# Smart Content Studio AI

Smart Content Studio AI is a full-stack workspace that helps creative strategists, content teams, and indie game builders ideate, refine, and ship ideas faster. It pairs an Apple-inspired glass UI with production-ready AI tools powered by Google Gemini 2.0 Flash, a Firebase-authenticated React frontend, and a FastAPI backend.

## Highlights

- **Unified creative cockpit** – Summarizer, Idea Generator, Content Refiner, Chatbot, GameForge, and Image Generator share one consistent dashboard.
- **Multi-model AI routing** – Smart provider selection across Google Gemini 2.0 Flash, xAI Grok, with automatic fallback for resilience.
- **Authentic AI output** – Every tool streams real responses from production AI APIs with markdown/code formatting.
- **Flexible image generation** – Pollinations AI for instant visuals, with FAL.ai as an alternative provider.
- **Apple-style sign-in** – Glassmorphism login with email/password flows plus Google OAuth via Firebase.
- **Responsive & accessible** – Tailwind-based layout adapts to any screen with sensible keyboard/focus states.
- **Resilient UX** – Error boundaries, loading shimmer, and optimistic messaging keep the experience polished.

## Tech Stack

### Frontend

- React 19 with hooks and Suspense-ready patterns
- React Router DOM for client routing
- Tailwind CSS + custom glass utilities
- Firebase Authentication (email/password + Google provider)
- React Markdown + Remark/rehype pipeline for formatted AI responses
- Framer Motion accents in creative tools

### Backend

- FastAPI + Uvicorn
- Multi-model AI routing:
  - Google Gemini 2.0 Flash (text generation)
  - xAI Grok (text generation with fallback)
  - Pollinations AI (image generation)
  - FAL.ai (optional image provider)
- Pydantic for validation, CORS middleware, `.env` driven configuration

## Directory Map

```
SM-content-React-app/
├── README.md                 # This document
├── DESIGN.md                # Product & UX decisions
├── package.json             # Frontend scripts & deps
├── tailwind.config.js       # Tailwind setup
├── postcss.config.js
├── public/                  # CRA static assets
├── src/
│   ├── App.js               # Routing + auth shell
│   ├── index.js
│   ├── index.css
│   ├── service-worker.js
│   ├── services/
│   │   ├── firebase.js      # Firebase bootstrap
│   │   └── realApi.js       # FastAPI client helpers
│   └── components/
│       ├── Chatbot.js
│       ├── ContentRefiner.js
│       ├── ErrorBoundary.js
│       ├── FormattedAIResponse.js
│       ├── GameForge.js
│       ├── Header.js
│       ├── IdeaGenerator.js
│       ├── ImageGenerator.js
│       ├── Loader.js
│       ├── Sidebar.js
│       └── Summarizer.js
└── backend/
    ├── main.py
    ├── requirements.txt
    ├── start.sh
    ├── README.md
    └── .env.template
```

## Prerequisites

- Node.js 18+ and npm
- Python 3.10+
- At least one AI API key:
  - Google Gemini API key from AI Studio
  - xAI Grok API key from X.AI Console
- Firebase project configured for Web auth (email/password + Google)

## Backend Setup

```
cd backend
python3 -m venv venv
source venv/bin/activate   # Windows: venv\Scripts\activate
```

```
pip install -r requirements.txt
cp .env.template .env      # add GEMINI_API_KEY and optional settings
uvicorn main:app --reload --host 0.0.0.0 --port 8000
```

Key environment variables (`backend/.env`):

```
# === AI Model Configuration ===
GEMINI_API_KEY=your-google-gemini-key
GROK_API_KEY=your-xai-grok-key

# Primary AI provider (gemini or grok)
PRIMARY_AI_PROVIDER=gemini

# Enable automatic fallback to secondary provider if primary fails
ENABLE_AI_FALLBACK=true

# === Image Generation ===
IMAGE_API_PROVIDER=pollinations  # or fal
IMAGE_API_KEY=                   # only for fal

ENVIRONMENT=development
```

**Routing behavior:**

- If `PRIMARY_AI_PROVIDER=gemini` and `ENABLE_AI_FALLBACK=true`, requests go to Gemini first, then Grok if Gemini fails.
- If `PRIMARY_AI_PROVIDER=grok`, Grok is tried first with optional Gemini fallback.
- If only one key is set, that provider is used exclusively.
- Image requests route to Pollinations by default (no auth required); set `IMAGE_API_PROVIDER=fal` and provide `IMAGE_API_KEY` for FAL.ai.

API docs are available at `http://localhost:8000/docs`.

## Frontend Setup

```
npm install
npm start
```

Optional `.env` keys (create `.env` in the project root):

```
REACT_APP_API_URL=http://localhost:8000
REACT_APP_FIREBASE_API_KEY=your-firebase-key
REACT_APP_FIREBASE_AUTH_DOMAIN=project.firebaseapp.com
(...other Firebase config values)
```

## Running the Stack Locally

1. Start the FastAPI service (see “Backend Setup”).
2. In a separate terminal, run `npm start` for the React dev server.
3. Visit `http://localhost:3000` for the UI and ensure the backend is reachable at `http://localhost:8000`.

## Available Scripts

- `npm start` – CRA development server with hot reload.
- `npm run build` – Production bundle (lints as part of the build).
- `npm run lint` – Optional script if added for static analysis (configure in `package.json`).
- `./backend/start.sh` – Helper script to bootstrap the API with one command.

## Feature Walkthrough

- **Login** – Email/password form with mode toggle and Google sign-in button, all wrapped in a glassmorphism hero that showcases `into.jpg`.
- **Dashboard shell** – `Sidebar` for navigation, `Header` for user info/sign-out, and glass `main` container hosting active tools.
- **Multi-model AI routing** – All text-generation endpoints intelligently route between Gemini and Grok with automatic failover if one provider is unavailable.
- **Summarizer / Idea Generator / Content Refiner** – Text-based workflows with saved history and markdown output.
- **Chatbot** – Streaming conversation with error-safe markdown rendering via `FormattedAIResponse`.
- **GameForge** – Multi-card experience for narrative, dialogue, mechanics tuning, and code snippets tailored to game creators.
- **Image Generator** – Framer Motion-enhanced prompt panel using Pollinations AI for quick visual ideation.

## Deployment Notes

Frontend:

```
npm run build
```

Deploy the `build/` folder to any static host (Netlify, Vercel, S3, etc.). Set `homepage` in `package.json` if hosting at a subpath.

Backend:

```
cd backend
uvicorn main:app --host 0.0.0.0 --port 8000
```