DesignGPT

AI-Native Design Platform

Author: Mausam Kar

ntation Release v1.0December 25, 2025

Contents

Chapter 1

Introduction

1.1 What is DesignGPT?

DesignGPT (formerly xdesign-ai) is a production-ready, AI-native design platform that transforms natural language descriptions into high-fidelity, editable UI mockups.

In the modern era of software development, bridging the gap between an idea and a visual prototype is often the most time-consuming step. DesignGPT aims to solve this by allowing users to simply *describe* their idea. The system's autonomous agents then analyze, plan, and generate a multi-screen mobile application design in real-time.

Core Value Proposition

Describe your idea, and DesignGPT's autonomous agents will analyze, plan, and generate a multi-screen mobile application design in real-time.

1.2 Key Capabilities

- **Text-to-UI**: Convert simple text prompts into full mobile UI designs.
- **Real-time Generation**: Watch as the interface is built component-by-component using WebSockets.
- **Editable Code**: The output is not just an image; it is valid, production-ready React and Tailwind CSS code.

1.3 Project Status

The project is currently active and evolving.

- **Version**: 0.1.0
- **Status**: Active/Success
- **License**: MIT

1.4 The Concept

The gap between a product idea and a tangible design is often the biggest bottleneck in software development. Traditional workflow involves:

1. Brainstorming ideas.
2. Hiring expensive UI/UX designers or spending hours in Figma.
3. Iterating on static mockups.
4. Manually translating designs into code.

DesignGPT eliminates these steps. By leveraging large language models (specifically Google Gemini 2.0 Flash) and a specialized agentic workflow, it automates the entire design-to-code pipeline.

1.5 Target Audience

- **Founders Entrepreneurs**: Quickly visualize MVPs to pitch to investors.
- **Developers**: Generate frontend scaffolding instantly, saving days of CSS work.
- **Product Managers**: Create high-fidelity prototypes for stakeholder review without needing design resources.

> "Design is not just what it looks like and feels like. Design is how it works." – Steve Jobs

DesignGPT embodies this by generating code that doesn't just look good, but works as a foundation for real applications.

1.6 The Long-Term Vision

DesignGPT is just the beginning of a larger shift in software engineering: **Agentic UI Development**.

Currently, we treat AI as a "Co-pilot" that helps us write small chunks of code. DesignGPT moves towards an "Autopilot" model where the AI acts as a **Senior Engineer**, capable of:

- Understanding high-level business requirements.
- Making architectural decisions (choosing themes, layouts).
- Executing complex, multi-step implementation plans.
- Self-correcting when the output doesn't match the vision.

1.7 Roadmap

We have an ambitious roadmap for the future of this platform:

1. **v0.2**: Support for connecting to external APIs (bringing the generated UI to life).
2. **v0.3**: Export to React Native for instant mobile app deployment.
3. **v1.0**: Full "Figma-to-Code" bidirectional sync.

Chapter 2

Advanced Features

DesignGPT is packed with features designed to make the design process seamless and professional.

2.1 Intelligent Generation Engine

At the heart of DesignGPT lies a sophisticated AI engine powered by **Google Gemini 2.0 Flash**.

- **Context-Aware Design**: The system analyzes your request to determine consistency with existing screens, ensuring a cohesive look and feel across the entire application.
- **Automated Planning**: Before writing a single line of code, the AI explicitly plans the App Structure, defining User Flows and Screen Purposes.
- **Smart Remediation**: Need a change? The `regenerateFrame` function understands visual context, preserving your layout while applying specific edits.

2.1.1 Deep Dive: The Logician Agent

The "Logician" agent is responsible for the initial analysis. It uses a Zod schema to enforce structured output, ensuring that every generated design has clarity and purpose.

Analysis Schema

The agent outputs a JSON object containing:

- **Theme**: Visual theme ID (e.g., 'midnight', 'ocean-breeze').
- **Screens**: An array of objects, each with a unique ID, descriptive name, purpose statement, and a dense visual description.

The `visualDescription` field is critical. It acts as a high-fidelity directive for the subsequent creative agent, describing layout, specific data examples, component hierarchy, and physical attributes like "Chunky cards" or "Floating header".

2.1.2 Deep Dive: The Creative Agent

Once the plan is set, the "Creative" agent takes over. It iterates through the planned screens, generating the actual HTML/Tailwind code. Crucially, it is **state-aware**. It receives the context of all previously generated screens to ensure consistency in navigation, typography, and spacing.

2.2 Design Tools Integration

DesignGPT isn't limited to just its own internal knowledge. It reaches out to the world to make designs realistic.

2.2.1 Tool Use: Unsplash API

To avoid the "lorem ipsum" placeholder effect, the agents are equipped with the `searchUnsplash` tool.

1. The agent infers the context (e.g., "Sushi Restaurant").
2. It calls the tool with a query like `"california roll high res"`.
3. The returned image URL is directly embedded into the `<img>` tags of the generated design.

This results in designs that feel "lived-in" and finished from the very first moment.

2.3 Real-Time Experience

We prioritize user feedback loops over raw speed.

2.3.1 Optimistic UI Websockets

While the backend (Inngest) processes the heavy AI workload, the frontend remains responsive.

- **Step-by-Step Feeds**: As soon as a screen is generated, it is pushed to the client. The user doesn't wait for the entire app to be finished before seeing the first screen.
- **Live Status**: The "Analyzer" and "Generator" states are broadcasted, so the user sees exactly what the AI is "thinking" (e.g., "Planning Navigation", "Selecting Theme").

2.4 Security First

Integrated with **Kinde Auth**, the platform ensures:

- Secure OAuth2.0 authentication.
- JWT-based session management.
- User-scoped data isolation (users can only see their own projects).

2.5 Professional Design System

We believe that AI-generated designs should essentially look handcrafted by a top-tier designer.

- **Dynamic Theming**: The system automatically selects and applies visual themes (e.g., Midnight, Ocean Breeze) or extracts them directly from your prompt.
- **Tailwind v4 Native**: All generated code is pure, utility-first Tailwind CSS. This makes it instantly copy-pasteable into your own projects without complex build configurations.
- **Unsplash Integration**: Through tool calling, the agent automatically populates designs with high-quality, relevant stock imagery from Unsplash.

2.6 Enterprise-Grade Architecture

Built for scale and reliability.

- **Event-Driven Backend**: Heavy AI tasks are offloaded to **Inngest** serverless queues, ensuring the UI never freezes even during complex generations.
- **Real-Time Websockets**: Users can watch their designs appear component-by-component with optimistic UI updates.
- **Secure Authentication**: Fully integrated with **Kinde** for secure, passwordless authentication flows.

Chapter 3

System Architecture

DesignGPT employs a specialized multi-agent workflow to ensure high-quality output. The architecture is designed to be event-driven, scalable, and resilient.

3.1 The Generation Pipeline

The following diagram illustrates the flow of data from the initial user request to the final UI generation.

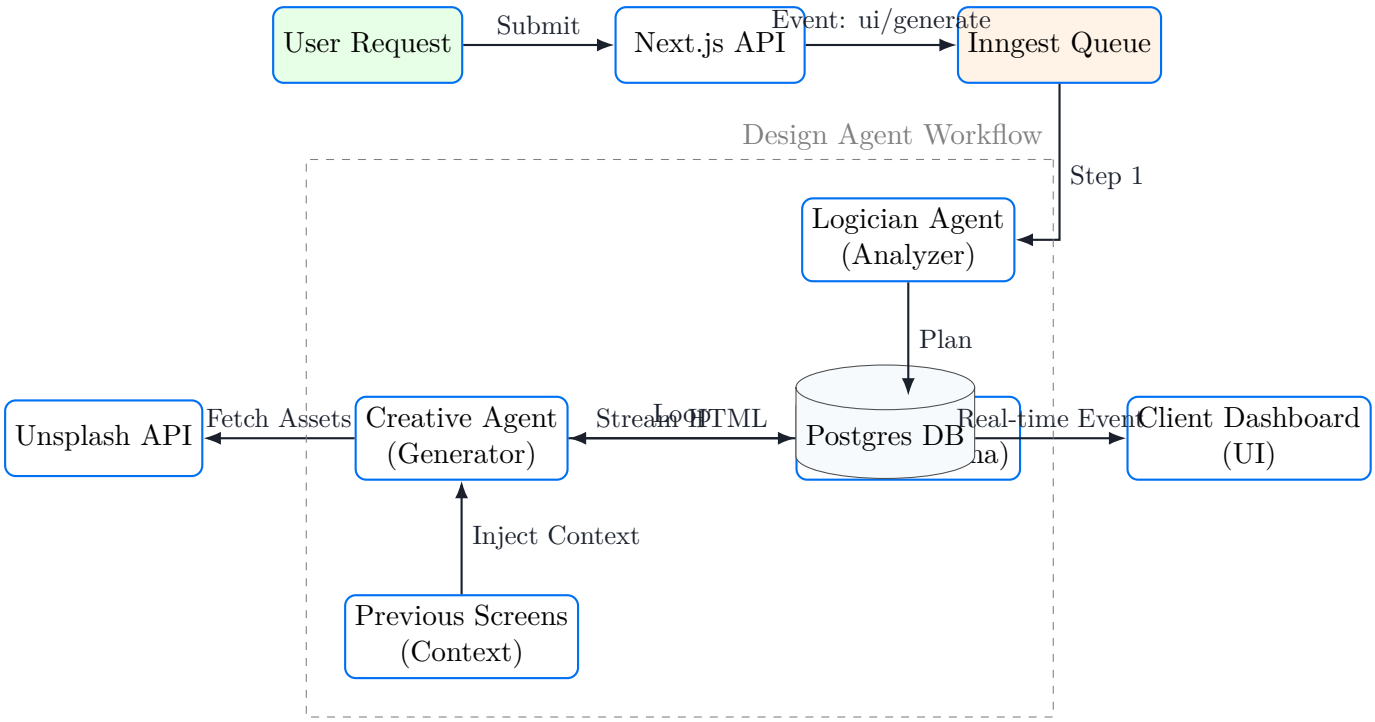

Figure 3.1: DesignGPT Generation Pipeline Architecture

3.2 Workflow Details

1. **Submission**: The user submits a prompt via the Client Dashboard.
2. **Queueing**: The API triggers an Inngest event, offloading the work to a background worker.

3. **Analysis**: The *Logician Agent* analyzes the prompt and existing screens to ensure consistency.
4. **Planning**: A plan is generated in JSON format, outlining specific screens and their purposes.
5. **Generation**: The *Creative Agent* executes the plan, iterating through screens. It fetches assets from Unsplash and maintains context from previous generations.
6. **Streaming**: The generated code is saved to the Postgres database, which pushes updates to the Client Dashboard in real-time.

3.3 Code Spotlight: The Inngest Function

The core orchestration happens within ‘inngest/functions/generateScreens.ts‘. This function defines the multi-step workflow.

Listing 3.1: Inngest Step Function Structure

```
export const generateScreens = inngest.createFunction(
  { id: "generate-ui-screens" },
  { event: "ui/generate.screens" },
  async ({ event, step, publish }) => {
    // 1. Analyze and Plan
    const analysis = await step.run("analyze-and-plan-screens", async
        () => {
      // ... calls Gemini with AnalysisSchema
    });

    // 2. Execution Loop
    for (let i = 0; i < analysis.screens.length; i++) {
      await step.run(‘generated-screen-${i}‘, async () => {
         // ... calls Gemini with Tool Use (Unsplash)
         // ... Generates HTML
         // ... Saves to Prisma DB
      });
    }
  }
);
```

This ‘step.run‘ architecture is vital. It allows the long-running generation process (which can take 30-60 seconds for multiple screens) to be reliable. If the generation of Screen 2 fails, Inngest retries *only* Screen 2, without re-running the analysis or regenerating Screen 1.

3.4 Data Models

The application relies on a robust schema to track projects and frames.

- **Project**: The container for a design session. Stores the ‘theme‘ and ownership ‘userId‘.
- **Frame**: Represents a single screen. Stores the ‘htmlContent‘ and ‘title‘.

Chapter 4

Technology Stack

This project leverages a bleeding-edge stack designed for **speed**, **type-safety**, and **autonomous agent capabilities**.

4.1 Core Framework: Next.js 16 (React 19)

- **Purpose**: Provides the hybrid runtime for Server Components (RSC) and Client interactive islands.
- **Implementation**:
  - Uses **Server Actions** (`app/action/action.ts`) for mutation logic.
  - Leverages **App Router** for nested layouts.
  - React 19 features are used for optimized rendering.

4.2 Styling Engine: Tailwind CSS 4.0

- **Purpose**: Zero-runtime styling with a utility-first approach.
- **Implementation**:
  - **Native CSS Variables**: The theming system injects CSS variables directly into the DOM.
  - **JIT Compilation**: Instant compilation allowing the AI to generate arbitrary, valid classes.

4.3 AI & Orchestration

Google Gemini 2.0 Flash + Inngest

- **Gemini Flash**: Offers the best reasoning-to-latency ratio for complex layout generation.
- **Inngest**: Manages long-running, multi-step generation workflows.
- **Tool Calling**: AI has access to external tools like `searchUnsplash`.

4.4 Database & State

4.5 Database State

Prisma + MongoDB

The project uses Prisma as an ORM. While the typical stack is PostgreSQL, we use **MongoDB** for its flexibility with JSON-like document structures which are common in AI responses.

4.5.1 Schema Design

The data model is deliberately simple to reduce friction.

Listing 4.1: Prisma Schema

```
model Project {
    id         String @id @default(auto()) @map("_id") @db.ObjectId
    userId     String
    name       String
    theme      String?
    thumbnail  String?
    frames     Frame[]

    createdAt  DateTime @default(now())
    updatedAt  DateTime @updatedAt
}

model Frame {
    id          String   @id @default(auto()) @map("_id") @db.ObjectId
    title       String
    htmlContent String   // Stores the raw HTML output
    projectId   String
    project     Project  @relation(fields: [projectId], references: [id
        ])

    createdAt   DateTime @default(now())
    updatedAt   DateTime @updatedAt
}
```

- **Project**: Top-level container. One-to-many relationship with Frames.
- **Frame**: Stores the generated HTML blob. We store full HTML strings rather than JSON components.

4.6 Authentication

Kinde

Kinde was chosen over NextAuth for better developer experience with features like:

- Built-in User Management Dashboard.
- "Passwordless" magic link support out of the box.
- Easier token handling for server-side API calls to pass user context to Inngest.

Chapter 5

Getting Started Guide

5.1 Prerequisites

Before you begin, ensure you have the following installed:

- Node.js 18+
- PostgreSQL Database URL
- Kinde Auth Account
- OpenRouter API Key

5.2 Installation

1. **Clone the repository**

```
git clone https://github.com/mausamkar/xdesign-ai.git
cd xdesign-ai
```

2. **Install dependencies**

```
npm install
```

5.3 Configuration

Create a `.env` file in the root directory and configure your keys:

```
# Database
DATABASE_URL="postgresql://user:password@localhost:5432/designgpt"

# Authentication (Kinde)
KINDE_CLIENT_ID="your_client_id"
KINDE_CLIENT_SECRET="your_client_secret"
KINDE_ISSUER_URL="https://your-app.kinde.com"
KINDE_SITE_URL="http://localhost:3000"
KINDE_POST_LOGOUT_REDIRECT_URL="http://localhost:3000"
KINDE_POST_LOGIN_REDIRECT_URL="http://localhost:3000/dashboard"

# AI & Services
OPENROUTER_API_KEY="sk-or-..."
INNGEST_EVENT_KEY="test_key"
INNGEST_SIGNING_KEY="test_signing_key"
```

5.4 Running the Application

DesignGPT requires two concurrent processes:

1. **The Application (Next.js)**

```
npm run dev
# Running at http://localhost:3000
```

2. **The Worker (Inngest)**

```
npx inngest-cli@latest dev
# Running at http://localhost:8288
```

Ready to Design

Once both processes are running, navigate to `http://localhost:3000` to start generating designs!

5.5 Troubleshooting

Here are some common issues you might encounter:

5.5.1 Database Connection Errors

If you see `PrismaClientInitializationError`:

- Ensure your MongoDB connection string in `.env` is correct.
- If using a local DB, ensure the service is running.
- Verify that you have run `npx prisma db push` to sync the schema.

5.5.2 Inngest Not Picking Up Events

If the AI generation never starts:

- Check that the Inngest Dev Server is running in a separate terminal.
- Ensure the application is pointing to the local Inngest instance (default behavior in dev).
- Check the "Events" tab in the Inngest dashboard (`http://localhost:8288`) to see if events like `ui/generate.screens` are being received.

5.5.3 AI Timeouts

If the generation hangs:

- Large prompts may take longer than the default timeout.
- Check your OpenRouter credits/API key status.