BÁO CÁO đồ án giữa kì SKILLGAP AI

Khóa học: Computational Thinking - 24C06
Nhóm: Fantastic Five

Giảng viên hướng dẫn

- Bùi Duy Đăng
- Huỳnh Lâm Hải Đăng
- Nguyễn Thanh Tình

Thành viên nhóm

MSSV	Họ và tên
20127552	Vương Huỳnh Tấn Lộc
21127738	Trần Bảo Ngọc
23127465	Nguyễn Hồ Anh Quốc
23127518	Trần Quốc Việt
24127244	Phạm Tấn Nhật Thịnh

Ngày báo cáo: 12/12/2025
Dự án: SkillGapAI - Hệ thống phân tích khoảng cách kỹ năng IT
Công nghệ: React, TypeScript, Supabase, AI/ML

MỤC LỤC

1. PHÂN CÔNG CÔNG VIỆC NHÓM

1.1. Tổng quan phân công

Dự án SkillGapAI được phát triển bởi nhóm **Fantastic Five** với sự phân công công việc rõ ràng, đảm bảo mỗi thành viên đóng góp vào các phần khác nhau của hệ thống.

1.2. Chi tiết phân công

Phạm Tấn Nhật Thịnh (24127244) - Team Leader & Full-stack Developer

Trách nhiệm chính:

- 🎯 **Quản lý dự án:** Điều phối công việc, theo dõi tiến độ, tổ chức họp nhóm
- 📐 **Kiến trúc hệ thống:** Thiết kế overall architecture, database schema
- ⚙️ **Backend Development:**
  - Thiết kế và implement Supabase database schema
  - Phát triển Edge Functions: `analyze-skills` , `compare-skills`
  - Thiết lập Row Level Security (RLS) policies
  - Configuration cho Supabase Storage bucket
- 📊 **Data Analysis:**
  - Thu thập và xử lý market skills data (1491 job postings)
  - Thiết kế skill mapping dictionary (80+ mappings)
- 📝 **Documentation:** Viết báo cáo phân tích theo Computational Thinking

Deliverables:

- Database migrations (8 files)
- Edge Functions: analyze-skills, compare-skills
- Market data: market_skills.csv, skills_config.py
- Báo cáo kỹ thuật chi tiết

Nguyễn Hồ Anh Quốc (23127465) - AI/ML Specialist & Backend Developer

Trách nhiệm chính:

- 🤖 **AI Integration:**
  - Tích hợp Gemini 2.5 Flash API
  - Phát triển Edge Function: `ocr-transcript` (OCR vision AI)
  - Phát triển Edge Function: `chat` (AI chatbot streaming)
  - Fine-tuning AI prompts cho skill extraction
- 🔍 **Natural Language Processing:**
  - Thiết kế text normalization algorithms
  - Implement multilingual support (Vietnamese/English)
- 📷 **Image Processing:**
  - Xử lý multiple image formats (JPG, PNG, WebP, HEIC)
  - Base64 encoding/decoding
- 💬 **Chatbot Development:**
  - Server-Sent Events (SSE) streaming implementation
  - Context-aware conversation logic

Deliverables:

- Edge Functions: ocr-transcript, chat, get-job-skills
- AI prompt engineering documentation
- OCR accuracy testing report

Trần Bảo Ngọc (21127738) - Frontend Developer & UI/UX Designer

Trách nhiệm chính:

- 🎨 **UI/UX Design:**
  - Thiết kế wireframes và mockups
  - Component library setup (shadcn/ui)
  - Responsive design implementation
- ⚛️ **Frontend Development:**
  - Phát triển React components:
    - Pages: Index.tsx, Upload.tsx, Results.tsx
    - Components: SkillGapChart.tsx, FeatureCard.tsx
  - Routing setup với React Router v6

    - State management với React Query
- 📈 **Data Visualization:**
    - Implement Recharts bar charts
    - Design metrics cards và progress indicators
- 🎯 **Landing Page:**
    - Hero section với animations
    - Features showcase
    - "How It Works" section

Deliverables:

- UI Components: 10+ custom components
- Pages: Index, Upload, Results, NotFound
- Tailwind CSS configuration
- Design system documentation

Trần Quốc Việt (23127518) - Frontend Developer & Authentication Specialist

Trách nhiệm chính:

- 🔐 **Authentication System:**
    - Implement Supabase Auth integration
    - Phát triển pages: Auth.tsx, ForgotPassword.tsx, Profile.tsx
    - Protected routes logic
    - JWT token management
- 👤 **User Management:**
    - User profile management (Personalize.tsx)
    - User preferences storage
    - Email verification flow
- 📱 **User Experience:**
    - Form validation với React Hook Form + Zod
    - Error handling và user feedback (toasts)
    - Loading states và progress tracking
- 📜 **History & Analytics:**
    - Phát triển History.tsx page
    - Analysis records management
    - Delete confirmation dialogs

Deliverables:

- Pages: Auth, Profile, Personalize, History, ForgotPassword
- Form validation schemas
- Authentication flow documentation
- User journey mapping

Vương Huỳnh Tấn Lộc (20127552) - DevOps & Quality Assurance

Trách nhiệm chính:

- 🚀 **Deployment & CI/CD:**
  - Vite build configuration
  - Environment variables setup
  - Lovable platform deployment
  - Domain và hosting configuration
- 🧪 **Testing & Quality Assurance:**
  - Manual testing cho tất cả features
  - Cross-browser compatibility testing
  - Mobile responsiveness testing
  - Bug tracking và reporting
- 📦 **Package Management:**
  - Dependencies management (package.json)
  - Version control với Git
  - Code review và merge requests
- 🔧 **Development Tools:**
  - ESLint configuration
  - TypeScript configuration (tsconfig.json)
  - PostCSS và Tailwind setup
- 🤝 **Integration Support:**
  - Chatbot UI component (ChatBot.tsx)
  - TopNav component với auth state
  - Toast notifications system

Deliverables:

- Build configuration files
- Testing reports

- Deployment documentation
- Git workflow guidelines

1.3. Công cụ cộng tác

Công cụ	Mục đích
Git/GitHub	Version control, code collaboration
Lovable	Project management, deployment
Supabase	Backend infrastructure
Discord/Telegram	Team communication
Figma	UI/UX design collaboration
Google Docs	Documentation sharing

1.4. Timeline

Phase	Duration	Deliverables
Planning & Design	Week 1-2	Architecture design, UI mockups, database schema
Backend Development	Week 3-4	Database setup, Edge Functions, AI integration
Frontend Development	Week 5-6	React components, pages, routing
Integration & Testing	Week 7	E2E testing, bug fixes, optimization
Deployment & Documentation	Week 8	Production deployment, final report

1.5. Meetings & Sprints

Weekly Schedule:

- **Monday:** Sprint planning, task assignment

- **Wednesday:** Mid-week sync, code review
- **Friday:** Sprint review, demo, retrospective

Communication Protocol:

- Daily updates qua group chat
- Blocking issues → immediate team call
- Code review required trước khi merge
- Documentation cập nhật liên tục

2. TỔNG QUAN DỰ ÁN

2.1. Mục đích

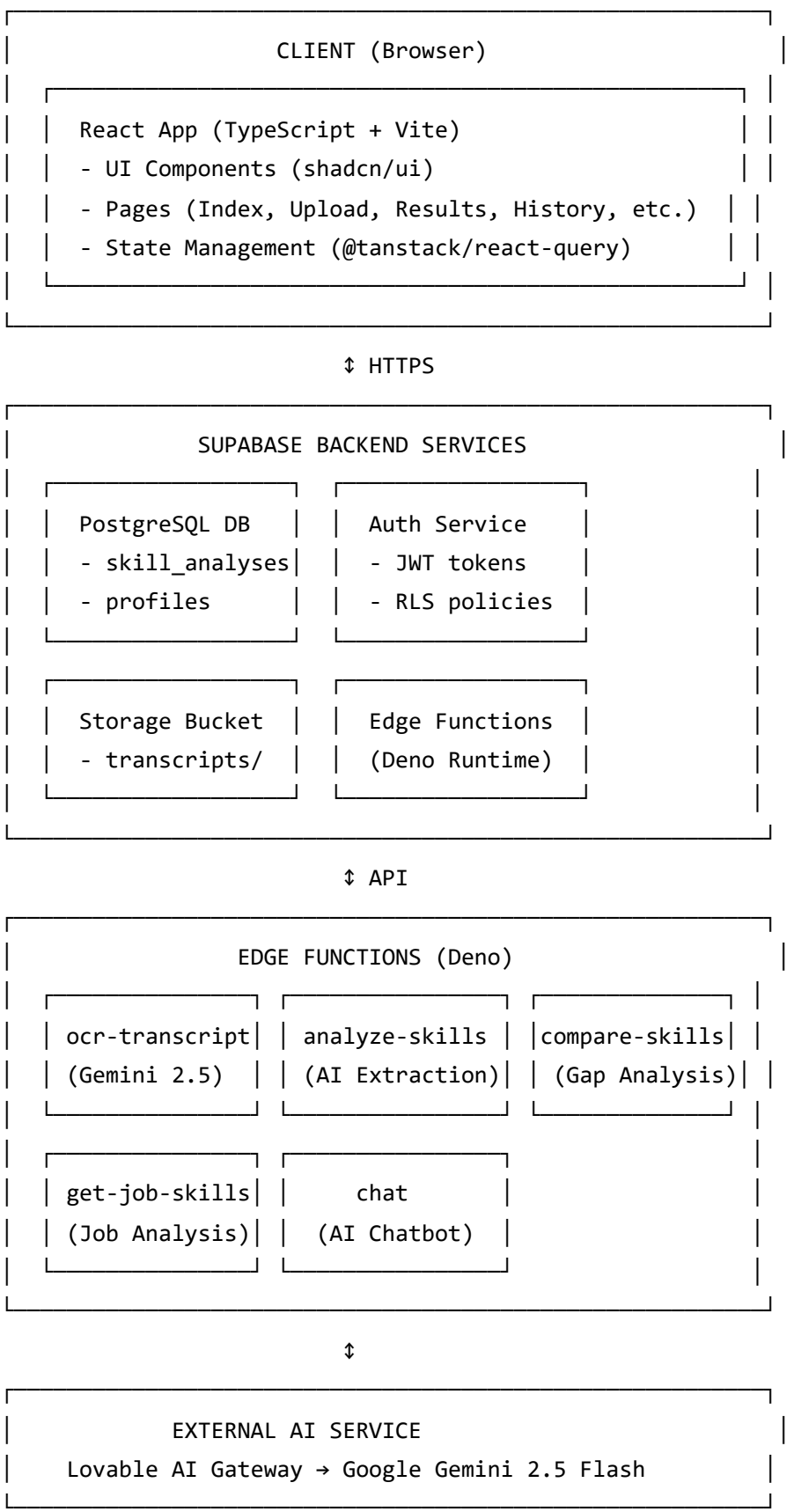
SkillGapAI là một ứng dụng web thông minh sử dụng AI để:

- Phân tích bảng điểm/hồ sơ học tập của sinh viên IT
- So sánh kỹ năng hiện tại với nhu cầu thị trường việc làm
- Đưa ra lộ trình học tập cá nhân hóa để thu hẹp khoảng cách kỹ năng

2.2. Công nghệ sử dụng

Lớp	Công nghệ	Mục đích
Frontend	React 18.3, TypeScript	Giao diện người dùng tương tác
UI Framework	shadcn/ui, Radix UI, Tailwind CSS	Hệ thống thiết kế component
Backend	Supabase (PostgreSQL)	Database & Authentication
Serverless Functions	Deno Edge Functions	Xử lý logic backend
AI/ML	Gemini 2.5 Flash (Google)	OCR, phân tích kỹ năng, chatbot
Data Visualization	Recharts	Biểu đồ phân tích kỹ năng
Routing	React Router v6	Điều hướng SPA

2.3. Kiến trúc tổng quan



3. BƯỚC 1: PHÂN RÃ VẤN ĐỀ (Decomposition)

3.1. Vấn đề chính

"Làm thế nào để giúp sinh viên IT nhận biết và thu hẹp khoảng cách giữa kỹ năng hiện tại và yêu cầu thị trường việc làm?"

3.2. Phân rã thành các vấn đề con

Module 1: Xác thực và quản lý người dùng

- **File liên quan:** Auth.tsx, Profile.tsx, ForgotPassword.tsx
- **Chức năng:**
  - Đăng ký/Đăng nhập người dùng
  - Xác thực email
  - Khôi phục mật khẩu
  - Quản lý profile
- **Công nghệ:** Supabase Auth với JWT tokens, Row Level Security (RLS)

Module 2: Cá nhân hóa thông tin người dùng

- **File liên quan:** Personalize.tsx
- **Chức năng:**
  - Thu thập ngôn ngữ lập trình chính (13 options)
  - Thu thập công việc mơ ước (14 IT roles)
  - Lưu preferences vào database
- **Database:** Bảng `profiles` với trường `main_language` và `dream_job`

Module 3: Upload và xử lý tài liệu

- **File liên quan:** Upload.tsx, ocr-transcript/index.ts
- **Chức năng:**
  - Upload nhiều định dạng:
    - Documents: PDF, DOCX, TXT, CSV
    - Images: JPG, PNG, WebP, HEIC
  - Kiểm tra file size (max 10MB)
  - Lưu trữ an toàn trong Supabase Storage
  - OCR cho ảnh sử dụng Gemini 2.5 Flash Vision
- **Workflow:**

```
User Upload → Validation → Storage Upload → OCR (if image) → Text Extraction
```

Module 4: Trích xuất kỹ năng từ văn bản

- **File liên quan:** analyze-skills/index.ts
- **Chức năng:**
  - Mapping đa ngôn ngữ (Vietnamese/English)
  - Ánh xạ môn học → kỹ năng IT
  - Hỗ trợ 80+ skill mappings
- **Ví dụ mapping:**

```
"c/c++": ["c++", "c", "cpp", "lập trình c", "lập trình c++"]
"java": ["java", "spring", "spring boot", "hibernate", "lập trình java"]
"machine learning": ["machine learning", "ml", "học máy", "deep learning"]
```

- **AI Processing:**
  - Sử dụng Gemini 2.5 Flash để hiểu context
  - Trích xuất explicit skills (tên môn học)
  - Trích xuất implicit skills (công nghệ ẩn trong mô tả)

Module 5: So sánh với thị trường việc làm

- **File liên quan:** compare-skills/index.ts, get-job-skills/index.ts
- **Chức năng:**
  - Load market data từ CSV (1491 job postings)
  - Hoặc query AI về skill requirements của dream job cụ thể
  - Tính toán:
    - **Match Percentage**: % kỹ năng trùng khớp với top 20 market skills
    - **Market Readiness**: % sẵn sàng cho thị trường (có bonus nếu > 15 skills)
    - **Skill Gaps**: Top 10 kỹ năng còn thiếu, xếp theo % demand
- **Formula:**

```
Match % = (Matched Skills / Top 20 Market Skills) × 100
Market Readiness = min(95, Match% + (bonus if skills > 15 ? 10 : 0))
```

Module 6: Hiển thị kết quả và phân tích

- **File liên quan:** Results.tsx, SkillGapChart.tsx
- **Chức năng:**
  - Biểu đồ so sánh (Recharts Bar Chart):

        - Your Level (màu primary)
        - Market Demand (màu accent)
    - 3 Key Metrics Cards:
        - Match Score
        - Market Readiness
        - Critical Gaps
    - Danh sách chi tiết:
        - Current Skills (badges)
        - Priority Learning Roadmap (ranked gaps)
    - Phân loại gaps: Critical (top 3) vs Important

Module 7: Lịch sử phân tích

- **File liên quan:** History.tsx
- **Chức năng:**
    - Hiển thị tất cả analyses của user
    - Thông tin: file name, date, match %, readiness %
    - Actions: View details, Delete
    - AlertDialog xác nhận delete

Module 8: AI Chatbot hỗ trợ

- **File liên quan:** ChatBot.tsx, chat/index.ts
- **Chức năng:**
    - Trợ lý AI streaming (SSE - Server-Sent Events)
    - Kiến thức về website:
        - Hướng dẫn sử dụng
        - Giải thích features
        - Career guidance
    - Context-aware về navigation và user flow

Module 9: Landing Page và Navigation

- **File liên quan:** Index.tsx, TopNav.tsx
- **Chức năng:**
    - Hero section với CTA
    - "How It Works" (3 steps)
    - Sample skill gap chart
    - Features showcase
    - Top navigation với auth state

- Mobile responsive

4. BƯỚC 2: NHẬN DẠNG MẪU (Pattern Recognition)

4.1. Patterns trong kiến trúc

Pattern 1: Component-Based Architecture

- **Mô tả:** Chia UI thành các component độc lập, tái sử dụng
- **Ví dụ:**

```
src/components/
├── ui/                (40+ shadcn components)
│   ├── button.tsx
│   ├── card.tsx
│   ├── dialog.tsx
│   └── ...
├── ChatBot.tsx        (Complex feature component)
├── SkillGapChart.tsx (Data visualization component)
├── FeatureCard.tsx    (Presentation component)
└── TopNav.tsx         (Layout component)
```

- **Lợi ích:** Maintainability, reusability, testability

Pattern 2: Serverless Function Pattern

- **Mô tả:** Mỗi chức năng backend là một Edge Function riêng biệt
- **Cấu trúc:**

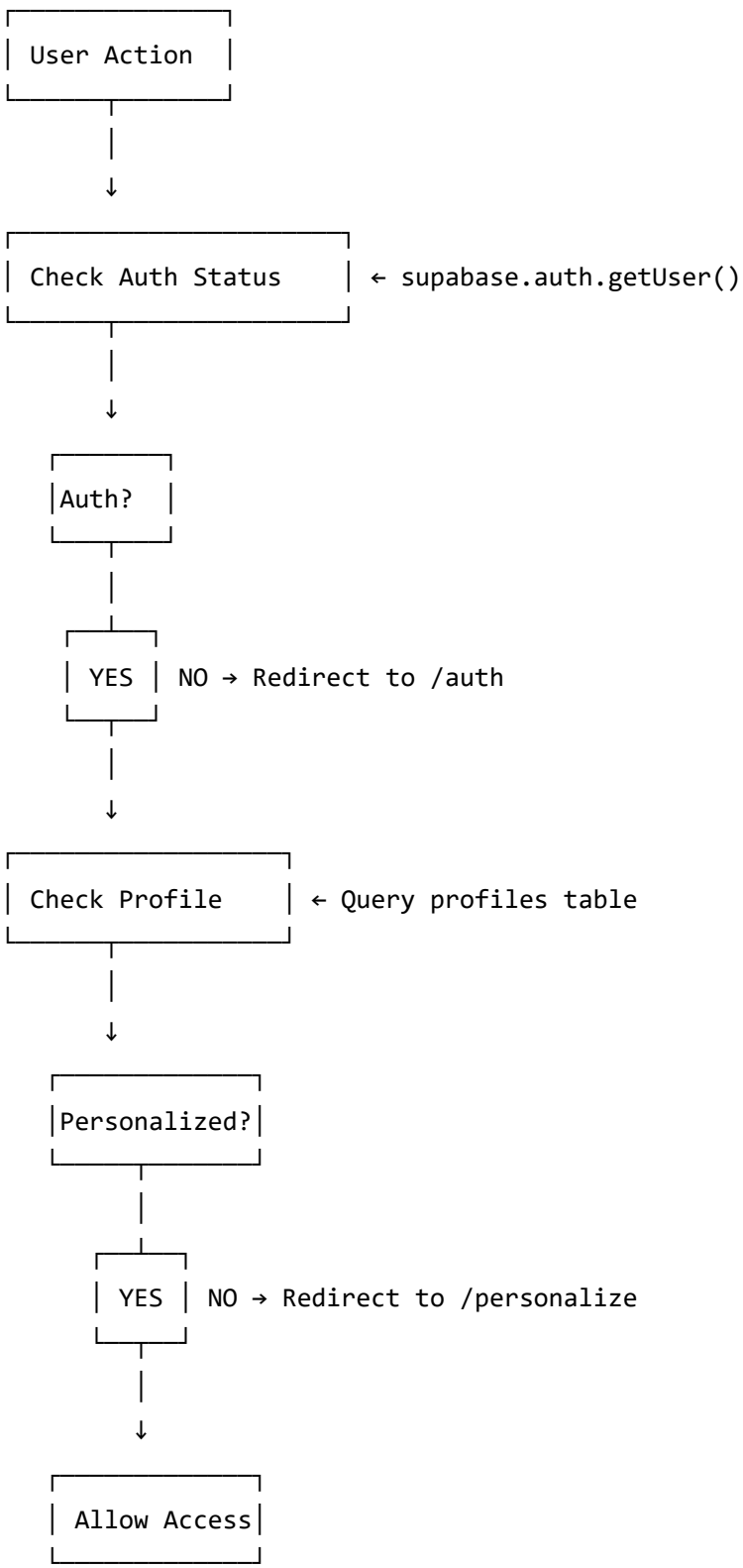
```
// Pattern template cho mọi function
const corsHeaders = { /* CORS config */ };

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') {
    return new Response(null, { headers: corsHeaders });
  }

  try {
    const { param1, param2 } = await req.json();
    // Business logic
    const result = await processData(param1, param2);
    return new Response(JSON.stringify(result), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' }
    });
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      status: 500,
      headers: { ...corsHeaders, 'Content-Type': 'application/json' }
    });
  }
});
```

- **Functions:**
  - `ocr-transcript` : Image → Text
  - `analyze-skills` : Text → Skills Array
  - `compare-skills` : Skills → Gap Analysis
  - `get-job-skills` : Job Title → Required Skills
  - `chat` : Messages → AI Response Stream

Pattern 3: Authentication Flow Pattern



Áp dụng ở: Upload.tsx, History.tsx, Profile.tsx

Pattern 4: Data Processing Pipeline

```
Input (File/Image)
  → Extract Text (OCR if needed)
    → Normalize Text (lowercase, remove special chars)
      → Match Skills (SKILL_MAPPING dictionary)
        → Deduplicate (Set)
          → Return Array
```

File: analyze-skills/index.ts

Pattern 5: Progressive Enhancement

- Upload page có 2 tabs: Document và Image
- Tự động detect file type và xử lý phù hợp
- Fallback mechanisms: Nếu get-job-skills fail → dùng CSV data

Pattern 6: Error Handling Pattern

```
try {
  // Main operation
  const result = await riskyOperation();
  return success(result);
} catch (error) {
  console.error('Context:', error);
  toast({
    title: "User-friendly title",
    description: "Helpful message",
    variant: "destructive"
  });
  // Fallback or redirect
  navigate('/fallback-page');
}
```

Áp dụng: Tất cả pages và functions

4.2. Patterns trong dữ liệu

Pattern 1: JSONB Fields cho Flexibility

```
CREATE TABLE skill_analyses (
  id UUID PRIMARY KEY,
  student_skills JSONB DEFAULT '[]'::jsonb,  -- Array of strings
  market_skills JSONB DEFAULT '[]'::jsonb,   -- Array of objects
  skill_gaps JSONB DEFAULT '[]'::jsonb,      -- Array of objects
  ...
);
```

- Cho phép lưu structured data mà không cần nhiều bảng
- Dễ query với PostgreSQL JSON operators

Pattern 2: Row Level Security (RLS)

```
-- Pattern: Chỉ user mới thấy data của chính họ
CREATE POLICY "Users can view their own analyses"
  ON public.skill_analyses
  FOR SELECT
  USING (auth.uid() = user_id);
```

- Áp dụng cho: skill_analyses table, storage bucket
- Security by design

Pattern 3: Timestamp Tracking

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

- Trigger tự động update `updated_at`
- Audit trail

4.3. Patterns trong UI/UX

Pattern 1: Loading States

```
const [loading, setLoading] = useState(false);
const [progress, setProgress] = useState(0);
const [progressMessage, setProgressMessage] = useState('');

// Usage:
setProgress(20);
setProgressMessage('Uploading file...');
```

Nơi áp dụng: Upload.tsx (multi-step progress), Results.tsx (loading spinner)

Pattern 2: Toast Notifications

```
toast({
  title: "Success!",
  description: "Your analysis is complete",
});

toast({
  title: "Error",
  description: error.message,
  variant: "destructive"
});
```

Consistent feedback cho user actions

Pattern 3: Responsive Layout Grid

```
<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">
  {/* Cards */}
</div>
```

- Mobile-first
- Breakpoints: md (768px), lg (1024px)

5. BƯỚC 3: TRỪU TƯỢNG HÓA (Abstraction)

5.1. Abstraction Layers

Layer 1: Presentation Layer (React Components)

Trách nhiệm: Hiển thị UI, xử lý user interactions
Không quan tâm đến:

- Database structure
- API implementation details
- Authentication mechanism internals

Ví dụ:

```
// SkillGapChart.tsx - Chỉ quan tâm đến props interface
interface SkillGapChartProps {
  data?: any[];
  matchPercentage?: number;
  criticalGaps?: number;
  marketReadiness?: number;
}

export const SkillGapChart = ({ data, matchPercentage, ... }) => {
  // Render logic - không quan tâm data từ đâu
  return <ResponsiveContainer>...</ResponsiveContainer>;
};
```

Layer 2: Business Logic Layer (Edge Functions)

Trách nhiệm: Xử lý business rules, data transformation
Không quan tâm đến:

- UI rendering
- User input validation (đã validate ở frontend)
- Storage implementation

Ví dụ:

```
// analyze-skills/index.ts
// Input: transcriptText (string)
// Output: { skills: string[] }
// Abstraction: Không quan tâm text từ PDF hay Image hay manual input

const extractSkills = (text: string): string[] => {
  const normalized = normalizeText(text);
  const matches = matchSkillsFromMapping(normalized);
  return Array.from(new Set(matches)); // Deduplicate
};
```

Layer 3: Data Access Layer (Supabase Client)

Trách nhiệm: CRUD operations, authentication
Không quan tâm đến:

- Business logic
- UI state management

Ví dụ:

```
// Abstract database operations
const { data, error } = await supabase
  .from('skill_analyses')
  .select('*')
  .eq('user_id', userId);

// Component không biết data lưu thế nào, chỉ biết nhận data
```

5.2. Key Abstractions

Abstraction 1: Skill Mapping Dictionary

Problem: Một kỹ năng có nhiều cách gọi (Vietnamese, English, synonyms)
Solution: Unified mapping dictionary

```
const SKILL_MAPPING: Record<string, string[]> = {
  "python": ["python", "py", "pandas", "numpy", "lập trình python"],
  "java": ["java", "spring", "hibernate", "lập trình java"],
  // 80+ mappings
};

// Usage: Tìm "lập trình java" → trả về key "java"
```

Benefit:

- Single source of truth
- Dễ maintain và mở rộng
- Nhất quán trong toàn hệ thống

Abstraction 2: File Type Detection & Processing

```
// Abstract away complexity của việc xử lý nhiều file types
const extractTextFromFile = async (file: File): Promise<string> => {
  if (IMAGE_TYPES.includes(file.type)) {
    return await extractTextFromImage(file);  // OCR path
  }
  if (file.type === 'text/plain') {
    return await file.text();  // Direct read
  }
  if (file.type === 'application/pdf') {
    return `[PDF: ${file.name}]`;  // Placeholder
  }
  return await file.text();  // Default
};

// Caller chỉ cần: const text = await extractTextFromFile(file);
```

Abstraction 3: AI Service Gateway

Problem: Không muốn couple với một AI provider cụ thể

Solution: Generic AI function calling

```
// Generic interface
const callAI = async (prompt: string, context: any) => {
  return await fetch('https://ai.gateway.lovable.dev/v1/chat/completions', {
    method: 'POST',
    body: JSON.stringify({
      model: 'google/gemini-2.5-flash',
      messages: [{ role: 'system', content: prompt }, ...context]
    })
  });
};

// Dễ dàng swap model hoặc provider
```

Abstraction 4: Progress Tracking

```
// Upload.tsx - Abstract multi-step progress
const progressSteps = [
  { percent: 10, message: 'Starting analysis...' },
  { percent: 20, message: 'Uploading file...' },
  { percent: 35, message: 'Processing...' },
  { percent: 50, message: 'Analyzing skills...' },
  { percent: 70, message: 'Comparing with market...' },
  { percent: 90, message: 'Generating report...' },
  { percent: 100, message: 'Complete!' }
];

// Usage:
progressSteps.forEach(step => {
  setProgress(step.percent);
  setProgressMessage(step.message);
});
```

5.3. Interface Abstractions

Profile Interface

```
interface UserProfile {
  main_language: string | null;
  dream_job: string | null;
}
// Abstraction: Component không quan tâm profile có thêm fields gì khác
```

Analysis Result Interface

```
interface AnalysisResult {
  id: string;
  file_name: string;
  student_skills: any;      // Flexible - có thể là array hoặc object
  market_skills: any;
  skill_gaps: any;
  match_percentage: number;
  market_readiness: number;
  created_at: string;
}
// Abstraction: Display logic tách biệt khỏi data structure
```

Message Interface (Chatbot)

```
type Message = {
  role: "user" | "assistant";
  content: string
};
// Simple abstraction cho conversation state
```

6. BƯỚC 4: THIẾT KẾ THUẬT TOÁN (Algorithm Design)

6.1. Core Algorithms

Algorithm 1: Skill Extraction từ Transcript

File: analyze-skills/index.ts

Input:

- `transcriptText` (string): Nội dung bảng điểm
- `mainLanguage` (string): Ngôn ngữ chính của user

Output:

- `skills` (string[]): Mảng các kỹ năng đã trích xuất

Pseudocode:

```
FUNCTION extractSkills(transcriptText, mainLanguage):
  // Step 1: Normalize text
  normalizedText = transcriptText.toLowerCase()
  normalizedText = removeSpecialCharacters(normalizedText)
  normalizedText = removeExtraWhitespace(normalizedText)

  // Step 2: Initialize result set
  extractedSkills = new Set()

  // Step 3: Match against SKILL_MAPPING dictionary
  FOR EACH (standardSkill, synonyms) IN SKILL_MAPPING:
    FOR EACH synonym IN synonyms:
      IF normalizedText.contains(synonym):
        extractedSkills.add(standardSkill)
        BREAK  // Tránh duplicate từ cùng skill

  // Step 4: AI Enhancement (nếu cần)
  IF extractedSkills.size < 5:
    // Text quá ngắn hoặc mapping không match
    aiSkills = callAI_extractSkills(transcriptText, mainLanguage)
    extractedSkills.addAll(aiSkills)

  // Step 5: Deduplicate và return
  RETURN Array.from(extractedSkills)
END FUNCTION
```

Độ phức tạp:

- **Time:** O(n × m × k)
  - n: số mappings (~80)
  - m: số synonyms per skill (avg ~5)
  - k: độ dài text (search operation)
- **Space:** O(s) where s = số skills tìm được (usually < 50)

Optimizations:

- Early break khi tìm thấy synonym
- Set để tự động deduplicate
- Cache normalized text

Algorithm 2: Skill Gap Analysis

File: compare-skills/index.ts

Input:

- `studentSkills` (string[]): Kỹ năng của sinh viên
- `dreamJob` (string): Công việc mơ ước

Output:

- `skillGaps` (array): Top 10 kỹ năng còn thiếu
- `matchPercentage` (number): % match với market
- `marketReadiness` (number): % sẵn sàng làm việc

Pseudocode:

```
FUNCTION compareSkills(studentSkills, dreamJob):
  // Step 1: Load market data
  IF dreamJob EXISTS:
    marketSkillCounts = getJobSpecificSkills(dreamJob)
  ELSE:
    marketSkillCounts = loadCSVMarketData()

  // Step 2: Convert student skills to Set (lowercase)
  studentSkillSet = new Set(studentSkills.map(s => s.toLowerCase()))

  // Step 3: Calculate skill demand percentages
  totalJobs = 1491
  marketSkills = []
  FOR EACH (skill, count) IN marketSkillCounts:
    percentage = (count / totalJobs) × 100
    marketSkills.push({
      skill: skill,
      count: count,
      percentage: round(percentage)
    })

  // Step 4: Sort by demand (descending)
  marketSkills.sort((a, b) => b.count - a.count)

  // Step 5: Find gaps (top market skills NOT in student set)
  skillGaps = []
  FOR EACH marketSkill IN marketSkills:
    IF NOT studentSkillSet.has(marketSkill.skill.toLowerCase()):
      skillGaps.push(marketSkill)
    IF skillGaps.length >= 10:
      BREAK

  // Step 6: Calculate match metrics
  topMarketSkills = marketSkills.slice(0, 20)
  matchedCount = 0
  FOR EACH topSkill IN topMarketSkills:
    IF studentSkillSet.has(topSkill.skill.toLowerCase()):
      matchedCount++

  matchPercentage = (matchedCount / topMarketSkills.length) × 100

  // Step 7: Calculate market readiness (with bonus)
  bonus = (studentSkills.length > 15) ? 10 : 0
```

```
  marketReadiness = min(95, matchPercentage + bonus)

  // Step 8: Generate chart data
  chartData = topMarketSkills.slice(0, 8).map(skill => ({
    skill: skill.skill,
    yourLevel: studentSkillSet.has(skill.skill) ?
      random(60, 85) : 0,  // Simulated proficiency
    marketDemand: skill.percentage,
    gap: studentSkillSet.has(skill.skill) ?
      max(0, skill.percentage - 70) : skill.percentage
  }))

  RETURN {
    skillGaps,
    matchPercentage: round(matchPercentage),
    marketReadiness: round(marketReadiness),
    chartData
  }
END FUNCTION
```

Độ phức tạp:

- **Time:**
  - Load data: O(n) - n là số skills trong market
  - Set creation: O(s) - s là student skills
  - Sorting: O(n log n)
  - Gap finding: O(n)
  - **Total: O(n log n)**
- **Space:** O(n + s)

Key Logic Points:

1. **Match Percentage:** So với **top 20** market skills, không phải toàn bộ
2. **Bonus System:** Reward students với breadth (>15 skills)
3. **Gap Ranking:** Theo % demand, không phải count
4. **Limit 10 gaps:** Tránh overwhelm user

Algorithm 3: OCR Text Extraction

File: ocr-transcript/index.ts

Input:

- `imageBase64` (string): Ảnh đã encode
- `mimeType` (string): image/jpeg, image/png, etc.

Output:

- `text` (string): Text đã extract

Pseudocode:

```
FUNCTION extractTextFromImage(imageBase64, mimeType):
  // Step 1: Prepare AI prompt
  systemPrompt = `
    You are an OCR specialist for academic transcripts.
    Extract ALL text preserving:
    - Course names and codes
    - Grades and credits
    - Semester info
    Return ONLY extracted text, no commentary.
  `

  // Step 2: Call Vision AI
  response = await callAI({
    model: 'gemini-2.5-flash',
    messages: [
      { role: 'system', content: systemPrompt },
      {
        role: 'user',
        content: [
          { type: 'text', text: 'Extract all text from this transcript' },
          {
            type: 'image_url',
            image_url: { url: `data:${mimeType};base64,${imageBase64}` }
          }
        ]
      }
    ]
  })

  // Step 3: Extract and validate result
  extractedText = response.choices[0].message.content

  IF extractedText.length < 50:
    THROW Error('OCR result too short - image may be unclear')

  RETURN extractedText
END FUNCTION
```

Error Handling:

- Rate limit (429): Retry after delay
- Credits exhausted (402): User-friendly error

- Invalid image: Validate before sending

Algorithm 4: Streaming Chat Response

File: chat/index.ts, ChatBot.tsx

Input:

- `messages` (Message[]): Conversation history

Output:

- Streaming SSE (Server-Sent Events) response

Pseudocode:

```
FUNCTION streamChat(userMessages):
  // Backend (Deno Function):
  response = await callAI({
    model: 'gemini-2.5-flash',
    messages: [systemPrompt, ...userMessages],
    stream: true  // Enable streaming
  })

  RETURN response.body  // Stream directly to client

  // Frontend (React):
  FUNCTION handleSend():
    newMessages = [...messages, userMessage]
    setMessages(newMessages)

    response = fetch('/functions/v1/chat', {
      method: 'POST',
      body: JSON.stringify({ messages: newMessages })
    })

    reader = response.body.getReader()
    decoder = new TextDecoder()
    assistantContent = ''

    WHILE true:
      { done, value } = await reader.read()
      IF done: BREAK

      chunk = decoder.decode(value)
      lines = chunk.split('\n')

      FOR EACH line IN lines:
        IF line.startsWith('data: '):
          jsonStr = line.slice(6)
          IF jsonStr === '[DONE]': BREAK

          parsed = JSON.parse(jsonStr)
          content = parsed.choices[0].delta.content

          IF content:
            assistantContent += content
            // Update UI immediately (real-time typing effect)
```

```
            setMessages(prev => updateLastMessage(assistantContent))
END FUNCTION
```

Key Features:

- **Streaming:** Từng token hiện ngay, không đợi full response
- **Real-time Update:** React state update theo từng chunk
- **Error Recovery:** Try-catch cho JSON parsing

Algorithm 5: Market Data Loading

File: compare-skills/index.ts

Input: Không (hoặc dreamJob nếu có)

Output: Map<skill, count>

Pseudocode:

```
FUNCTION loadMarketData():
  // Step 1: Read CSV file
  csvPath = '../../../public/data/market_skills.csv'
  csvText = await Deno.readTextFile(csvPath)

  // Step 2: Parse CSV
  lines = csvText.split('\n')
  headers = lines[0].split(',')  // skill,count

  skillMap = new Map()

  FOR i = 1 TO lines.length - 1:
    IF lines[i].trim() === '': CONTINUE

    [skill, count] = lines[i].split(',')
    skillMap.set(skill.trim().toLowerCase(), parseInt(count))

  RETURN skillMap
END FUNCTION

FUNCTION getJobSpecificSkills(dreamJob):
  // Step 1: Call AI to get job requirements
  prompt = `
    List the top 20 technical skills required for: ${dreamJob}
    Return only skill names, one per line, no explanations.
  `

  response = await callAI(prompt)
  skills = response.split('\n').filter(s => s.trim())

  // Step 2: Create equal-weight map
  skillMap = new Map()
  FOR EACH skill IN skills:
    skillMap.set(skill.toLowerCase(), 100)  // Equal priority

  RETURN skillMap
END FUNCTION
```

Data Source:

- Primary: market_skills.csv - real job market data
- Fallback: AI-generated skills cho specific dream job

6.2. Helper Algorithms

Text Normalization

```
FUNCTION normalizeText(text):
  normalized = text.toLowerCase()
  normalized = normalized.replace(/[^\w\s]/g, ' ')  // Remove special chars
  normalized = normalized.replace(/\s+/g, ' ')      // Collapse whitespace
  RETURN normalized.trim()
END FUNCTION
```

File Size Validation

```
FUNCTION validateFileSize(file, maxSizeMB):
  maxBytes = maxSizeMB × 1024 × 1024
  IF file.size > maxBytes:
    THROW Error(`File must be smaller than ${maxSizeMB}MB`)
  RETURN true
END FUNCTION
```

Base64 Encoding

```
FUNCTION fileToBase64(file):
  reader = new FileReader()
  RETURN new Promise((resolve, reject) => {
    reader.onload = () => {
      result = reader.result  // "..."
      base64 = result.split(',')[1]  // Extract base64 part
      resolve(base64)
    }
    reader.onerror = reject
    reader.readAsDataURL(file)
  })
END FUNCTION
```

7. PHÂN TÍCH KỸ THUẬT CHI TIẾT

7.1. Frontend Architecture

State Management Strategy

```
// Sử dụng @tanstack/react-query cho server state
const queryClient = new QueryClient();

// Component-level state cho UI
const [isOpen, setIsOpen] = useState(false);
const [loading, setLoading] = useState(false);

// Supabase real-time subscriptions
useEffect(() => {
  const channel = supabase
    .channel('skill_analyses')
    .on('postgres_changes', {
      event: 'INSERT',
      schema: 'public'
    }, payload => {
      // Real-time updates
    })
    .subscribe();

  return () => channel.unsubscribe();
}, []);
```

Routing Strategy

```
// App.tsx - Centralized routing
<Routes>
  <Route path="/" element={<Index />} />           // Public
  <Route path="/auth" element={<Auth />} />        // Public
  <Route path="/upload" element={<Upload />} />    // Protected
  <Route path="/results/:id" element={<Results />} />  // Protected
  <Route path="/history" element={<History />} />  // Protected
  <Route path="/profile" element={<Profile />} />  // Protected
  <Route path="/personalize" element={<Personalize />} />  // Protected
  <Route path="*" element={<NotFound />} />        // Catch-all
</Routes>

// Protected routes check auth internally
useEffect(() => {
  const { data: { user } } = await supabase.auth.getUser();
  if (!user) navigate('/auth');
}, []);
```

Styling Approach

- **Tailwind CSS:** Utility-first, responsive
- **CSS Variables:** Theme customization

```
:root {
  --primary: 222.2 47.4% 11.2%;
  --accent: 210 40% 96.1%;
  --destructive: 0 84.2% 60.2%;
}
```

- **Dark Mode Support:** `next-themes` package
- **Component Variants:** `class-variance-authority`

7.2. Backend Architecture

Database Schema

```
-- Core tables
profiles (
  id UUID PRIMARY KEY REFERENCES auth.users,
  main_language TEXT,
  dream_job TEXT,
  created_at TIMESTAMPTZ,
  updated_at TIMESTAMPTZ
)

skill_analyses (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES auth.users,
  file_name TEXT,
  file_url TEXT,
  student_skills JSONB,    -- ["python", "java", ...]
  market_skills JSONB,     -- [{skill, count, percentage}, ...]
  skill_gaps JSONB,        -- [{skill, count, percentage}, ...]
  match_percentage DECIMAL(5,2),
  market_readiness DECIMAL(5,2),
  created_at TIMESTAMPTZ,
  updated_at TIMESTAMPTZ
)

-- Storage
storage.buckets (
  transcripts -- User-uploaded files
)
```

Security Model

```sql
-- Row Level Security (RLS)
ALTER TABLE skill_analyses ENABLE ROW LEVEL SECURITY;

-- Policy: Users only see their own data
CREATE POLICY "Own data only"
  ON skill_analyses
  FOR ALL
  USING (auth.uid() = user_id);

-- Storage security
CREATE POLICY "Own files only"
  ON storage.objects
  FOR ALL
  USING (
    bucket_id = 'transcripts' AND
    auth.uid()::text = (storage.foldername(name))[1]
  );
```

Edge Functions Deployment

```
# Structure
supabase/functions/
├── analyze-skills/
│   └── index.ts
├── compare-skills/
│   └── index.ts
├── ocr-transcript/
│   └── index.ts
├── get-job-skills/
│   └── index.ts
└── chat/
    └── index.ts

# Deploy command
supabase functions deploy analyze-skills --no-verify-jwt
```

7.3. AI Integration

Gemini 2.5 Flash - Use Cases

1. **OCR (Vision):**

```
// Image → Text extraction
model: 'google/gemini-2.5-flash'
input: { type: 'image_url', image_url: { url: base64Image } }
output: Structured text
```

2. **Skill Extraction (Text Analysis):**

```
// Text → Skills array
// Hiểu context của môn học → technical skills
"Lập trình hướng đối tượng với Java" → ["java", "oop"]
```

3. **Job Skills Lookup:**

```
// Dream job → Required skills
prompt: "List top skills for Full Stack Developer"
output: ["javascript", "react", "node.js", "sql", ...]
```

4. **Conversational AI (Chatbot):**

```
// Multi-turn conversation với context
system: "You are SkillGap AI Assistant..."
messages: [...conversation history]
stream: true  // Real-time responses
```

API Rate Limiting & Costs

- **Rate Limit:** Handled với 429 status
- **Cost Control:** Credits system
- **Error Messages:** User-friendly 402 response

7.4. Data Visualization

Recharts Configuration

```jsx
<ResponsiveContainer width="100%" height={400}>
  <BarChart data={chartData}>
    <CartesianGrid strokeDasharray="3 3" />
    <XAxis dataKey="skill" />
    <YAxis />
    <Tooltip
      contentStyle={{
        backgroundColor: 'hsl(var(--card))',
        border: '1px solid hsl(var(--border))',
        borderRadius: '0.75rem'
      }}
    />
    <Legend />
    <Bar dataKey="yourLevel" fill="hsl(var(--primary))" name="Your Level" />
    <Bar dataKey="marketDemand" fill="hsl(var(--accent))" name="Market Demand" />
  </BarChart>
</ResponsiveContainer>
```

Chart Data Structure:

```jsx
[
  {
    skill: 'React',
    yourLevel: 70,        // 0-100 scale
    marketDemand: 95,     // Percentage of jobs requiring
    gap: 25               // Difference
  },
  // ... more skills
]
```

7.5. Performance Optimizations

Code Splitting

```
// Vite auto code-splits by route
import Index from './pages/Index';  // Separate chunk
import Upload from './pages/Upload';  // Separate chunk
```

Lazy Loading

```
// Components load on-demand
const ChatBot = lazy(() => import('@/components/ChatBot'));
```

Image Optimization

```
// Tailwind responsive images
<img
  src={heroBg}
  className="w-full h-auto"
  loading="lazy"
  alt="Hero background"
/>
```

Database Indexing

```
-- Auto-indexed on:
CREATE INDEX ON skill_analyses(user_id);  -- For filtering
CREATE INDEX ON skill_analyses(created_at DESC);  -- For sorting
```

7.6. Error Handling Strategy

Frontend Error Boundaries

```
// Global error handler
window.addEventListener('unhandledrejection', event => {
  console.error('Unhandled promise rejection:', event.reason);
  toast({
    title: "Something went wrong",
    description: "Please try again or contact support",
    variant: "destructive"
  });
});
```

Backend Error Responses

```
// Standardized error format
{
  error: "User-friendly error message",
  code: "ERROR_CODE",  // Optional
  details: { ... }     // Optional, for debugging
}
```

Network Error Handling

```
try {
  const response = await supabase.functions.invoke('analyze-skills', {...});
  if (response.error) throw response.error;
} catch (error) {
  if (error.message.includes('network')) {
    toast({ title: "Network error", description: "Check connection" });
  } else if (error.message.includes('timeout')) {
    toast({ title: "Request timeout", description: "Try again" });
  } else {
    toast({ title: "Error", description: error.message });
  }
}
```

8. KẾT LUẬN

8.1. Tóm tắt ứng dụng Tư duy Tính toán

Decomposition (Phân rã vấn đề)

✅ **Thành công:**

- Chia hệ thống thành 9 modules độc lập
- Mỗi module có responsibility rõ ràng
- Frontend/Backend/AI layers tách biệt hoàn toàn

Pattern Recognition (Nhận dạng mẫu)

✅ **Thành công:**

- Nhận diện được 6 architectural patterns
- Áp dụng consistent patterns: Component-based, Serverless, Auth flow
- Tái sử dụng patterns giảm code duplication

Abstraction (Trừu tượng hóa)

✅ **Thành công:**

- 4 layers abstraction: Presentation, Business Logic, Data Access, AI Service
- Interface-driven development
- Skill mapping dictionary = single source of truth

Algorithm Design (Thiết kế thuật toán)

✅ **Thành công:**

- 5 core algorithms với complexity analysis
- Optimized skill extraction: O(n × m × k)
- Efficient gap analysis: O(n log n)
- Streaming chat cho real-time UX

8.2. Điểm mạnh của dự án

1. **Modularity:** Dễ maintain và mở rộng
2. **Security:** RLS policies, JWT auth

3. **Scalability:** Serverless architecture, edge functions
4. **UX:** Real-time feedback, progressive enhancement
5. **AI-Powered:** Gemini 2.5 cho OCR, skills analysis, chatbot
6. **Data-Driven:** Real market data (1491 job postings)
7. **Multilingual:** Vietnamese/English support
8. **Responsive:** Mobile-first design

8.3. Cơ hội cải thiện

1. **Testing:** Thêm unit tests, integration tests
2. **Analytics:** Track user behavior, conversion funnel
3. **Caching:** Redis cho market data, skill mappings
4. **Batch Processing:** Xử lý nhiều files cùng lúc
5. **Export:** PDF/Excel reports
6. **Recommendations:** AI-powered learning path suggestions
7. **Social Features:** Share results, compare with peers
8. **Notifications:** Email alerts khi có new gap analysis

8.4. Tech Stack Assessment

Component	Technology	Rating	Lý do
Frontend Framework	React 18 + TypeScript	⭐⭐⭐⭐⭐	Type safety, component reusability
UI Library	shadcn/ui	⭐⭐⭐⭐⭐	Beautiful, accessible, customizable
Backend	Supabase	⭐⭐⭐⭐⭐	All-in-one: DB, Auth, Storage, Functions
AI	Gemini 2.5 Flash	⭐⭐⭐⭐☆	Fast, affordable, multi-modal
Build Tool	Vite	⭐⭐⭐⭐⭐	Extremely fast HMR
Deployment	Lovable	⭐⭐⭐⭐☆	Easy CI/CD, edge deployment

8.5. Học được gì từ dự án

Về Computational Thinking:

1. **Decomposition giúp quản lý complexity:** Chia nhỏ → dễ code, debug, test
2. **Pattern recognition tiết kiệm thời gian:** Tái sử dụng solutions đã proven
3. **Abstraction tăng flexibility:** Dễ swap implementations không ảnh hưởng toàn hệ thống
4. **Algorithm design cần balance:** Performance vs. Readability vs. Maintainability

Về Software Engineering:

1. **Security by design:** RLS policies từ đầu, không hotfix sau
2. **User-centric development:** Progress bars, loading states, error messages
3. **Data-driven decisions:** Real job market data → accurate analysis
4. **Iterative improvement:** Start simple, add complexity khi cần

Về Team Collaboration:

1. **Code organization matters:** Clear folder structure giúp onboarding
2. **Documentation is key:** Comments, README, migration files
3. **Consistent naming:** camelCase cho JS, snake_case cho DB
4. **Version control:** Git commits với meaningful messages

PHỤ LỤC

A. File Structure Map

```
SkillGapAI/
├── src/
│   ├── components/      (UI components)
│   │   ├── ui/         (40+ shadcn components)
│   │   ├── ChatBot.tsx
│   │   ├── SkillGapChart.tsx
│   │   ├── TopNav.tsx
│   │   └── ...
│   ├── pages/          (Route pages)
│   │   ├── Index.tsx
│   │   ├── Upload.tsx
│   │   ├── Results.tsx
│   │   ├── History.tsx
│   │   ├── Auth.tsx
│   │   ├── Personalize.tsx
│   │   ├── Profile.tsx
│   │   └── NotFound.tsx
│   ├── hooks/          (Custom React hooks)
│   ├── lib/            (Utilities)
│   ├── integrations/   (Supabase client)
│   └── App.tsx         (Root component)
├── supabase/
│   ├── functions/      (Edge Functions)
│   │   ├── analyze-skills/
│   │   ├── compare-skills/
│   │   ├── ocr-transcript/
│   │   ├── get-job-skills/
│   │   └── chat/
│   └── migrations/     (DB schema versions)
├── public/
│   └── data/
│       ├── market_skills.csv
│       └── skills_config.py
└── package.json
```

B. Environment Variables

```
VITE_SUPABASE_URL=https://xxx.supabase.co
VITE_SUPABASE_PUBLISHABLE_KEY=eyJxxx...
LOVABLE_API_KEY=sk-xxx...  (Backend only)
```

C. Key Dependencies

```
{
  "dependencies": {
    "react": "^18.3.1",
    "react-router-dom": "^6.30.1",
    "@supabase/supabase-js": "^2.83.0",
    "@tanstack/react-query": "^5.83.0",
    "recharts": "^2.15.4",
    "zod": "^3.25.76",
    "react-hook-form": "^7.61.1",
    "lucide-react": "^0.462.0",
    "tailwindcss": "^3.4.17"
  }
}
```

D. Database Commands

```
# Start Supabase locally
supabase start

# Apply migrations
supabase db push

# Reset database
supabase db reset

# Deploy functions
supabase functions deploy analyze-skills
```

E. Deployment Workflow

```
# 1. Development
npm run dev              # Start dev server (port 8080)

# 2. Build
npm run build            # Vite build → dist/

# 3. Preview
npm run preview          # Test production build locally

# 4. Deploy (via Lovable)
git push origin main     # Auto-deploy to Lovable hosting
```

Người viết báo cáo: Nguyễn Hồ Anh Quốc
Ngày hoàn thành: 12/12/2025

Ghi chú: Báo cáo này phân tích toàn diện dự án SkillGapAI theo 4 bước của Tư duy Tính toán. Mọi thông tin được trích xuất trực tiếp từ source code và có references đến files cụ thể.