🌸 PetalClone - Vision-Powered AI Website Cloner

Completed in 8 hours including documentation

PetalClone is an advanced AI-powered tool that clones websites with exceptional visual and structural accuracy. It leverages a vision-capable AI agent, a robust browser automation backend, and a real-time frontend to deconstruct, understand, and recreate websites from just a URL.

The system goes beyond simple HTML scraping. It uses a headless browser to capture a pixel-perfect screenshot and render JavaScript, then feeds this visual information to a multi-step AI agent. The agent analyzes the layout and style like a human developer would, ensuring the final clone is a high-fidelity replica of the original.

✨ Core Features

- **Vision-Powered Cloning**
  Uses GPT-4 Vision and a sophisticated prompt strategy to analyze screenshots for pixel-perfect layout, color, and typography replication.

- **Full Website Cloning**
  Discovers and clones an entire website by crawling all internal links up to a specified page limit.

- **Robust Scraping Engine**
  Built with **Playwright** to handle modern, JavaScript-heavy websites, capturing not just HTML but also computed styles and high-resolution screenshots.

- **Real-Time Progress**
  The UI features a live log stream (via SSE) that shows the agent's progress, from crawling and scraping to AI-powered code generation.

- **Interactive Results View**
  A resizable multi-panel interface allows you to compare the original screenshot, the live preview, and the generated code side-by-side.

- **Asset Handling**
  Downloads all site assets (CSS, JS, images) and rewrites links for a fully self-contained, offline-ready clone.

- **Downloadable Archives**
  Packages the entire cloned site into a convenient `.zip` file for download.

- **Multi-Model Support**
  Easily configurable to use different large language models (e.g., Claude 3.5 Sonnet, GPT-4o, Gemini).

🏗️ Technical Architecture

PetalClone consists of a FastAPI backend that orchestrates the cloning process and a Next.js frontend that provides a rich, interactive user experience.

1. **Job Request**: The user submits a URL through the Next.js frontend.
2. **Orchestration**: The FastAPI backend kicks off a cloning job.
3. **Site Crawling (Full Site Mode)**: A `SiteCrawler` discovers all accessible pages on the target domain.
4. **Scraping**: The `PlaywrightScraper` launches a headless browser for each page, executes JavaScript, and takes a full-page screenshot.
5. **AI Vision Cloning**:
   - Deconstructs the page into logical components
   - Extracts a design system (colors, fonts)
   - Generates a pixel-perfect HTML clone with embedded CSS
6. **Live Logging**: SSE streams real-time logs to the frontend.
7. **Asset Packaging**: Assets are downloaded and bundled into a `.zip` archive.

🧠 Tech Stack

Component	Technology	Purpose

Frontend	Next.js (App Router), React, TypeScript, Tailwind CSS, Shadcn UI	Modern, interactive, real-time UI
Backend	FastAPI, Python, Uvicorn	High-performance API server and job orchestration
Scraping	Playwright	Headless browser automation for accurate scraping
AI Models	GPT-4 Vision, Claude 3.5 Sonnet, Gemini	Code generation and design analysis
Real-time	Server-Sent Events (SSE)	Live logging stream to UI
UI Layout	`react-resizable-panels`	Resizable panels for result comparison

🚀 Getting Started

Prerequisites

- Python 3.10+
- Node.js 20+ and `npm`
- API Keys for LLM providers and Hyperbrowser

---

1. Backend Setup

```
cd backend
python -m venv .venv
source .venv/bin/activate  # On Windows: .\.venv\Scripts\activate
pip install -r requirements.txt
pip install playwright
playwright install --with-deps
touch .env
```

Example `.env` file:

```
OPENAI_API_KEY="sk-..."
HYPERBROWSER_API_KEY="..."
ANTHROPIC_API_KEY="sk-ant-..."
GOOGLE_AI_API_KEY="..."
```

Start the backend:

```
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

---

2. Frontend Setup

```
cd frontend
npm install
npm run dev
```

Access the app at: http://localhost:3000

📝 Project Structure

```
orchids-challenge/
├── backend/
│   ├── app/
│   │   ├── core/        # Configuration
│   │   ├── models/      # Pydantic data models
│   │   ├── routers/     # API routes
│   │   └── services/    # AI, scraping, and business logic
│   ├── .env             # API keys (user-provided)
│   └── requirements.txt  # Python dependencies
├── frontend/
│   ├── public/          # Static files
│   ├── src/
│   │   ├── app/         # App Router pages
│   │   ├── components/   # Shadcn UI + custom components
│   │   └── lib/         # Utility functions, API wrapper
│   └── package.json     # Node packages
└── README.md
```

Built for the Orchids SWE Internship Challenge.
Developer: Sahil Jagtap

Let me know if you'd like this exported into a `.docx` or PDF version!