CoutureAI - Personalized Avatar & Clothing Visualization

Project Title:

Personalized Avatar & Clothing Visualization

Team Name: TechnoTrio

Team Members:

- Preethi Kamal Gajula
- Utprekshya Jena
- Goureddy Harshitha Reddy

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered web application that allows users to generate realistic images of custom clothing based on textual descriptions.

Key Points:

1. **Problem Statement:**
    - Traditional online shopping platforms lack the ability to offer personalized previews of clothing based on user descriptions.
    - This limitation often leads to customer dissatisfaction and increased return rates.

2. **Proposed Solution:** CoutureAI enables users to
    - Input detailed outfit descriptions.
    - Generate and visualize realistic clothing images using the Hugging Face API (Stable Diffusion model).
    - Download the generated images for further use.
3. **Target Users:**
    - **Online Shoppers:** Users who want to visualize custom outfits before making a purchase.
    - **Fashion Designers:** Professionals who need quick visual prototypes of their clothing designs.
    - **Retailers Offering Virtual Try-On Experiences:** Businesses providing customers with the ability to preview outfits on personalized avatars.
4. **Expected Outcome:**
    - A functional web application that allows users to describe and visualize custom clothing designs in real-time.

Phase-2: Requirement Analysis

Objective:

Analyze and outline the technical and functional requirements for the **CoutureAI** application.

Key Points:

1. **Technical Requirements:**
    - Programming Language: **Python**
    - Frontend: **Streamlit Web Framework**
    - Image Generation: **Hugging Face Stable Diffusion API**
    - Environment: **Anaconda (for virtual environment management)**
2. **Functional Requirements:**
    - **User Input Handling:** Accept and validate user input for outfit descriptions and Hugging Face API credentials.
    - **Image Generation:** Generate realistic images of outfits based on user descriptions using the Hugging Face Stable Diffusion model.
    - **Image Display:** Display high-quality, realistic images of the generated outfit within the Streamlit interface.
    - **Image Download:** Allow users to download the generated outfit image in PNG format for offline use

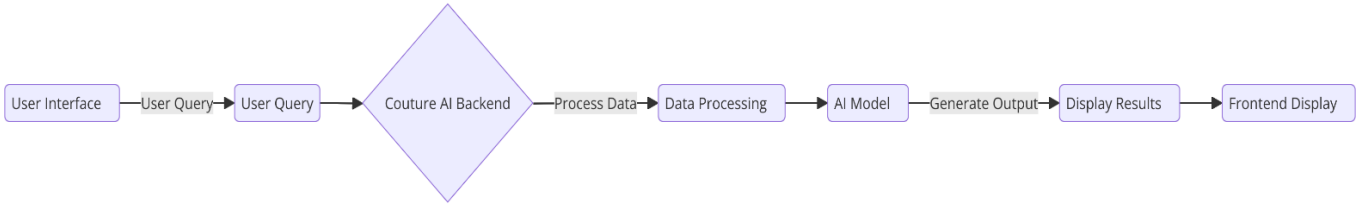
3. **Constraints & Challenges:**

    - Ensuring image accuracy and alignment with the provided text.
    - Handling **API rate limits** and optimizing API calls.
    - Providing a **smooth UI experience** with Streamlit.

Phase-3: Project Design

Objective:

Design the system architecture and user flow for CoutureAI.



Key Points:

1. **System Architecture:**
    - User inputs outfit descriptions and API credentials.
    - The backend sends requests to the **Hugging Face API** to generate images.
    - AI Generated images are displayed and made available for download.
2. **User Flow:**
    - Step 1: User inputs a clothing description and API token.
    - Step 2: The system validates the token and processes the input.
    - Step 3: The clothing image is generated and displayed.
    - Step 4: Generated images are available for download.
3. **UI/UX Considerations:**
    - **Minimalist, user-friendly interface** for seamless navigation.
    - **Intuitive input interface for outfit descriptions.**
    - **Clear display of generated images.**
    - **Download button for easy image access**

Phase-4: Project Planning (Agile Methodologies)

Objective:

Plan and execute the development process using Agile sprints.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Set up environment and integrate Hugging Face API.	□ High	6 hours (Day 1)	End of Day 1	Member 1	Hugging Face API , Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	□ Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with input fields
Sprint 2	Core Image Generation Feature	□ High	4 hours (Day 2)	Mid-Day 2	Member 1& 2	API response, UI elements ready	Functional image generation
Sprint 2	Error Handling & Debugging	□ High	1.5 hours (Day 2)	Mid-Day 2	Member 1&3	Image generation working	Clear error messaging
Sprint 3	Final Presentation & Deployment	□ Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(□ High Priority) Set up the **environment** & install dependencies.
(□ High Priority) Integrate **Hugging Face API**
(□ Medium Priority) Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 2)

(□ High Priority) Implement **Image generation** **(□ High Priority)** Debug API issues & handle **errors in queries**.

Sprint 3 – Testing, Enhancements & Submission (Day 2)

(□ Medium Priority) Test API responses, refine UI, & fix UI bugs.
(□ Low Priority) Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Implement and test the core functionality of the CoutureAI application.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit
    - **Backend:** Hugging Face API
    - **Programming Language:** Python
2. **Development Process:**
    - Set up Python environment and dependencies.
    - Build Streamlit interface for user input and image display.
    - Integrate Hugging Face API for image generation.
    - Implement image download functionality.
3. **Challenges & Fixes:**
    - **Challenge:** Delayed API rate limits.
      Fix: Optimize input and reduce frequent calls.
    - **Challenge:** Image Generation Time.
      Fix: Use GPU-based environments for speed.

      .

Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Input Validation	Validate API Token	Successful validation or error message	✅ Passed	Tester 1

TC-002	Image Generation	Generate outfit image from text description	Display accurate, realistic image .	✓ Passed	Tester 2

TC-003	Image Display	Display generated image	Image is shown clearly in Streamlit UI	☑ Passed	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✓ Fixed	Developer
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	☐ Deployed	DevOps

---