Content Alchemy

📋 Table of Contents

🎯 Overview

Content Alchemy is a cutting-edge web application that transforms content creation through modern technologies and AI-powered development. Built with an emphasis on type safety and developer experience, this project leverages the Lovable platform for rapid, intelligent application development.

Repository: https://github.com/johnwesly08/content-alchemy

Key Highlights:

- 96.5% TypeScript for maximum type safety
- AI-powered development workflow via Lovable
- Modern component architecture
- Production-ready build configuration

🛠 Technology Stack

Core Technologies

Technology	Version	Purpose
Vite	Latest	Ultra-fast build tool with instant HMR
TypeScript	Latest	Type-safe JavaScript (96.5% coverage)
React	Latest	Component-based UI library

UI & Design System

Technology	Purpose
shadcn-ui	Premium accessible component library
Tailwind CSS	Utility-first CSS framework (2.6% of codebase)
PostCSS	CSS transformation and autoprefixing

Development Tools

Tool	Purpose
Bun	Fast JavaScript runtime and package manager
ESLint	Code quality and consistency enforcement
Vitest	Lightning-fast unit testing framework

Platform

Platform	Features
Lovable	AI-powered development, instant deployment, automatic commits

📁 Project Structure

```
content-alchemy/
├── public/                # Static assets
│   ├── favicon.ico        # Site favicon
│   └── assets/            # Images, fonts, etc.
│
├── src/                   # Source code
│   ├── components/        # React components
│   │   ├── ui/            # shadcn-ui components
│   │   └── layout/        # Layout components
│   │
│   ├── pages/             # Page-level components
│   │   ├── Home.tsx
│   │   └── ...
│   │
│   ├── hooks/             # Custom React hooks
│   │   ├── useAuth.ts
│   │   └── ...
│   │
│   ├── lib/               # Utilities and helpers
│   │   ├── utils.ts
│   │   └── api.ts
│   │
│   ├── types/             # TypeScript type definitions
│   │   ├── index.ts
│   │   └── models.ts
│   │
│   ├── styles/            # Global styles
│   │   └── globals.css
│   │
│   ├── App.tsx            # Root component
│   └── main.tsx           # Application entry point
│
├── .gitignore             # Git ignore rules
├── components.json        # shadcn-ui configuration
├── eslint.config.js       # ESLint configuration
├── index.html             # HTML entry point
├── package.json           # Project dependencies
├── postcss.config.js      # PostCSS configuration
├── tailwind.config.ts     # Tailwind CSS configuration
├── tsconfig.json          # TypeScript base config
├── tsconfig.app.json      # App-specific TypeScript config
```

```
├── tsconfig.node.json    # Node.js TypeScript config
├── vite.config.ts        # Vite build configuration
└── vitest.config.ts      # Vitest test configuration
```

🚀 Getting Started

Prerequisites

Before starting, ensure you have:

- **Node.js** v18 or higher (Download)
  - Recommended: Use nvm for version management
- **npm** v9 or higher (bundled with Node.js)
- **Git** for version control

Quick Start

Follow these steps to get Content Alchemy running locally:

```bash
# 1. Clone the repository
git clone https://github.com/johnwesly08/content-alchemy.git

# 2. Navigate to project directory
cd content-alchemy

# 3. Install dependencies
npm install

# 4. Start development server
npm run dev
```

The application will be available at `http://localhost:5173`

Verify Installation

After running `npm run dev`, you should see output similar to:

```
VITE v5.x.x  ready in xxx ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

💻 Development

Available Scripts

Content Alchemy includes several npm scripts for development and production:

```bash
# Development
npm run dev              # Start dev server with hot reload
npm run dev -- --host    # Expose dev server on network

# Building
npm run build            # Build for production
npm run preview          # Preview production build locally

# Code Quality
npm run lint             # Run ESLint
npm run lint:fix         # Auto-fix ESLint issues
npm run type-check       # Check TypeScript types

# Testing
npm run test             # Run tests
npm run test:watch       # Run tests in watch mode
npm run test:coverage    # Generate coverage report
npm run test:ui          # Open Vitest UI
```

Development Workflow Options

Option 1: Lovable Platform (Recommended for Rapid Development)

The Lovable platform provides AI-powered development with instant feedback:

1. **Access Your Project**
   - Visit Lovable Project
2. **AI-Powered Development**

- Use natural language to describe features
- Get instant implementation suggestions
- See real-time previews

3. **Automatic Syncing**
    - Changes are automatically committed to GitHub
    - No manual Git operations needed
    - Full version history maintained

Best For:

- Rapid prototyping
- Feature iteration
- UI/UX experimentation
- Non-developers or beginners

Option 2: Local Development with IDE

Traditional development workflow for maximum control:

1. **Setup Your Environment**

```bash
# Clone and setup
git clone https://github.com/johnwesly08/content-alchemy.git
cd content-alchemy
npm install
```

2. **Start Development**

```bash
npm run dev
```

3. **Make Changes**
    - Edit files in your preferred IDE
    - Save and see instant hot-reload
    - Test your changes locally

4. **Commit and Push**

```bash
git add .
git commit -m "feat: add new feature"
git push origin main
```

Recommended IDE Setup:

- **VS Code** with extensions:
  - ESLint
  - Prettier
  - Tailwind CSS IntelliSense
  - TypeScript Vue Plugin (Volar)

Best For:

- Complex features
- Multiple file changes
- Refactoring
- Advanced debugging

Option 3: GitHub Web Editor

Quick edits without local setup:

1. Navigate to any file in the repository
2. Click the **Edit** (pencil) icon
3. Make changes in the web editor
4. Commit directly or create a pull request

Best For:

- Quick fixes
- Documentation updates
- Configuration changes
- Small tweaks

Option 4: GitHub Codespaces

Full cloud-based development environment:

1. **Launch Codespace**
    - Go to the repository
    - Click **Code → Codespaces → New codespace**
2. **Develop in the Cloud**
    - VS Code runs in your browser
    - Full terminal access
    - All dependencies pre-installed
3. **Commit and Push**
    - Use built-in Git tools
    - Changes sync to repository

Best For:

- Working on different devices
- Avoiding local setup
- Testing in isolated environment
- Collaboration

Code Style Guidelines

Content Alchemy follows strict code quality standards:

TypeScript Standards

```typescript
```

```typescript
// ✅ Good - Use explicit types
interface User {
  id: string;
  name: string;
  email: string;
}

function getUser(id: string): Promise<User> {
  return fetchUser(id);
}

// ❌ Bad - Avoid 'any'
function getUser(id: any): any {
  return fetchUser(id);
}
```

Component Structure

typescript

```typescript
// ✅ Good - Functional components with TypeScript
import { FC } from 'react';

interface ButtonProps {
  label: string;
  onClick: () => void;
  disabled?: boolean;
}

export const Button: FC<ButtonProps> = ({ label, onClick, disabled = false }) => {
  return (
    <button onClick={onClick} disabled={disabled}>
      {label}
    </button>
  );
};
```

Import Organization

typescript

```
// ✅ Good - Organized imports
// 1. External libraries
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';

// 2. Internal utilities
import { formatDate } from '@/lib/utils';

// 3. Components
import { Button } from '@/components/ui/button';
import { Card } from '@/components/ui/card';

// 4. Types
import type { User } from '@/types';
```

Adding Components

Using shadcn-ui CLI

bash

```
# Add a single component
npx shadcn-ui@latest add button

# Add multiple components
npx shadcn-ui@latest add card dialog toast

# List available components
npx shadcn-ui@latest add
```

Creating Custom Components

typescript

```tsx
// src/components/CustomCard.tsx
import { FC, ReactNode } from 'react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';

interface CustomCardProps {
  title: string;
  children: ReactNode;
}

export const CustomCard: FC<CustomCardProps> = ({ title, children }) => {
  return (
    <Card>
      <CardHeader>
        <CardTitle>{title}</CardTitle>
      </CardHeader>
      <CardContent>{children}</CardContent>
    </Card>
  );
};
```

🌐 Deployment

Deploy with Lovable (One-Click Deployment)

The fastest way to deploy your application:

1. **Open Lovable Dashboard**
   - Visit Lovable Project
2. **Publish Your App**
   - Click **Share → Publish**
   - Your app is instantly deployed
   - Receive a public URL immediately
3. **Features Include:**
   - ✅ Automatic SSL certificates
   - ✅ Global CDN distribution
   - ✅ Instant deployments
   - ✅ Zero configuration
   - ✅ Automatic builds on commits

Custom Domain Setup

Connect your own domain to your Lovable deployment:

1. **Navigate to Domain Settings**
   - Go to **Project → Settings → Domains**
2. **Add Your Domain**
   - Click **Connect Domain**
   - Enter your domain name (e.g., `contentalchemy.com`)
3. **Configure DNS**
   - Add CNAME record pointing to Lovable
   - Or update A records as instructed
4. **Verify and Deploy**
   - SSL certificate provisioned automatically
   - Domain activated within minutes

📚 **Full Guide:** Lovable Custom Domain Documentation

Alternative Deployment Options

Vercel

```bash
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel

# Production deployment
vercel --prod
```

Environment Variables: Add in Vercel dashboard under Settings → Environment Variables

Netlify

```bash
```

```
# Install Netlify CLI
npm install -g netlify-cli

# Deploy
netlify deploy

# Production deployment
netlify deploy --prod
```

Build Settings:

- Build command: `npm run build`
- Publish directory: `dist`

Manual Deployment

```bash
# Build the project
npm run build

# The 'dist' folder contains production files
# Upload to any static hosting:
# - AWS S3 + CloudFront
# - DigitalOcean Spaces
# - GitHub Pages
# - Cloudflare Pages
```

⚙️ Configuration

Environment Variables

Create a `.env` file in the root directory:

```env
```

```
# API Configuration
VITE_API_URL=https://api.example.com
VITE_API_KEY=your_api_key_here

# Feature Flags
VITE_ENABLE_ANALYTICS=true
VITE_ENABLE_DEBUG=false

# Third-party Services
VITE_STRIPE_PUBLIC_KEY=pk_test_xxx
VITE_GOOGLE_ANALYTICS_ID=GA-XXXXXXXXX
```

Important:

- Prefix all variables with `VITE_`
- Never commit `.env` to version control
- Use `.env.example` for documentation

TypeScript Configuration

Content Alchemy uses three TypeScript configurations:

`tsconfig.json` (Base)

```json
{
  "compilerOptions": {
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  }
}
```

`tsconfig.app.json` (Application)

json

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "composite": true,
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "jsx": "react-jsx",
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"]
    }
  },
  "include": ["src"]
}
```

Tailwind CSS Configuration

Customize your design system in `tailwind.config.ts`:

typescript

```ts
import type { Config } from 'tailwindcss';

export default {
  darkMode: ['class'],
  content: [
    './pages/**/*.{ts,tsx}',
    './components/**/*.{ts,tsx}',
    './app/**/*.{ts,tsx}',
    './src/**/*.{ts,tsx}',
  ],
  theme: {
    extend: {
      colors: {
        border: 'hsl(var(--border))',
        input: 'hsl(var(--input))',
        ring: 'hsl(var(--ring))',
        background: 'hsl(var(--background))',
        foreground: 'hsl(var(--foreground))',
        primary: {
          DEFAULT: 'hsl(var(--primary))',
          foreground: 'hsl(var(--primary-foreground))',
        },
        // Add custom colors
        brand: {
          50: '#f0f9ff',
          100: '#e0f2fe',
          // ... more shades
        },
      },
      fontFamily: {
        sans: ['Inter', 'system-ui', 'sans-serif'],
        mono: ['Fira Code', 'monospace'],
      },
      spacing: {
        '128': '32rem',
        '144': '36rem',
      },
    },
  },
  plugins: [require('tailwindcss-animate')],
} satisfies Config;
```

Vite Configuration

Build optimizations in `vite.config.ts`:

```typescript
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import path from 'path';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
    },
  },
  build: {
    rollupOptions: {
      output: {
        manualChunks: {
          vendor: ['react', 'react-dom'],
          ui: ['@radix-ui/react-dialog', '@radix-ui/react-dropdown-menu'],
        },
      },
    },
  },
  server: {
    port: 5173,
    open: true,
  },
});
```

🎨 Features

Current Features

Based on the project structure and technology stack:

Core Capabilities

- ⚡ **Lightning-Fast Performance** - Vite-powered development and builds
- 🎯 **Type Safety** - 96.5% TypeScript coverage for robust code
- 🎨 **Modern UI** - shadcn-ui component library with Tailwind CSS

- 📱 **Responsive Design** - Mobile-first approach
- 🔄 **Hot Module Replacement** - Instant feedback during development
- 🧪 **Testing Ready** - Vitest configured for unit and integration tests

Developer Experience

- 🤖 **AI-Powered Development** - Lovable platform integration
- 📦 **Component Library** - Extensible shadcn-ui components
- 🎯 **Path Aliases** - Clean imports with `@/` prefix
- 🔍 **TypeScript IntelliSense** - Full autocomplete support
- 🛠️ **Modern Tooling** - ESLint, PostCSS, Bun support

Production Features

- 🚀 **Optimized Builds** - Code splitting and tree shaking
- 🔒 **Secure by Default** - Type-safe APIs and validation
- 📊 **Performance Monitoring** - Built-in optimization
- 🌍 **SEO Friendly** - Meta tags and semantic HTML
- ♿ **Accessibility** - WCAG 2.1 compliant components

🏆 Best Practices

Component Design

```typescript
```

```typescript
// ✅ Good - Composable, reusable components
interface CardProps {
  title: string;
  description?: string;
  children: React.ReactNode;
  className?: string;
}

export const ContentCard: FC<CardProps> = ({
  title,
  description,
  children,
  className,
}) => {
  return (
    <Card className={cn('p-6', className)}>
      <CardHeader>
        <CardTitle>{title}</CardTitle>
        {description && <CardDescription>{description}</CardDescription>}
      </CardHeader>
      <CardContent>{children}</CardContent>
    </Card>
  );
};
```

State Management

typescript

```typescript
// ✅ Good - Custom hooks for state logic
import { useState, useEffect } from 'react';

interface User {
  id: string;
  name: string;
}

export const useUser = (userId: string) => {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

  useEffect(() => {
    const fetchUser = async () => {
      try {
        const response = await fetch(`/api/users/${userId}`);
        const data = await response.json();
        setUser(data);
      } catch (err) {
        setError(err as Error);
      } finally {
        setLoading(false);
      }
    };

    fetchUser();
  }, [userId]);

  return { user, loading, error };
};
```

Error Handling

typescript

```
// ✅ Good - Comprehensive error handling
import { toast } from '@/components/ui/use-toast';

const handleSubmit = async (data: FormData) => {
  try {
    const response = await api.post('/endpoint', data);

    if (!response.ok) {
      throw new Error('Failed to submit');
    }

    toast({
      title: 'Success',
      description: 'Data submitted successfully',
    });
  } catch (error) {
    console.error('Submission error:', error);
    toast({
      variant: 'destructive',
      title: 'Error',
      description: 'Failed to submit data. Please try again.',
    });
  }
};
```

🤝 Contributing

We welcome contributions! Here's how to get started:

Contribution Workflow

1. **Fork the Repository**

```
bash

  # Click 'Fork' on GitHub, then clone your fork
  git clone https://github.com/YOUR_USERNAME/content-alchemy.git
```

2. **Create a Feature Branch**

```
bash
```

```
git checkout -b feature/amazing-feature
```

3. **Make Your Changes**
    - Write clean, documented code
    - Follow existing patterns
    - Add tests for new features
4. **Test Your Changes**

```bash
npm run lint
npm run type-check
npm run test
npm run build
```

5. **Commit Your Changes**

```bash
git add .
git commit -m "feat: add amazing feature"
```

6. **Push and Create PR**

```bash
git push origin feature/amazing-feature
# Then create a Pull Request on GitHub
```

Commit Message Convention

We follow Conventional Commits:

```
<type>(<scope>): <subject>

<body>

<footer>
```

Types:

- `feat`: New feature
- `fix`: Bug fix
- `docs`: Documentation changes
- `style`: Code style changes (formatting, etc.)
- `refactor`: Code refactoring
- `test`: Adding or updating tests
- `chore`: Maintenance tasks

Examples:

```bash
feat(auth): add OAuth login
fix(ui): resolve mobile menu overflow
docs(readme): update installation instructions
refactor(api): simplify error handling
test(hooks): add useUser hook tests
```

Code Review Guidelines

All contributions require review:

- ✅ Code follows project style
- ✅ Tests pass and coverage maintained
- ✅ Documentation updated
- ✅ No console errors or warnings
- ✅ TypeScript types are correct
- ✅ Responsive on mobile devices

🔧 Troubleshooting

Common Issues and Solutions

Build Errors

Problem: `npm install` fails

```bash
```

```
# Solution 1: Clear cache
npm cache clean --force
rm -rf node_modules package-lock.json
npm install

# Solution 2: Use exact Node version
nvm install 18
nvm use 18
npm install
```

Problem: TypeScript errors during build

```
bash

# Check TypeScript configuration
npm run type-check

# Regenerate TypeScript cache
rm -rf node_modules/.cache
npm run build
```

Development Server Issues

Problem: Port 5173 already in use

```
bash

# Use a different port
npm run dev -- --port 3000

# Or find and kill process using port 5173
lsof -ti:5173 | xargs kill -9  # macOS/Linux
netstat -ano | findstr :5173    # Windows
```

Problem: Changes not hot-reloading

```
bash
```

```
# Clear Vite cache
rm -rf node_modules/.vite
npm run dev

# If still not working, try hard refresh
# Ctrl+Shift+R (Windows/Linux)
# Cmd+Shift+R (macOS)
```

Module Resolution Issues

Problem: Cannot find module '@/...'

bash

```
# Verify tsconfig.json has correct paths
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"]
    }
  }
}

# Restart TypeScript server in VS Code
# Cmd+Shift+P → TypeScript: Restart TS Server
```

shadcn-ui Issues

Problem: Component styles not applying

bash

```
# Ensure Tailwind CSS is configured correctly
# Check if tailwind.config.ts includes:
content: [
  './src/**/*.{ts,tsx}',
]

# Restart dev server
npm run dev
```

Problem: Cannot add shadcn-ui components

```bash
# Update shadcn-ui CLI
npm install -g shadcn-ui@latest

# Verify components.json exists
cat components.json
```

Performance Issues

Problem: Slow build times

```bash
# Enable caching
# Update vite.config.ts:
export default defineConfig({
  cacheDir: 'node_modules/.vite',
  build: {
    target: 'esnext',
    minify: 'esbuild',
  },
})
```

Problem: Large bundle size

```bash
# Analyze bundle
npm run build
npx vite-bundle-visualizer

# Implement code splitting
# Use dynamic imports for large dependencies
const HeavyComponent = lazy(() => import('./HeavyComponent'));
```

Getting More Help

- 📖 **Documentation**: Check inline code comments
- 🐛 **Report Bugs**: GitHub Issues
- 💬 **Discussions**: GitHub Discussions
- 📚 **Lovable Docs**: https://docs.lovable.dev

- 🎯 **Vite Docs**: https://vitejs.dev
- ⚛️ **React Docs**: https://react.dev

📚 Resources

Official Documentation

- Lovable Documentation
- Vite Documentation
- React Documentation
- TypeScript Handbook
- Tailwind CSS Documentation
- shadcn-ui Documentation

Learning Resources

- React TypeScript Cheatsheet
- Tailwind CSS Best Practices
- Vite Plugin Development
- Testing with Vitest

Community

- Lovable Community
- React Community
- TypeScript Discord

📄 License

This project's license information is not specified. Please contact the repository owner for licensing details.

🙏 Acknowledgments

- **Lovable** - AI-powered development platform
- **shadcn** - Beautiful component library
- **Vercel** - Vite and React ecosystem
- **Tailwind Labs** - Tailwind CSS framework

Transform Your Content with Alchemy ✨

Built with ❤️ using Lovable | Last Updated: January 2026