
client/src/pages/Achievements.tsx - Student achievement dashboard
client/src/services/achievements.ts - Supabase queries for achievements
Visual progress bars and skill level indicators
Mentor can verify student achievements: Update verification_status via Supabase update
AI Features:
supabase/functions/ai-services/achievement-verification/index.ts - Edge Function for AI-powered verification

Google Vision API analyzes portfolio images to verify achievement claims (certificates, medals, performances)
Gemini API evaluates skill progression by analyzing portfolio posts over time
Automatic badge suggestions based on portfolio content analysis
Call Edge Function to trigger AI verification
Rationale: Aligns with NEP 2020's emphasis on skill verification and provides gamification to encourage continuous learning. AI adds automated verification capabilities.

5. Innovation Feature 2: Skill Event & Organization Management
Events & Workshops Hub

Implementation:
supabase/migrations/xxxx_create_events.sql - Events table with:
title, description, skill_id (foreign key)
organizer_id, organizer_type (mentor | organization)
event_date, location, capacity, registration_count
supabase/migrations/xxxx_create_event_registrations.sql - Event registrations junction table
supabase/migrations/xxxx_create_organizations.sql - Organizations table for schools/clubs
client/src/pages/Events.tsx - Browse and filter events
client/src/services/events.ts - Supabase queries for events and registrations
Students can discover and register: .from('event_registrations').insert()
AI Features:
supabase/functions/ai-services/event-recommendations/index.ts - Edge Function for personalized event recommendations
AI analyzes student's skill gaps and suggests relevant events/workshops
Call Edge Function to get AI-recommended events
Rationale: Facilitates real-world skill application, community building, and verified participation in organized activities. AI enhances discovery with personalized recommendations.

Database Schema Overview (PostgreSQL)
Core Tables:

auth.users - Supabase Auth users (managed by Supabase)
profiles - Extended user profiles (linked to auth.users.id)
skills - Skills catalog with hierarchy (parent_id for tree structure)
mentors - Mentor profiles (linked to profiles.id)
posts - Portfolio posts
achievements - Student achievements
organizations - School/club organizations
events - Events and workshops
Junction Tables:

mentor_skills - Many-to-many: mentors â†” skills
post_skills - Many-to-many: posts â†” skills (tags)

likes - Many-to-many: users â†” posts
comments - Comments on posts (one-to-many: posts â†’ comments)
event_registrations - Many-to-many: users â†” events
auth_users
profiles
mentors
posts
achievements
event_registrations
skills
mentor_skills
post_skills
likes
comments
events
organizations
has
can_be
creates
earns
registers
mentored_by
teaches
tagged_in
has_tags
receives
has
for_skill
categorized_by
organized_by
can_organize
Key Design Notes:

All tables use UUID primary keys (Supabase default)
Foreign keys reference profiles.id (which references auth.users.id)
RLS policies ensure data security at the database level
JSONB columns used for flexible data (student_data, mentor_data, milestone_data)
Timestamps (created_at, updated_at) auto-managed by Supabase
API Structure (Supabase)
Authentication (Supabase Auth)

supabase.auth.signUp() - Register (Student/Mentor) with email/password
supabase.auth.signInWithPassword() - Login
supabase.auth.signOut() - Logout

supabase.auth.getUser() - Get current authenticated user
Skills & Catalog (Supabase REST API)

supabase.from('skills').select() - Get all skills (hierarchical with parent_id)
supabase.from('skills').select().eq('id', skillId).single() - Get skill details
supabase.functions.invoke('skill-recommendations') - AI-recommended skills (Gemini API via Edge Function)
supabase.from('mentors').select('*, mentor_skills(skills(*))').eq('mentor_skills.skill_id', skillId) - Get mentors by skill
supabase.functions.invoke('mentor-matching') - AI-recommended mentors (Gemini API)
Portfolio & Posts (Supabase REST API)

supabase.from('posts').select('*, profiles(*), post_skills(skills(*))').order('created_at', {ascending: false}) - Get feed (paginated)
supabase.from('posts').insert() - Create post (with media URLs from Storage)
supabase.functions.invoke('vision-analysis') - AI analysis of uploaded media (auto-tagging, verification)
supabase.from('posts').delete().eq('id', postId) - Delete post
supabase.from('likes').upsert() - Like/unlike (upsert for toggle)
supabase.from('comments').insert() - Add comment
Profile (Supabase REST API)

supabase.from('profiles').select().eq('id', userId).single() - Get profile
supabase.from('profiles').update().eq('id', userId) - Update profile
supabase.storage.from('avatars').upload() - Upload avatar, then update profile.avatar_url
Achievements (Supabase REST API)

supabase.from('achievements').select('*, skills(*)').eq('user_id', userId) - Get user achievements
supabase.from('achievements').update({verification_status: 'mentor_verified'}).eq('id', achievementId) - Verify achievement (mentor)
supabase.functions.invoke('achievement-verification') - AI-powered achievement verification (Vision API)
Events (Supabase REST API)

supabase.from('events').select('*, skills(*)') - Browse events
supabase.functions.invoke('event-recommendations') - AI-recommended events (Gemini API)
supabase.from('events').insert() - Create event (org/mentor)
supabase.from('event_registrations').insert() - Register for event
Development Phases
Phase 1: Foundation (Setup & Auth)
Initialize project structure (React app + Supabase setup)
Set up Supabase project and get API keys
Install Supabase client: npm install @supabase/supabase-js
Set up React frontend with routing (React Router)

Create database migrations for users, profiles, and auth setup
Implement Supabase authentication (sign up, sign in, sign out)
Create AuthContext for managing user state
Build login/register pages with Supabase Auth
Set up Row Level Security (RLS) policies for profiles
Phase 2: Core Features
Implement Skill catalog structure and UI
Create Mentor matching system
Build portfolio feed with post creation
Implement like/comment functionality
Add profile management
Phase 3: Innovation Features
Build achievement badge system
Implement skill progress tracking
Create event management system
Add organization profiles
Phase 4: AI Integration
Integrate Google Vision API for image/video analysis
Set up Google Gemini API for recommendations
Implement OpenAI API for content generation
Build AI service utilities and middleware
Add auto-tagging and content moderation
Implement AI-powered mentor and skill recommendations
Phase 5: Polish & Enhancement
Configure Supabase Storage buckets (portfolio-media, avatars, badges)
Implement search and filtering using Supabase PostgREST
Add real-time notifications using Supabase Realtime
Responsive design optimization
Testing and bug fixes
AI feature optimization and error handling
Optimize RLS policies for performance
Key Files to Create
Database Migrations (Supabase)

supabase/migrations/20240101000000_create_profiles.sql
supabase/migrations/20240101000001_create_skills.sql
supabase/migrations/20240101000002_create_mentors.sql
supabase/migrations/20240101000003_create_mentor_skills.sql
supabase/migrations/20240101000004_create_posts.sql
supabase/migrations/20240101000005_create_post_skills.sql
supabase/migrations/20240101000006_create_likes.sql
supabase/migrations/20240101000007_create_comments.sql
supabase/migrations/20240101000008_create_achievements.sql
supabase/migrations/20240101000009_create_organizations.sql

supabase/migrations/20240101000010_create_events.sql
supabase/migrations/20240101000011_create_event_registrations.sql
supabase/migrations/20240101000012_setup_rls_policies.sql
Frontend Pages

client/src/pages/ActivityCatalog.tsx
client/src/pages/MentorList.tsx
client/src/pages/PortfolioFeed.tsx
client/src/pages/Profile.tsx
client/src/pages/Achievements.tsx
client/src/pages/Events.tsx
client/src/pages/Login.tsx
client/src/pages/Register.tsx
Shared Components

client/src/components/PostCard.tsx
client/src/components/MentorCard.tsx
client/src/components/SkillCard.tsx
client/src/components/Navbar.tsx
Supabase Services (Client)

client/src/services/supabase.ts - Supabase client initialization
client/src/contexts/AuthContext.tsx - Authentication context provider
client/src/services/profiles.ts - Profile queries
client/src/services/skills.ts - Skills queries
client/src/services/mentors.ts - Mentors queries
client/src/services/posts.ts - Posts queries
client/src/services/interactions.ts - Likes and comments queries
client/src/services/achievements.ts - Achievements queries
client/src/services/events.ts - Events queries
client/src/services/aiRecommendations.ts - Edge Function invocations for AI features
Supabase Edge Functions (AI Services)

supabase/functions/ai-services/skill-recommendations/index.ts - Google Gemini API for skill recommendations
supabase/functions/ai-services/mentor-matching/index.ts - Google Gemini API for mentor matching
supabase/functions/ai-services/vision-analysis/index.ts - Google Vision API for image/video analysis
supabase/functions/ai-services/caption-generation/index.ts - OpenAI API for caption generation
supabase/functions/ai-services/achievement-verification/index.ts - AI-powered achievement verification
supabase/functions/ai-services/event-recommendations/index.ts - Google Gemini API for event recommendations

AI Integration Details
Google Gemini API
Use Cases:

Skill Recommendations: Analyze student profile, portfolio, and interests to suggest relevant skills
Mentor Matching: Match students with mentors based on learning style, goals, and preferences
Event Recommendations: Suggest events based on skill gaps and learning trajectory
Achievement Analysis: Evaluate portfolio progression over time
Implementation:

Create Supabase Edge Functions for each AI use case
Install @google/generative-ai package in Edge Functions
Store API key in Supabase secrets: supabase secrets set GOOGLE_GEMINI_API_KEY=xxx
Create prompt templates for different recommendation scenarios
Edge Functions fetch user data from Supabase, call Gemini API, return recommendations
Client calls: supabase.functions.invoke('skill-recommendations', { body: { userId } })
Google Vision API
Use Cases:

Auto-tagging: Detect skills/activities in uploaded images (e.g., detecting piano, dance, sports equipment)
Achievement Verification: Analyze images to verify certificates, medals, trophies
Content Moderation: Flag inappropriate content before publication
Object Detection: Identify musical instruments, sports equipment, art supplies in photos
Implementation:

Create Supabase Edge Function: vision-analysis
Install @google-cloud/vision package in Edge Function
Set up service account JSON and store as Supabase secret
Edge Function receives image URL from Supabase Storage
Process images via Vision API, return detected objects/tags
Client uploads to Storage, then calls Edge Function for analysis
Store analysis results in post metadata (JSONB column)
OpenAI API
Use Cases:

Caption Generation: Suggest engaging captions for portfolio posts
Feedback Generation: Provide constructive feedback on student posts
Skill Description: Generate rich descriptions for skills and achievements
Learning Path Suggestions: Create personalized learning paths based on goals
Implementation:

Create Supabase Edge Function: caption-generation

Install openai package in Edge Function
Store API key in Supabase secrets: supabase secrets set OPENAI_API_KEY=xxx
Use GPT-4 or GPT-3.5 for text generation
Edge Function receives context (post content, skills), returns generated text
Client calls: supabase.functions.invoke('caption-generation', { body: { postData } })
Implement rate limiting and error handling in Edge Functions
Environment Variables Required
Client-side (.env.local)

# Supabase
VITE_SUPABASE_URL=https://your-project.supabase.co
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key
Supabase Secrets (for Edge Functions)Set via Supabase CLI:

supabase secrets set GOOGLE_GEMINI_API_KEY=your_gemini_api_key
supabase secrets set OPENAI_API_KEY=your_openai_api_key
supabase secrets set GOOGLE_VISION_CREDENTIALS='{"type":"service_account",...}'
Supabase Storage Buckets to Create

avatars - Profile pictures (public read, authenticated write)
portfolio-media - Post images/videos (public read, authenticated write)
badges - Achievement badge images (public read, admin write)
Database Setup Notes

Enable Row Level Security (RLS) on all tables
Create RLS policies for:
Users can read their own profile, update their own profile
Users can read all public posts, create/update/delete their own posts
Users can read mentors, but only mentors can update their mentor profile
Users can create likes/comments, read all likes/comments
Mentors can verify achievements
Set up foreign key relationships with CASCADE where appropriate

Platform
Enhancement Plan

Overview
This plan adds 8 major feature categories to improve user engagement, social interaction, discovery, and platform experience. Features are organized into phases where later phases depend on earlier ones.

Architecture Overview
Phase 1: Foundation

Phase 2: Social Core
Phase 3: Search & Discovery
Phase 4: Notifications
Phase 5: Portfolio Enhancements
Phase 6: Analytics
Phase 7: Mentor Features
Phase 8: Gamification
Phase 9: UX Polish

Phase 1: Foundation & Database Schema
1.1 Database Migrations
File: supabase/migrations/20240101000013_create_follows.sql

Create follows table: follower_id, following_id, created_at
RLS: Users can follow/unfollow, view all follows
Unique constraint on (follower_id, following_id)
File: supabase/migrations/20240101000014_create_notifications.sql

Create notifications table: id, user_id, type, related_user_id, related_post_id, related_event_id, message, read, created_at
Types: like, comment, follow, mention, event_reminder, achievement, mentor_message
RLS: Users can only read/update their own notifications
File: supabase/migrations/20240101000015_create_messages.sql

Create messages table: id, sender_id, receiver_id, content, read, created_at
RLS: Users can only see messages they sent/received
File: supabase/migrations/20240101000016_create_bookmarks.sql

Create bookmarks table: user_id, post_id, created_at
RLS: Users manage their own bookmarks
File: supabase/migrations/20240101000017_add_post_privacy.sql

Add privacy column to posts table: enum('public', 'followers', 'private')
Default: 'public'
Update RLS policies to respect privacy settings
File: supabase/migrations/20240101000018_create_user_stats.sql

Create user_stats view or materialized view aggregating:
Posts count, followers count, following count
Total likes received, total comments received
Achievement count, events attended
File: supabase/migrations/20240101000019_create_ratings.sql

Create ratings table: id, rater_id, rated_id, rating (1-5), review_text, created_at
For mentor ratings (mentor_id references profiles)
RLS: Users can rate mentors, view all ratings
File: supabase/migrations/20240101000020_create_bookings.sql

Create bookings table: id, student_id, mentor_id, event_id (nullable), start_time, end_time, status, notes
Status: pending, confirmed, completed, cancelled
RLS: Users can manage their own bookings
File: supabase/migrations/20240101000021_create_points_system.sql

Create user_points table: user_id, total_points, level, updated_at
Create point_transactions table: id, user_id, points, type, description, created_at
Types: post_created, achievement_earned, daily_streak, comment_received, etc.
Create function to calculate level from points
File: supabase/migrations/20240101000022_create_leaderboard.sql

Create materialized view leaderboard with:
user_id, points, rank, level, category (all_time, weekly, monthly)
Refresh strategy for performance
File: supabase/migrations/20240101000023_add_search_indexes.sql

Add PostgreSQL full-text search indexes on:
posts.caption, profiles.bio, skills.name, events.title
Create GIN indexes for JSONB columns used in search
1.2 Type Definitions
File: client/src/types/notifications.ts

Define Notification interface and notification types
File: client/src/types/social.ts

Define Follow, Message, Bookmark interfaces
File: client/src/types/analytics.ts

Define UserStats, ActivityData interfaces
File: client/src/types/mentor.ts

Define Rating, Booking, MentorStats interfaces
File: client/src/types/gamification.ts

Define Points, LeaderboardEntry, Streak interfaces
Phase 2: Social Core Features
2.1 Follow System
File: client/src/services/follows.ts

followUser(userId), unfollowUser(userId), getFollowers(userId), getFollowing(userId), isFollowing(userId)
Real-time subscriptions for follow counts
File: client/src/components/FollowButton.tsx

Reusable follow/unfollow button component
Show follower/following counts
Handle follow state with optimistic updates
File: client/src/pages/UserProfile.tsx (new page)

Public profile view showing: posts, achievements, stats, follow button
Tabs: Posts, Achievements, About
Display follower/following counts
2.2 Direct Messaging
File: client/src/services/messages.ts

sendMessage(receiverId, content), getConversations(), getMessages(userId), markAsRead(messageId)
Real-time subscriptions for new messages
File: client/src/components/Messaging.tsx

Message list and chat interface
Conversation threads, unread indicators
Send/receive messages in real-time
File: client/src/pages/Messages.tsx (new page)

Full messaging interface with conversation list and active chat
2.3 Bookmarks & Collections
File: client/src/services/bookmarks.ts

bookmarkPost(postId), unbookmarkPost(postId), getBookmarkedPosts()
Check if post is bookmarked
File: client/src/components/BookmarkButton.tsx

Bookmark icon button for posts
Save posts for later viewing
File: client/src/pages/Bookmarks.tsx (new page)

Display all bookmarked posts in grid/list view
Phase 3: Search & Discovery
3.1 Global Search
File: supabase/functions/search/index.ts (Edge Function)

Full-text search across posts, profiles, skills, events
Return ranked results by relevance
Filter by type (posts, users, skills, events)
File: client/src/services/search.ts

search(query, filters) - Call Edge Function
searchPosts(query), searchUsers(query), searchSkills(query), searchEvents(query)
File: client/src/components/SearchBar.tsx

Global search bar in navbar
Autocomplete/suggestions
Recent searches
File: client/src/pages/SearchResults.tsx (new page)

Display search results with filters
Tabs for different result types
Sorting options (relevance, date, popularity)
3.2 Advanced Filtering
File: client/src/components/FilterPanel.tsx

Reusable filter component for:
Skills, date range, media type, privacy
User roles, verification status
Applied filters display and clear
File: client/src/pages/PortfolioFeed.tsx (enhancement)

Add filter sidebar/panel
Filter posts by skills, date, media type
Save filter preferences
3.3 Trending & Popular Content
File: supabase/functions/trending/index.ts (Edge Function)

Calculate trending posts based on:
Recent likes/comments ratio
Time decay algorithm
Skill popularity
File: client/src/services/trending.ts

getTrendingPosts(timeframe), getPopularSkills(), getTrendingMentors()
File: client/src/components/TrendingPanel.tsx

Sidebar showing trending posts/skills
Refresh periodically
File: client/src/pages/Trending.tsx (new page)

Full trending content page
Tabs: Posts, Skills, Mentors
Phase 4: Notifications System
4.1 Notification Service
File: client/src/services/notifications.ts

getNotifications(), markAsRead(notificationId), markAllAsRead(), getUnreadCount()
Real-time subscriptions for new notifications
File: supabase/functions/send-notification/index.ts (Edge Function)

Trigger function called by database triggers
Send push notifications via Supabase Realtime
Queue email notifications
4.2 Notification Triggers
File: supabase/migrations/20240101000024_create_notification_triggers.sql

Database triggers on:
likes insert â†' create notification
comments insert â†' create notification
follows insert â†' create notification
posts insert with mentions â†' create notifications
Trigger function: handle_notification_creation()
4.3 Notification UI
File: client/src/components/NotificationBell.tsx

Notification icon in navbar
Badge with unread count
Dropdown with recent notifications
File: client/src/components/NotificationItem.tsx

Individual notification display
Click handlers for different notification types
Mark as read on interaction
File: client/src/pages/Notifications.tsx (new page)

Full notifications page
Filter by type, mark all as read
Infinite scroll
4.4 Push Notifications
File: client/src/services/pushNotifications.ts

Request browser notification permission
Service worker for background notifications

Handle notification clicks
File: client/public/sw.js (Service Worker)

Background notification handling
Cache notification data
Phase 5: Portfolio Enhancements
5.1 Post Editing & Management
File: client/src/services/posts.ts (enhancement)

updatePost(postId, data), deletePost(postId)
Update existing posts (caption, skills, media)
File: client/src/components/PostCard.tsx (enhancement)

Add edit/delete buttons (if user owns post)
Edit modal with existing data
Confirm delete dialog
File: client/src/components/MediaUploader.tsx (enhancement)

Support video uploads with preview
Image cropping/editing
Drag-and-drop interface
Multiple file selection with preview
5.2 Privacy Settings
File: client/src/components/PrivacySelector.tsx

Dropdown for post privacy: Public, Followers Only, Private
Show privacy icon on posts
File: client/src/pages/PortfolioFeed.tsx (enhancement)

Filter posts by privacy in create/edit modal
Respect privacy when displaying posts
5.3 Video Player
File: client/src/components/VideoPlayer.tsx

Custom video player with controls
Thumbnail generation
Playback controls, fullscreen
File: client/src/components/MediaGallery.tsx

Lightbox for image galleries
Swipe navigation, zoom
Video playback in gallery
5.4 Portfolio Pages
File: client/src/pages/UserPortfolio.tsx (new page)

Public portfolio view for any user
Grid layout of posts
Filter by skills, date range
Share portfolio link
Phase 6: Analytics & Progress Tracking
6.1 User Statistics
File: client/src/services/analytics.ts

getUserStats(userId), getActivityData(userId, timeframe), getSkillProgress(userId)
Aggregate data from database
File: client/src/components/StatsCard.tsx

Display stat cards: Posts, Followers, Following, Likes, Achievements
Animated counters
File: client/src/pages/Analytics.tsx (new page)

Comprehensive analytics dashboard
Charts for: Post frequency, Skill progress, Engagement over time
Activity calendar heatmap
6.2 Progress Tracking
File: client/src/components/ProgressChart.tsx

Chart component using Chart.js or Recharts
Display skill progression over time
Achievement milestones
File: client/src/components/ActivityCalendar.tsx

GitHub-style activity calendar
Show posting frequency, achievements earned
Clickable dates to see activity
6.3 Portfolio Export
File: supabase/functions/export-portfolio/index.ts (Edge Function)

Generate PDF portfolio from user data
Include posts, achievements, stats
Return download link
File: client/src/services/export.ts

exportPortfolio(userId, format) - Trigger PDF generation
File: client/src/components/ExportButton.tsx

Button to export portfolio as PDF
Loading state during generation

Phase 7: Mentor Features
7.1 Rating & Review System
File: client/src/services/ratings.ts

submitRating(mentorId, rating, review), getMentorRatings(mentorId), getAverageRating(mentorId)
Check if user has rated mentor
File: client/src/components/RatingStars.tsx

Star rating display component
Editable rating input
File: client/src/components/MentorReviews.tsx

Display mentor reviews and ratings
Average rating, review count
Sort by date/rating
File: client/src/pages/MentorProfile.tsx (enhancement)

Add ratings section
Display average rating prominently
7.2 Booking System
File: client/src/services/bookings.ts

createBooking(mentorId, startTime, endTime), getBookings(userId), updateBookingStatus(bookingId, status)
getMentorAvailability(mentorId, date) - Check available slots
File: client/src/components/BookingCalendar.tsx

Calendar component for selecting booking times
Show available/unavailable slots
Time slot selection
File: client/src/components/BookingModal.tsx

Modal for creating bookings
Date/time picker, notes field
Confirmation flow
File: client/src/pages/Bookings.tsx (new page)

Manage bookings (student and mentor views)
Upcoming, past, cancelled bookings
Mentor: accept/reject pending bookings
7.3 Video Calls Integration
File: client/src/services/videoCalls.ts

Integrate with Zoom/Daily.co/Jitsi API
createMeetingRoom(bookingId), getMeetingLink(bookingId)
Generate meeting links for bookings
File: client/src/components/VideoCallButton.tsx

Button to join video call
Show meeting link for scheduled sessions
7.4 Mentor Verification
File: supabase/migrations/20240101000025_add_mentor_verification.sql

Add verified boolean to profiles table
Add verification_documents JSONB column
Admin-only update for verification status
File: client/src/components/VerificationBadge.tsx

Verified badge icon for verified mentors
Display on mentor cards and profiles
Phase 8: Gamification
8.1 Points System
File: client/src/services/points.ts

getUserPoints(userId), getPointHistory(userId), calculateLevel(points)
Award points on actions (via database triggers)
File: client/src/components/PointsDisplay.tsx

Show user's current points and level
Progress bar to next level
Display in navbar or profile
File: supabase/migrations/20240101000026_create_points_triggers.sql

Database triggers to award points:
Post created: +10 points
Achievement earned: +50 points
Daily streak: +5 points per day
Comment received: +2 points
Follow gained: +3 points
8.2 Streaks
File: client/src/services/streaks.ts

getCurrentStreak(userId), getLongestStreak(userId), checkDailyPost()
Calculate posting streaks
File: client/src/components/StreakDisplay.tsx

Fire icon with streak count

Motivational messages
Warning when streak is about to break
8.3 Leaderboards
File: client/src/services/leaderboard.ts

getLeaderboard(category, timeframe), getUserRank(userId, category)
Categories: All-time, Weekly, Monthly, By Skill
File: client/src/components/LeaderboardCard.tsx

Display top users in category
Show rank, points, level
Click to view profile
File: client/src/pages/Leaderboard.tsx (new page)

Full leaderboard page
Tabs for different categories
Search user rank
8.4 Achievement Enhancements
File: client/src/components/AchievementShowcase.tsx

Enhanced achievement display
Badge animations on unlock
Achievement progress indicators
File: client/src/pages/Achievements.tsx (enhancement)

Group achievements by skill
Filter by verification status
Share achievements
Phase 9: UX Improvements
9.1 Dark Mode
File: client/src/contexts/ThemeContext.tsx

Theme context for dark/light mode
Persist preference in localStorage
File: client/tailwind.config.js (enhancement)

Add dark mode configuration
Dark theme colors
File: client/src/components/ThemeToggle.tsx

Toggle button for dark/light mode
Add to navbar
File: Update all components with dark mode classes

Add dark: variants to Tailwind classes
9.2 Loading States & Skeletons
File: client/src/components/SkeletonLoader.tsx

Reusable skeleton component
Skeleton variants: Post, Profile, Card
File: Update PortfolioFeed.tsx, ActivityCatalog.tsx, etc.

Replace loading spinners with skeletons
Improve perceived performance
9.3 Error Boundaries
File: client/src/components/ErrorBoundary.tsx

React error boundary component
Display friendly error messages
Log errors to monitoring service
File: client/src/App.tsx (enhancement)

Wrap routes with ErrorBoundary
9.4 Optimistic UI Updates
File: client/src/hooks/useOptimisticUpdate.ts

Custom hook for optimistic updates
Rollback on error
File: Update like/comment/follow actions

Use optimistic updates for instant feedback
9.5 Infinite Scroll Optimization
File: client/src/hooks/useInfiniteScroll.ts

Custom hook for infinite scroll
Intersection Observer API
Load more posts on scroll
File: client/src/pages/PortfolioFeed.tsx (enhancement)

Implement infinite scroll
Virtual scrolling for large lists (react-window)
9.6 Image Optimization
File: client/src/utils/imageOptimization.ts

Image compression before upload
Lazy loading utility
Generate thumbnails
File: Update PostCard.tsx, MediaUploader.tsx

Implement lazy loading
Show blur placeholder while loading
9.7 Mobile Improvements
File: Update all pages/components

Ensure mobile-responsive layouts
Touch-friendly buttons and interactions
Bottom navigation for mobile
Swipe gestures for posts
File: client/src/components/MobileNav.tsx

Bottom navigation bar for mobile
Quick access to main features
9.8 Keyboard Shortcuts
File: client/src/hooks/useKeyboardShortcuts.ts

Custom hook for keyboard shortcuts
Shortcuts: / (search), n (new post), Esc (close modals)
File: client/src/components/ShortcutsModal.tsx

Modal showing available keyboard shortcuts
Access via ? key
9.9 Performance Monitoring
File: client/src/utils/performance.ts

Performance monitoring utilities
Track page load times, API call durations
Log slow operations
Implementation Order
Phase 1: Database schema (all migrations)
Phase 2: Social features (follow, messaging, bookmarks)
Phase 3: Search & discovery
Phase 4: Notifications
Phase 5: Portfolio enhancements
Phase 6: Analytics
Phase 7: Mentor features
Phase 8: Gamification
Phase 9: UX polish (can be done in parallel with other phases)
Key Files to Create/Modify
New Database Migrations: 14 migration files

New Services: 15+ service files

New Components: 30+ component files

New Pages: 10+ page files

New Hooks: 5+ custom hooks

Edge Functions: 5+ new functions

Enhanced Files: Update 20+ existing files

Dependencies to Add

```
{
  "recharts": "^2.10.0", // Charts for analytics
  "react-window": "^1.8.10", // Virtual scrolling
  "react-intersection-observer": "^9.5.0", // Infinite scroll
  "react-hot-toast": "^2.4.0", // Toast notifications
  "date-fns": "^2.30.0", // Date utilities
  "react-player": "^2.13.0", // Video player
  "react-image-gallery": "^1.3.0", // Image gallery
  "jspdf": "^2.5.1" // PDF generation
}
```