WeVibeCode.ai - Project Documentation & Current State

Last Updated: January 9, 2026 **Status:** ✅ Successfully Deployed to Vercel **Live URL:** [Your Vercel URL]

🎯 Project Overview

WeVibeCode.ai is an AI-powered platform that generates professional websites and applications in seconds using GPT-4. Users describe their vision in plain text, and the AI creates complete HTML/CSS/JS code with modern design, vibrant colors, and responsive layouts.

Core Value Proposition

- **Input:** Plain text description (e.g., "A landing page for a fitness app")
- **Output:** Stunning, production-ready website with professional design
- **Time:** 15-30 seconds per generation
- **Cost:** 1 credit per website (~$0.03-0.10 API cost)

🛠️ Tech Stack

Frontend

- **Framework:** Next.js 16.1.1 (App Router with Turbopack)
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Internationalization:** next-intl (5 languages)
- **Icons:** Lucide React + Emoji

Backend

- **Database:** Supabase (PostgreSQL)
- **Auth:** Supabase Auth with RLS
- **AI:** OpenAI GPT-4o & GPT-4o-mini
- **Storage:** Supabase Storage (for future file uploads)

Deployment

- **Platform:** Vercel
- **Repository:** [Your GitHub repo if connected]
- **Domain:** [Your custom domain if set up]

📂 Project Structure

```
wevibecode-ai/
├── app/
│   ├── [locale]/                # Internationalized routes
│   │   └── auth/
│   │       ├── login/page.tsx
│   │       └── signup/page.tsx
│   ├── api/
│   │   ├── generate-website/route.ts    # Main AI generation
│   │   ├── generate-image/route.ts      # DALL-E integration
│   │   └── preview/[id]/route.ts        # Preview serving
│   ├── dashboard/
│   │   ├── page.tsx                # Main dashboard
│   │   ├── generate/page.tsx        # Website generation form
│   │   ├── history/page.tsx         # Generation history
│   │   ├── preview/[id]/page.tsx    # Preview viewer
│   │   ├── test-preview/page.tsx    # Preview testing
│   │   └── create-website/page.tsx  # Redirect to generate
│   └── page.tsx                    # Homepage
├── components/
│   ├── Preview.tsx                 # Preview component (Phase 4)
│   └── LanguageSwitcher.tsx         # Language selector
├── lib/
│   ├── preview.ts                  # Preview CRUD utilities
│   └── hooks/
│       └── useAuth.ts              # Auth hook
├── public/
│   └── locales/                    # Translation files
│       ├── en/common.json
│       ├── es/common.json
│       ├── it/common.json
│       ├── pl/common.json
│       └── de/common.json
├── middleware.ts                   # Simple auth middleware
└── .env.local                      # Environment variables
```

Database Schema

Tables

profiles

```sql
- id (UUID, PK) → references auth.users
- email (TEXT)
- full_name (TEXT)
- credits_remaining (INT) DEFAULT 10
- subscription_tier (VARCHAR) DEFAULT 'free'
- preferred_language (VARCHAR) DEFAULT 'en'
- created_at (TIMESTAMPTZ)
```

previews

```sql
- id (UUID, PK)
- user_id (UUID) → references auth.users
- title (VARCHAR)
- html_content (TEXT)
- css_content (TEXT)
- js_content (TEXT)
- preview_type (VARCHAR) DEFAULT 'website'
- generation_prompt (TEXT)
- generation_type (VARCHAR)
- credits_used (INT) DEFAULT 1
- is_published (BOOLEAN) DEFAULT false
- created_at (TIMESTAMPTZ)
- updated_at (TIMESTAMPTZ)
```

credits_log

```sql
```

```
- id (UUID, PK)
- user_id (UUID) → references auth.users
- credits_used (INT)
- action (VARCHAR) -- 'generate_website', 'generate_image'
- details (JSONB)
- created_at (TIMESTAMPTZ)
```

RLS Policies

All tables have Row Level Security enabled:

- Users can only access their own data
- INSERT/UPDATE/DELETE restricted to authenticated users
- SELECT policies check `auth.uid() = user_id`

🔑 Environment Variables

Required in `.env.local` and Vercel:

```
env

# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://your-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbGc...

# OpenAI
OPENAI_API_KEY=sk-proj-...
```

✅ Completed Phases

Phase 1: Core Setup ✅

Status: Complete **Features:**

- Multi-language support (EN, ES, IT, PL, DE)
- User authentication (login/signup)
- Character encoding fixed (proper UTF-8)
- Login button in header

- Language switcher with flags

Files:

- `app/[locale]/auth/login/page.tsx`
- `app/[locale]/auth/signup/page.tsx`
- `components/LanguageSwitcher.tsx`
- `public/locales/*/common.json`

Phase 2: Language Persistence ✅

Status: Complete **Features:**

- localStorage for guest users
- Database storage for authenticated users
- Automatic language detection
- Cookie-based fallback

Implementation:

- Language stored in `profiles.preferred_language`
- Simple middleware for auth
- LanguageSwitcher saves to both cookie and DB

Phase 3: AI Website Generation ✅ ⭐

Status: Complete (CORE FEATURE) **Features:**

- **Two-step AI generation:**
  1. GPT-4o-mini expands user prompt (3 sec, $0.001)
  2. GPT-4o generates website (15 sec, $0.03-0.10)
- **Business name options:** AI-generated or user-provided
- **6 website types:** Landing, Portfolio, Business, Restaurant, Blog, E-commerce
- **6 color schemes:** With vibrant gradients
- **Emoji icons:** Throughout interface and generated sites

- **Credits system:** 1 credit per generation
- **Generation history:** View, re-open, delete past generations
- **Auto-preview:** Seamlessly opens in Phase 4 preview

Key Files:

- `app/api/generate-website/route.ts` (SUPER ENHANCED version)
- `app/dashboard/generate/page.tsx` (with emoji icons)
- `app/dashboard/history/page.tsx`
- `lib/preview.ts`

Prompt Enhancement System: User writes: "A landing page for a fitness app"

AI expands to:

- Creative business name (e.g., "FitPulse Pro")
- Detailed sections (hero, features, testimonials, pricing)
- Specific headlines and CTAs
- Color palette with hex codes
- Emoji icons for each section
- Visual styling instructions

Result: Professional, impressive website that looks like it cost $10,000+

Phase 4: Live Preview System ✅

Status: Complete **Features:**

- **Responsive preview:** Desktop (100%), Tablet (768px), Mobile (375px)
- **Interactive controls:** Device toggles, refresh, fullscreen, close
- **Secure rendering:** Sandboxed iframe with CSP
- **API serving:** `/api/preview/[id]` serves full HTML
- **Database integration:** Stores HTML/CSS/JS in `previews` table

Key Files:

- `components/Preview.tsx`

- `app/api/preview/[id]/route.ts` (Next.js 16 compatible)
- `app/dashboard/preview/[id]/page.tsx`
- `app/dashboard/test-preview/page.tsx`

Integration:

- After generation, auto-redirects to preview
- Shows live, interactive website
- Can toggle between device sizes
- Fullscreen mode for demos

🎨 Design System

Color Schemes (AI-Generated)

1. **Electric:** `#6366F1` (indigo) → `#EC4899` (pink) → `#8B5CF6` (purple)
2. **Energy:** `#F59E0B` (amber) → `#EF4444` (red) → `#F97316` (orange)
3. **Fresh:** `#10B981` (green) → `#3B82F6` (blue) → `#06B6D4` (cyan)
4. **Bold:** `#8B5CF6` (purple) → `#EC4899` (pink) → `#F43F5E` (rose)
5. **Modern:** `#0EA5E9` (sky) → `#6366F1` (indigo) → `#A855F7` (purple)

Website Types & Icons

- 🚀 Landing Page
- 💼 Portfolio
- 🏢 Business
- 🍽️ Restaurant
- 📝 Blog
- 🛒 E-commerce

Typography

- **Headlines:** 48-72px, bold (700-800)
- **Subheadlines:** 24-32px
- **Body:** 18px, line-height 1.7

- **Fonts:** System fonts (-apple-system, BlinkMacSystemFont, 'Segoe UI')

🔄 User Flow

1. New User Arrives

```
Homepage → Sign Up → Dashboard → Generate
```

2. Website Generation Flow

```
1. Dashboard → Click "Create Website"
2. Generation Form:
   - Choose: AI name or provide own
   - Describe website (textarea)
   - Select type (6 options with icons)
   - Pick color scheme (6 options)
3. Click "Generate Website (1 Credit)"
4. AI processes (15-20 seconds):
   - Step 1: Enhance prompt
   - Step 2: Generate HTML/CSS/JS
5. Auto-redirect to Preview
6. Preview page shows:
   - Device toggles (Desktop/Tablet/Mobile)
   - Refresh button
   - Fullscreen option
   - Live, interactive website
```

3. History Management

```
Dashboard → History → View past generations
- See all generated websites
- Click to preview again
- Delete unwanted ones
```

🚨 Known Issues & Fixes Applied

1. Supabase API Migration

Issue: Old `@supabase/auth-helpers-nextjs` package deprecated **Fix:** Updated all files to use `@supabase/ssr` with `createBrowserClient` and `createServerClient` **Files Fixed:**

- All auth pages
- API routes
- Components (LanguageSwitcher)
- Hooks (useAuth)

2. Next.js 16 Compatibility

Issue: `params` and `cookies()` are now async in Next.js 16 **Fix:** Added `await` for params and proper cookie handling **Example:**

```typescript
// OLD
const params = context.params;

// NEW
const params = await context.params;
```

3. Character Encoding

Issue: Non-UTF-8 characters showing as `Ã³`, `Å,`, etc. **Fix:** Re-created all translation files with proper UTF-8 encoding **Files:** `public/locales/{es,it,pl,de}/common.json`

4. TypeScript Build Errors

Issue: Vercel checking unused implementation folders **Fix:** Deleted `wevibecode-custom-implementation` and similar folders **Prevention:** Added `.vercelignore` if needed

---

💰 Cost Structure

OpenAI API Costs Per Generation

- **Prompt Enhancement:** $0.001 (GPT-4o-mini)
- **Website Generation:** $0.03-0.10 (GPT-4o)
- **Total:** ~$0.031-0.101 per website

Credits System

- New users: 10 free credits
- Cost per credit: ~$0.05 (covers API + margin)

- 1 credit = 1 website generation

Potential Pricing Tiers

- **Free:** 10 credits (one-time)
- **Starter:** $9/month (20 credits)
- **Pro:** $19/month (50 credits)
- **Business:** $49/month (150 credits)

🌍 Multi-Language Support

Supported Languages

1. **English (EN)** `GB` - Default
2. **Spanish (ES)** `ES` - Español
3. **Italian (IT)** `IT` - Italiano
4. **Polish (PL)** `PL` - Polski
5. **German (DE)** `DE` - Deutsch

Translation Structure

Each language has `common.json` with:

- `header` - Navigation items
- `hero` - Homepage hero section
- `apps` - App descriptions
- `howItWorks` - Process steps
- `pricing` - Pricing tiers
- `cta` - Call-to-action text
- `footer` - Footer content
- `auth` - Login/signup text
- `errors` - Error messages

Language Persistence

- **Guests:** Saved in `preferred_locale` cookie (365 days)

- **Logged-in users:** Saved in `profiles.preferred_language` column
- **Priority:** Database > Cookie > Browser Accept-Language

🔐 Security & Authentication

Supabase Auth

- Email/password authentication
- JWT tokens stored in cookies
- Refresh token rotation
- RLS policies on all tables

API Security

- All routes check authentication
- Credit verification before generation
- Rate limiting (planned)
- CORS configured for same-origin

Preview Security

- Sandboxed iframes
- Content Security Policy (CSP)
- X-Frame-Options: SAMEORIGIN
- No external script execution

📊 Key Metrics to Track

User Metrics

- New signups per day
- Active users (DAU/MAU)
- Conversion rate (visitor → signup)
- Retention rate (7-day, 30-day)

Generation Metrics

- Websites generated per day
- Average generation time
- Success rate (successful generations)
- Most popular website types
- Most popular color schemes

Financial Metrics

- API costs per generation
- Revenue per user
- Credits purchased
- Churn rate

🚀 Deployment Info

Vercel Configuration

- **Framework:** Next.js
- **Build Command:** `npm run build`
- **Output Directory:** `.next`
- **Install Command:** `npm install`
- **Node Version:** 18.x or higher

Environment Variables (Must be set in Vercel)

```
NEXT_PUBLIC_SUPABASE_URL
NEXT_PUBLIC_SUPABASE_ANON_KEY
OPENAI_API_KEY
```

Build Optimizations

- Turbopack enabled (faster builds)
- TypeScript strict mode
- Excluded unnecessary folders via `.vercelignore`

📝 Code Conventions & Patterns

File Naming

- Pages: `page.tsx`
- Components: `ComponentName.tsx` (PascalCase)
- Utilities: `utilityName.ts` (camelCase)
- API routes: `route.ts`

Component Patterns

- `'use client'` for interactive components
- Server components by default
- Async/await for all API calls
- Error boundaries with try/catch

API Route Pattern

```typescript
```

```
export async function POST(request: NextRequest) {
  try {
    // 1. Parse request
    const body = await request.json();

    // 2. Create Supabase client
    const supabase = createServerClient(...);

    // 3. Check authentication
    const { data: { user } } = await supabase.auth.getUser();
    if (!user) return unauthorized;

    // 4. Check credits
    const { data: profile } = await supabase...
    if (profile.credits_remaining < 1) return insufficientCredits;

    // 5. Do work (AI generation, etc.)
    const result = await doWork();

    // 6. Deduct credit
    await supabase.from('profiles').update({...});

    // 7. Log usage
    await supabase.from('credits_log').insert({...});

    // 8. Return success
    return NextResponse.json({ success: true, data: result });

  } catch (error) {
    return NextResponse.json({ error: error.message }, { status: 500 });
  }
}
```

🐛 Common Issues & Solutions

Issue: "Module not found: @supabase/auth-helpers-nextjs"

Solution: Update to `@supabase/ssr`:

```
typescript

import { createBrowserClient } from '@supabase/ssr';
```

Issue: "params is a Promise"

Solution: Await params in Next.js 16:

```typescript
const params = await context.params;
```

Issue: Preview shows "Preview not found"

Solution: Check preview exists in database and user has permission (RLS)

Issue: Language doesn't persist

Solution: Verify cookie is set and database column exists

Issue: Build fails on Vercel

Solution: Check all environment variables are set in Vercel dashboard

---

🎯 Future Phases (Not Yet Implemented)

Phase 5: Translation Tools (2-3 days)

- Auto-translate generated websites to all 5 languages
- Use DeepL or Google Translate API
- Manage translations in dashboard
- Export multilingual versions

Phase 6: Portfolio System (2-3 days)

- User portfolio pages (public URLs)
- Gallery of created sites/apps
- Share functionality (social, embed)
- Showcase best generations

Additional Features to Consider

- **Export/Download:** Download HTML/CSS/JS files
- **Deploy to Domain:** Publish to custom domains
- **Code Editor:** Edit generated code with live preview

- **Template Library:** Pre-made starting templates
- **Regenerate/Iterate:** Refine existing generations
- **SEO Optimization:** Meta tags, Open Graph, structured data
- **Analytics:** Track views, clicks, performance
- **Collaboration:** Share with team, comments
- **Version History:** Save iterations
- **A/B Testing:** Multiple versions, analytics
- **Mobile Apps:** Generate React Native code
- **Payment Integration:** Stripe for credit purchases
- **Admin Dashboard:** User management, analytics
- **Email Notifications:** Generation complete, low credits
- **API Access:** Let users generate via API

💡 Optimization Opportunities

Performance

- [ ] Implement caching for previews
- [ ] Optimize images with Next.js Image
- [ ] Add loading skeletons
- [ ] Lazy load preview iframe
- [ ] Compress generated code

UX Improvements

- [ ] Add onboarding tutorial
- [ ] Improve error messages
- [ ] Add progress indicators
- [ ] Implement undo/redo
- [ ] Add keyboard shortcuts
- [ ] Better mobile experience

AI Improvements

- [ ] Fine-tune prompts based on feedback
- [ ] Add more website types
- [ ] Include design variations

- [ ] Learn from user edits
- [ ] Implement feedback loop

Business Features

- [ ] Payment integration (Stripe)
- [ ] Subscription management
- [ ] Credit packages
- [ ] Referral system
- [ ] Affiliate program

📞 Quick Reference Commands

Development

```powershell
# Start dev server
npm run dev

# Build for production
npm run build

# Type check
npm run type-check
```

Deployment

```powershell
# Deploy to Vercel
vercel --prod

# Check deployment status
vercel ls

# View logs
vercel logs
```

Database

```sql
```

```sql
-- Check user credits
SELECT email, credits_remaining FROM profiles;

-- View recent generations
SELECT title, generation_type, created_at FROM previews ORDER BY created_at DESC LIMIT 10;

-- Check credit usage
SELECT user_id, SUM(credits_used) as total FROM credits_log GROUP BY user_id;

-- Reset user credits (for testing)
UPDATE profiles SET credits_remaining = 10 WHERE email = 'user@example.com';
```

🎓 Key Learnings & Best Practices

What Worked Well

1. **Two-step AI generation** - Dramatically improved output quality
2. **Emoji icons** - Made UI more visual and friendly
3. **Vibrant colors** - Made generated sites look premium
4. **Business name option** - Flexibility appreciated by users
5. **Preview system** - Essential for user confidence
6. **Credits system** - Clear, simple monetization

What to Avoid

1. **Generic prompts** - Always enhance/expand user input
2. **Dull color schemes** - Users want bold, exciting designs
3. **Missing visual elements** - Icons, images, gradients are crucial
4. **No preview** - Users need to see before committing
5. **Old Supabase packages** - Always use latest (@supabase/ssr)

Tips for Future Development

1. Always read SKILL.md files before implementing features
2. Test locally with `npm run build` before deploying
3. Use `.vercelignore` to exclude unnecessary folders
4. Keep API costs in mind when designing features

5. Prioritize user feedback over additional features
6. Document as you go (not after)

🤝 Collaboration Context for Claude

When continuing this project in a new conversation:

Start by saying:

"I'm continuing work on WeVibeCode.ai. Here's the project documentation: [paste this file]"

Claude should know:

- All phases completed (1, 2, 3, 4)
- Current file structure and locations
- Database schema and RLS policies
- AI generation system (two-step enhancement)
- Common issues and their fixes
- Supabase SSR patterns (not old auth-helpers)
- Next.js 16 async params pattern

Claude should ask:

- What feature/phase to work on next?
- Any specific issues or bugs to fix?
- Should we test locally or deploy?
- Any user feedback to address?

📈 Success Metrics (Current)

Technical

- ✅ Successfully deployed to Vercel
- ✅ All builds passing
- ✅ TypeScript errors resolved

- ✅ RLS policies working
- ✅ API routes functional

Feature Completeness

- ✅ Core generation working (Phase 3)
- ✅ Preview system working (Phase 4)
- ✅ Multi-language support (Phase 1)
- ✅ Authentication working (Phase 1)
- ✅ Credits system working (Phase 3)

User Experience

- ✅ Generation time: 15-30 seconds
- ✅ Output quality: Professional/impressive
- ✅ UI polish: Modern, clean, vibrant
- ✅ Mobile responsive: All pages
- ✅ Error handling: Graceful, informative

🎉 Current Milestone

Status: ✅ SUCCESSFULLY DEPLOYED TO PRODUCTION

Live Features:

- AI website generation (6 types)
- Live preview system (3 device sizes)
- Multi-language support (5 languages)
- User authentication
- Credits system
- Generation history

Ready For:

- User testing and feedback
- Marketing and launch

- Additional features (Phases 5-6)
- Monetization (Stripe integration)
- Scaling optimizations

🗓️ Project Timeline

- **Day 1-2:** Phase 1 (Setup, Auth, i18n)
- **Day 2:** Phase 2 (Language Persistence)
- **Day 2-3:** Phase 4 (Preview System)
- **Day 3-4:** Phase 3 (AI Generation)
- **Day 4:** Deployment & Fixes
- **Total:** ~4 days to MVP

Next: Phases 5-6 or polish/monetization

🔗 Important Links

- **Live Site:** [Your Vercel URL]
- **Supabase Dashboard:** https://app.supabase.com
- **Vercel Dashboard:** https://vercel.com/dashboard
- **OpenAI API:** https://platform.openai.com
- **GitHub Repo:** [If applicable]

📝 Notes for Future Claude

- User prefers working with scripts/automation
- Windows PowerShell environment
- VS Code as primary editor
- Likes emoji icons and vibrant colors
- Values speed and efficiency
- Appreciates detailed documentation

- Tests thoroughly before deploying

End of Documentation Last Updated: January 9, 2026 **Version:** 1.0.0 (MVP Complete)