adgener.ai

A modern SaaS platform for generating social media ads with AI. Create professional photo and video advertisements using industry-specific templates and AI-powered generation.

Features

- 🎨 **AI-Powered Generation**: Create photo (1 token) and video (5 tokens) ads
- 📱 **Industry Templates**: Pre-built templates for apartments, restaurants, gyms, dispensaries, e-commerce, universities, and corporate
- 💳 **Token Economy**: Pay-per-use system with Stripe integration
- 📚 **Asset Library**: Organize and manage all your generated content
- 🎯 **Brand Integration**: Upload logos, set brand colors, and maintain consistency
- 📱 **Mobile-First**: Responsive design that works on all devices
- 🔐 **Secure Auth**: Email magic links and Google OAuth via Auth.js

Tech Stack

- **Framework**: Next.js 14 (App Router) + TypeScript
- **Styling**: Tailwind CSS + shadcn/ui + Framer Motion
- **Database**: Prisma ORM (SQLite dev, Postgres prod)
- **Auth**: Auth.js (NextAuth) - Email + Google OAuth
- **Payments**: Stripe Checkout + Webhooks
- **AI**: Provider-agnostic with local mock fallback
- **Media**: FFmpeg for video processing, Puppeteer for image generation
- **Testing**: Vitest (unit) + Playwright (e2e)

Quick Start

Prerequisites

- Node.js 18+
- FFmpeg installed locally
- Git

Installation

```bash
# Clone the repository
git clone https://github.com/yourusername/adgener-ai.git
cd adgener-ai

# Install dependencies
npm install

# Set up environment variables
cp .env.example .env.local

# Set up the database
npx prisma db push
npx prisma db seed

# Start development server
npm run dev
```

Visit `http://localhost:3000` to see the application.

Environment Variables

Variable	Description	Required	Default
`DATABASE_URL`	Database connection string	Yes	`file:./dev.db`
`NEXTAUTH_SECRET`	Auth.js secret key	Yes	-
`NEXTAUTH_URL`	Application base URL	Yes	`http://localhost:3000`
`GOOGLE_CLIENT_ID`	Google OAuth client ID	No	-
`GOOGLE_CLIENT_SECRET`	Google OAuth client secret	No	-
`STRIPE_SECRET_KEY`	Stripe secret key	No	-
`STRIPE_PUBLISHABLE_KEY`	Stripe publishable key	No	-
`STRIPE_WEBHOOK_SECRET`	Stripe webhook secret	No	-
`AI_PROVIDER`	AI provider config	No	`image:mock,video:mock`
`OPENAI_API_KEY`	OpenAI API key	No	-
`REPLICATE_API_TOKEN`	Replicate API token	No	-
`STORAGE_PROVIDER`	Storage provider	No	`local`
`S3_BUCKET`	S3 bucket name (if using S3)	No	-

Variable	Description	Required	Default
`S3_REGION`	S3 region	No	-
`S3_ACCESS_KEY`	S3 access key	No	-
`S3_SECRET_KEY`	S3 secret access key	No	-

Available Scripts

```bash
npm run dev        # Start development server
npm run build      # Build for production
npm run start      # Start production server
npm run lint       # Run ESLint
npm run type-check # Run TypeScript type checking
npm run test       # Run unit tests
npm run test:e2e   # Run end-to-end tests
npm run db:push    # Push database schema changes
npm run db:seed    # Seed database with initial data
npm run format     # Format code with Prettier
```

AI Providers

The application supports multiple AI providers with automatic fallback:

Local Mock Provider (Default)

- **Images**: Uses Puppeteer to render React templates as PNG
- **Videos**: Uses FFmpeg to create videos from templates
- No API keys required - works out of the box

OpenAI Images

- Set `AI_PROVIDER=image:openai,video:mock`
- Requires `OPENAI_API_KEY`
- Uses DALL-E for image generation

Replicate Video

- Set `AI_PROVIDER=image:mock,video:replicate`
- Requires `REPLICATE_API_TOKEN`
- Uses various video generation models

Stripe Setup (Development)

1. Create a Stripe account and get your test keys
2. Add keys to `.env.local`
3. Install Stripe CLI: `stripe login`
4. Forward webhooks: `stripe listen --forward-to localhost:3000/api/stripe/webhook`
5. Copy webhook secret to `STRIPE_WEBHOOK_SECRET`

Testing

Unit Tests

```bash
npm run test
```

E2E Tests

```bash
# Start the application first
npm run dev

# In another terminal
npm run test:e2e
```

SELF-CHECK

After setup, verify these work:

✅ Basic Functionality

1. **App loads**: Visit `http://localhost:3000` - should see landing page
2. **Auth works**: Click "Get Started" → login with email or Google
3. **Dashboard**: After login, should see dashboard with token balance (10)
4. **Templates load**: Navigate to Templates tab - should see 8 template cards

✅ Mock Generation (No API Keys)

1. **Photo generation**:
   - Templates → Real Estate template → "Use Template"

- Fill form → "Generate Photo" → should create image in ~3-5s
- Check `/public/outputs/` folder for generated PNG
- Token balance should decrease by 1

2. **Video generation**:
   - Generate → Video → Select template → Fill form
   - "Generate Video" → should create MP4 in ~10-15s
   - Token balance should decrease by 5

✅ Library & Management

3. **Asset library**: Library tab shows generated assets
4. **Filters work**: Filter by Photo/Video, search by title
5. **Downloads**: Click download icon on any asset
6. **Mobile responsive**: Resize window - sidebar becomes drawer

✅ Stripe Integration (Test Mode)

7. **Token purchase**: Account → "Buy Tokens" → test card `4242 4242 4242 4242`
8. **Webhook**: If using `stripe listen`, tokens should be credited automatically

✅ Expected File Outputs

- **Generated photos**: `/public/outputs/photo_[timestamp].png`
- **Generated videos**: `/public/outputs/video_[timestamp].mp4`
- **Thumbnails**: `/public/outputs/thumb_[timestamp].jpg`
- **Database**: `prisma/dev.db` with seeded templates and user data

Architecture

Token System

- New users get 10 free tokens
- Photo generation: 1 token
- Video generation: 5 tokens
- Tokens deducted before generation, refunded on failure
- All transactions logged for audit

Template System

- Industry-specific templates with customizable fields
- Each template defines layout, aspect ratio, and editable fields
- Preview thumbnails and metadata stored in database

AI Provider Architecture

- Abstracted interface for swappable AI providers
- Local mock provider for development without API costs
- Graceful fallback system if external providers fail

Deployment

Vercel (Recommended)

1. Push to GitHub
2. Import to Vercel
3. Add environment variables
4. Deploy

Railway/Render

1. Connect GitHub repo
2. Add environment variables
3. Ensure FFmpeg is available in runtime

Docker

```bash
docker build -t adgener-ai .
docker run -p 3000:3000 adgener-ai
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests

5. Submit a pull request

License

MIT License - see LICENSE file for details.

Support

- 📧 Email: support@adgener.ai
- 📖 Docs: Check `/docs` folder
- 🐛 Issues: GitHub Issues tab

Built with ❤️ using Next.js, TypeScript, and modern web technologies.