BizForge: A GenAI-Powered Branding Suite using IBM

Project Description

BizForge is an intelligent, end-to-end branding automation platform that empowers startups, creators, and small businesses to build complete brand identities using Generative AI. The system integrates multiple cutting-edge models—**IBM Granite 4.0-H-350M** for conversational branding guidance, **Groq's ultra-fast LLaMA-3.3-70B** for text generation tasks, and **Stable Diffusion XL (SDXL)** for logo and image creation. BizForge automates the entire branding lifecycle:

- Brand name generation
- Logo & visual identity creation
- Product descriptions
- Social media content
- Email writing
- Tagline & slogan creation
- Sentiment analysis
- Long-text summarization
- AI-powered branding assistant

Built with a **FastAPI backend** and a **lightweight HTML/CSS/JavaScript frontend**, the platform ensures high performance, modularity, and production scalability. BizForge provides a single interface for all branding tasks—transforming hours of manual work into minutes.

Scenarios

Scenario 1: Startup Founder Pitching Investors A founder preparing for a pitch needs a **company name, tagline, and logo** within 24 hours.
BizForge generates:

- 10 brand name suggestions
- 3 taglines
- A full-color vector-style logo using SDXL
- A one-page brand story

Scenario 2: Small Business Owner Creating Product Listings A bakery owner needs consistent product descriptions for an online catalog. BizForge generates:

- Short and long product descriptions
- Tone-customized marketing blurbs
- Customer-focused bullet points

Scenario 3: Marketing Analyst Evaluating New Tagline

A marketing team wants to test the impact of a proposed tagline. BizForge performs:

- Sentiment analysis
- Tone and emotional impact scoring
- Customer perception summary

Architecture Overview

BizForge is a modular multi-AI system combining:

Core Components

- **FastAPI Backend**
  For API routing, model execution, and inter-service communication
- **Frontend (HTML + CSS + JavaScript)**
  Lightweight client interface with fetch-based API communication
- **AI Models Integrated**
  - **IBM Granite 4.0-H-350M** → AI Branding Chatbot
  - **Groq LLaMA-3.3-70B-Versatile** → Text generation (names, descriptions, posts)
  - **Stable Diffusion XL** → Logo & image creation
- **CORS Middleware**
  Ensures smooth cross-origin operations
- **RESTful API Design**
  Modular endpoints for each branding feature

Pre-requisites

Accounts & API Keys

- HuggingFace Account → https://huggingface.co
- Groq Cloud Account → https://console.groq.com

Tools & Frameworks

- **FastAPI** → https://fastapi.tiangolo.com
- **Uvicorn Server** → https://uvicorn.org
- **Python 3.10+** → https://python.org
- **Visual Studio Code** → https://code.visualstudio.com

Project Workflow

Phase 1: Environment Setup & Model Installation

- Activity 1.1: Set Up Python Environment and Install Dependencies
- Activity 1.2: Configure Access for IBM Granite & Groq Cloud Models
- Activity 1.3: Test model connectivity and response quality

Phase 2: Core Backend Development

- Activity 2.1: Set Up FastAPI Application Structure
- Activity 2.2: Implement Basic User Input Modules (No Login System)
- Activity 2.3: Create AI Utility Functions for Groq & IBM Granite Integration
- Activity 2.4: Develop Core Branding Features

Phase 3: Frontend Development

- Activity 3.1: Design responsive HTML templates

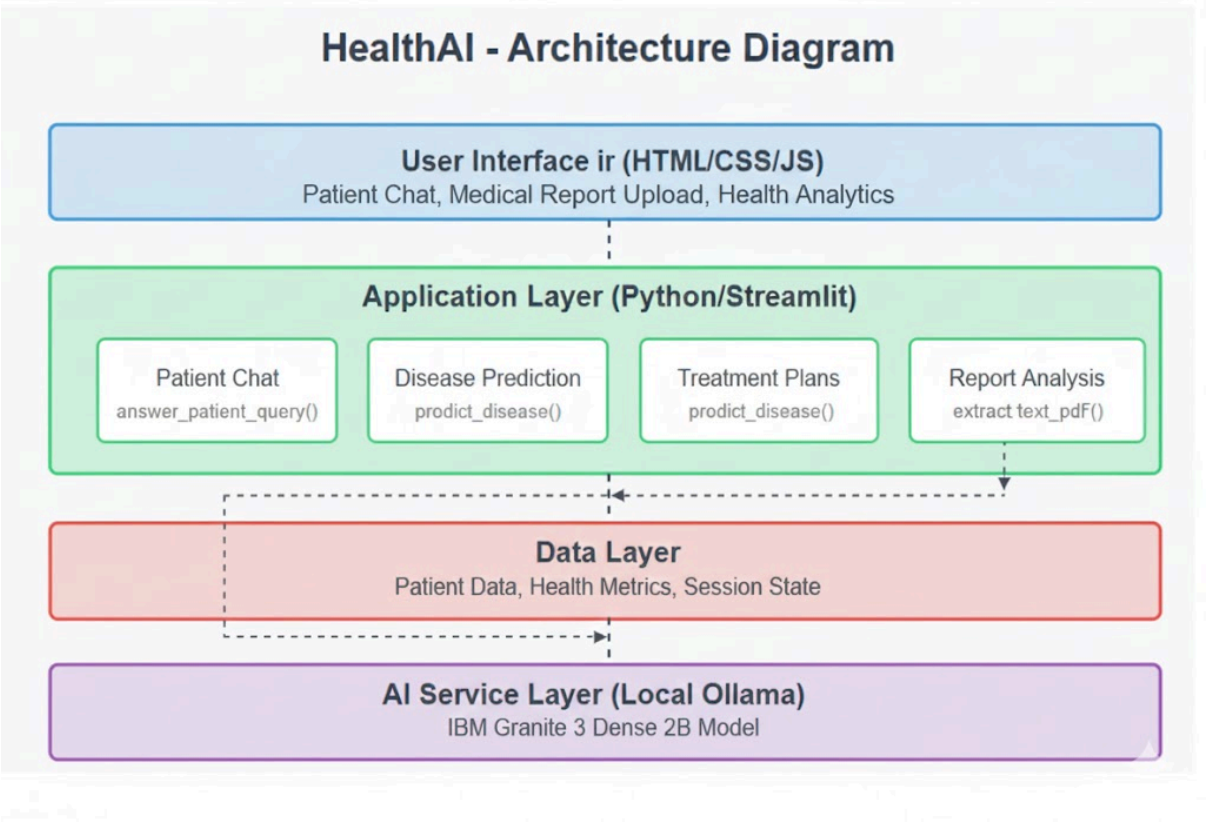
Phase 4: Deployment

- Activity 4.1: Deployment

Phase 5 : Testing & Optimization

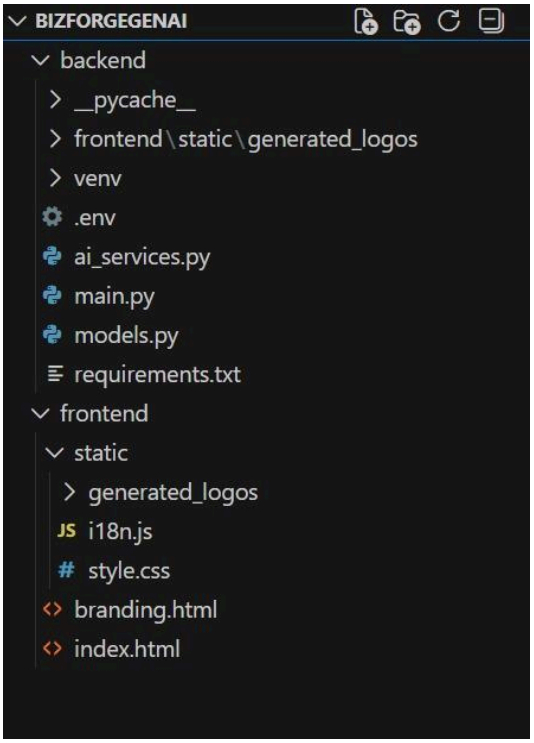
- Activity 5.1: Functional Testing

Technical architecture



Project Structure Setup

Create the following directory structure:



MILESTONE 1: Environment Setup & AI Model Configuration

This milestone establishes the core AI foundation for BizForge by installing required dependencies, configuring access to the IBM Granite model hosted on Hugging Face, and setting up the ultra-fast Groq Cloud environment for text generation tasks. Completing this milestone ensures that the BizForge backend can generate creative brand names, marketing content, and chatbot responses using the appropriate AI pipelines.

Activity 1.1: Install Python Environment & Required Dependencies

Before integrating branding models, the development environment must be properly configured.

For Windows / Mac / Linux:

1. **Install Python 3.10+**
   Download from: https://www.python.org/downloads/
   Create and activate virtual environment
   Windows:
   python -m venv venv

   venv\Scripts\activate

   Mac/Linux:
   python3 -m venv venv

   source venv/bin/activate

2. **Install core project dependencies**
   pip install fastapi uvicorn python-dotenv requests transformers groq
3. **Verification:**

   Run:

   python --version

   pip list

If the environment is properly configured, the list will include fastapi, transformers, and groq.

Activity 1.2: Configure IBM Granite Model via Hugging Face

BizForge uses **IBM Granite-4.0-H-350M** from Hugging Face for the AI Branding Assistant.

This model provides strategic branding suggestions, consultative responses, and brand identity guidance.

Steps:

1. Create (or log in to) a Hugging Face account
   https://huggingface.co/
2. Generate a **Hugging Face Access Token**
   - Go to *Settings → Access Tokens*
   - Click **New Token → Read**
   - Copy the key
3. Add the key to the BizForge .env file:
   HF_API_KEY="your_huggingface_api_key"

   IBM_MODEL="ibm-granite/granite-4.0-h-350m"

4. Verify model availability:
   Visit model page: https://huggingface.co/ibm-granite

Activity 1.3: Set Up Groq Cloud & Test LLaMA-3.3-70B-Versatile

BizForge uses Groq Cloud to run the **LLaMA-3.3-70B-Versatile** model for high-speed text generation tasks such as:

- brand name creation
- product descriptions
- email writing
- social media posts

Groq is chosen for its extremely low latency (sub-50ms response time) and high throughput.

Steps:

1. Create a Groq Cloud Account
   https://console.groq.com/

2. Generate an API Key

- Dashboard → API Keys → **Create Key**
- Add key to .env:
  GROQ_API_KEY="your_groq_cloud_api_key"

  GROQ_MODEL="llama-3.3-70b-versatile"

3. Install Groq Python SDK (if not installed):
   pip install groq
4. Test Connectivity with a Sample Prompt:

```
from groq import Groq

client = Groq(api_key="your_key")

response = client.chat.completions.create(

model="llama-3.3-70b-versatile",

messages=[{"role": "user", "content": "Generate 3 creative brand names for an AI startup."}]

)

print(response.choices[0].message["content"])
```

MILESTONE 2: Core Backend Development

Objective

Build the FastAPI-based backend infrastructure that powers all branding features in BizForge. This milestone focuses on creating a modular API structure, implementing AI service utilities for Groq and IBM Granite, integrating the Stable Diffusion XL model for logo generation, and developing the core branding functionality—such as name generation, content creation, sentiment analysis, and chatbot interactions.

Activity 2.1: FastAPI Application Initialization

Create requirements.txt:

```
fastapi==0.104.1
uvicorn==0.24.0
python-dotenv==1.0.0
torch>=2.0.0
transformers>=4.35.0
groq>=0.4.0
requests>=2.31.0
huggingface-hub>=0.19.0
Pillow>=10.0.0
```

Install Dependencies:

Open terminal/command prompt and run:

>> python -m venv venv

>> venv\Scripts\activate

>> pip install -r requirements.txt

```
PS C:\Users\ROHITH KARTHIKEYA\Desktop\BizForgeGenAI> cd backend
PS C:\Users\ROHITH KARTHIKEYA\Desktop\BizForgeGenAI\backend> venv/Scripts/activate
(venv) PS C:\Users\ROHITH KARTHIKEYA\Desktop\BizForgeGenAI\backend> pip install -r requirements.txt
Requirement already satisfied: fastapi==0.104.1 in c:\users\rohith karthikeya\desktop\bizforgegenai\backen
d\venv\lib\site-packages (from -r requirements.txt (line 1)) (0.104.1)
Requirement already satisfied: uvicorn==0.24.0 in c:\users\rohith karthikeya\desktop\bizforgegenai\backend
\venv\lib\site-packages (from -r requirements.txt (line 2)) (0.24.0)
Requirement already satisfied: python-dotenv==1.0.0 in c:\users\rohith karthikeya\desktop\bizforgegenai\ba
```

Activity 2.2: Initialize the FastAPI App in main.py

Import Required Libraries:

```python
from fastapi import FastAPI, HTTPException
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse
from fastapi.middleware.cors import CORSMiddleware
import os
from pathlib import Path
from fastapi.staticfiles import StaticFiles
from fastapi import FastAPI, UploadFile, File, HTTPException
from fastapi.responses import JSONResponse
app = FastAPI()
```

The `main.py` file performs:

- FastAPI app creation
- Router registration
- CORS configuration
- Environment variable loading

Install Required Dependencies

All backend requirements were installed:

pip install fastapi uvicorn python-dotenv transformers groq httpx pillow

Activity 2.3: Implement Basic Input Processing & Validation

This activity focuses on creating the foundational input-handling system for BizForge. Since the platform does **not use login or user account profiles**, all branding tools rely directly on user-provided information entered through the frontend. Ensuring the accuracy, structure, and clarity of these inputs is essential for generating high-quality branding outputs from Groq LLaMA and IBM Granite models.

The goal of this activity is to design a **robust input-processing pipeline** that validates and sanitizes user prompts, prevents errors, and prepares clean, optimized inputs for each AI feature.

backend > main.py > ...

```python
async def analyze_sentiment_endpoint(request: dict):
    try:
        data = request
        result = await analyze_sentiment(
            data.get("text"),
            data.get("brand_tone", "Professional")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

@app.post("/api/get-colors")
async def get_colors_endpoint(request: dict):
    try:
        data = request
        result = await get_color_palette(
            data.get("tone"),
            data.get("industry")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

@app.post("/api/chat")
async def chat_endpoint(request: dict):
    try:
        data = request
        result = await chat_with_ai(data.get("message"))
        return {"success": True, "data": {"content": result}}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

backend > main.py > ...

```python
@app.get("/{page}.html")
async def serve_page(page: str):
    file_path = frontend_path / f"{page}.html"
    if file_path.exists():
        return FileResponse(file_path)
    return FileResponse(frontend_path / "index.html")

@app.get("/{path}")
async def catch_all(path: str):
    file_path = frontend_path / path
    if file_path.exists():
        return FileResponse(file_path)
    return FileResponse(frontend_path / "index.html")

@app.on_event("startup")
async def startup():
    print("\n" + "="*60)
    print("🚀 BizForge Backend Started!")
    print("="*60)
    print(f"🌐 API running at http://localhost:8000")
    print(f"📁 Frontend path: {frontend_path}")
    print(f"📦 Static files path: {frontend_path / 'static'}")
    print("="*60 + "\n")

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Activity 2.4: Create AI Utility Functions for IBM Granite, Groq LLaMA & SDXL

This is the core engine of BizForge's intelligence.

All AI integrations were implemented inside ai_service.py.

Modules Implemented:

1. **IBM Granite Integration (Branding Chatbot)**

```
# ==================== IBM GRANITE (Local) for CHAT ====================
device = "cpu"
print("🔹 Loading IBM Granite 4.0-h-350m...")

granite_model = None
granite_tokenizer = None

try:
    model_id = "ibm-granite/granite-4.0-h-350m"
    granite_tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)
    granite_model = AutoModelForCausalLM.from_pretrained(
        model_id,
        torch_dtype=torch.float32,
        trust_remote_code=True,
    ).to(device)
    granite_model.eval()
    print("✅ IBM Granite loaded!")
except Exception as e:
    print(f"❌ Granite load failed: {e}")
```

2. **Groq LLaMA-3.3-70B Integration (Creative Text Generator)**

```
# ==================== GROQ CLIENT (Llama for BRANDING) ====================
if GROQ_API_KEY:
    groq_client = Groq(api_key=GROQ_API_KEY)
else:
    groq_client = None
    print("⚠️ Warning: GROQ_API_KEY not set in .env")

async def generate_with_groq(prompt: str, max_tokens: int = 150) -> str:
    """Generate using Groq Llama 3.3"""
    if not groq_client:
        return "❌ Error: GROQ_API_KEY not set in .env"

    try:
        message = groq_client.chat.completions.create(
            model="llama-3.3-70b-versatile",
            messages=[{"role": "user", "content": prompt}],
            max_tokens=max_tokens,
            temperature=0.7,
            top_p=0.95,
        )

        result = message.choices[0].message.content.strip()
        return result
```

3. **Stable Diffusion XL (Logo/Image Generator)**

```python
async def generate_logo_image(logo_prompt: str):
    try:
        print(f"🎨 Generating image with Stable Diffusion XL...")
        # Initialize HF Inference Client
        client = InferenceClient(api_key=HF_API_KEY)
        # Enhanced prompt for better logo generation
        enhanced_prompt = f"Professional brand logo for: {logo_prompt}. Modern minimalist vector de
        print(f"📝 Prompt: {enhanced_prompt}")
        # Generate image using Stable Diffusion XL
        image = client.text_to_image(
            enhanced_prompt,
            model="stabilityai/stable-diffusion-xl-base-1.0",
        )
        # Create directory
        os.makedirs("frontend/static/generated_logos", exist_ok=True)
        timestamp = int(time.time())
        # FIXED: Use absolute path
        image_filename = f"logo_{timestamp}.png"
        image_path = os.path.join("frontend/static/generated_logos", image_filename)
        # Save PIL image
        image.save(image_path)
        print(f"✅ Image saved to: {image_path}")
        # Return URL path for frontend
        return {
            "image_url": f"/static/generated_logos/{image_filename}",
            "success": True,
            "error": None
        }
```

Activity 2.5: Brand Name Generator Feature

Brand Naming (/api/generate-brand):

- **Input:** Receives the industry/business type, keywords, tone preference, and language settings from the user.
- **Prompt Construction:** Validates and formats the inputs, then builds a structured naming prompt to guide Groq LLaMA in generating unique, brand-ready name ideas.
- **AI Call:** Calls the `generate_brand_names()` function using Groq LLaMA-3.3-70B with the system instruction ("You are BizForge. Generate unique, memorable, and brand-ready names…").
- **Output:** Returns 10–20 AI-generated brand names with optional one-line explanations to the frontend.

```python
# Routes
@app.post("/api/generate-brand")
async def generate_brand_endpoint(request: dict):
    try:
        data = request
        result = await generate_brand_names(
            data.get("industry"),
            data.get("keywords"),
            data.get("tone", "Professional"),
            data.get("language", "en")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Brand Name Generator Features:

- Keyword-aware suggestions
- Tone-aligned naming (modern, luxury, minimal, playful, etc.)
- Industry-specific creativity
- Fast and concise outputs using Groq's low-latency model

Activity 2.6: Marketing Content Generation Feature

Content Creation (/api/generate-content):

- **Input:** Receives the brand description, selected tone (professional, playful, luxury, etc.), content type (product description, caption, ad copy), and language.
- **Prompt Construction:** The backend validates the inputs, enriches them with branding context, and constructs a detailed instruction prompt tailored for marketing content generation.
- **AI Call:** Calls the `generate_marketing_content()` function, using Groq LLaMA to produce polished, consistent, and on-brand marketing text.
- **Output:** Returns high-quality AI-generated marketing content formatted according to the selected content type.

```python
@app.post("/api/generate-content")
async def generate_content_endpoint(request: dict):
    try:
        data = request
        result = await generate_marketing_content(
            data.get("brand_description"),
            data.get("tone", "Professional"),
            data.get("content_type", "product_description"),
            data.get("language", "en")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Activity 2.7: Sentiment Analysis Feature

Brand Review Analysis (/api/analyze-sentiment):

- **Input:** Receives the text to analyze along with the brand tone reference.
- **Contextualization:** The backend evaluates the text against sentiment patterns and aligns it with the specified branding tone for accurate emotional interpretation.
- **AI Call:** Calls the `analyze_sentiment()` function, which uses an NLP sentiment model to classify polarity and tone.
- **Output:** Returns the sentiment classification (positive/neutral/negative), confidence score, and brand-tone alignment insights.

```python
@app.post("/api/analyze-sentiment")
async def analyze_sentiment_endpoint(request: dict):
    try:
        data = request
        result = await analyze_sentiment(
            data.get("text"),
            data.get("brand_tone", "Professional")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Activity 2.8: Color Palette Recommendation Feature

Brand Colors (/api/get-colors):

- **Input:** Receives the brand tone (minimal, bold, elegant, etc.) and industry category.
- **Contextualization:** The backend constructs a color-focused prompt that aligns visual suggestions with industry standards and the emotional tone of the brand.
- **AI Call:** Calls get_color_palette() to generate curated HEX/RGB color combinations using AI-driven visual branding rules.
- **Output:** Returns a palette of 3–5 optimized brand colors for consistent and professional branding.

```python
@app.post("/api/get-colors")
async def get_colors_endpoint(request: dict):
    try:
        data = request
        result = await get_color_palette(
            data.get("tone"),
            data.get("industry")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Activity 2.9: AI Chatbot Feature

Branding Consultation (/api/chat):

- **Input:** Receives the user_message containing branding questions such as naming suggestions, logo ideas, tagline feedback, or brand strategy queries.
- **Contextualization:** The backend constructs a branding-focused instruction prompt guiding IBM Granite to respond as an expert brand consultant.
- **AI Call:** Calls the `chat_with_ai()` function, using the system instruction ("You are BizForge, an expert branding assistant…") to generate strategic and actionable branding insights.
- **Output:** Returns the AI-generated branding response to the frontend.

```python
@app.post("/api/chat")
async def chat_endpoint(request: dict):
    try:
        data = request
        result = await chat_with_ai(data.get("message"))
        return {"success": True, "data": {"content": result}}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Activity 2.10: Logo Prompt Generation Feature

Logo Guidance (/api/generate-logo):

- **Input:** Receives the brand name, industry, and core keywords from the user.
- **Prompt Construction:** The backend builds a detailed, professional logo prompt describing style, colors, shapes, emotion, and visual branding elements suitable for the business.
- **AI Call:** Calls the `generate_logo_prompt()` function, using Groq LLaMA to refine and produce a clean, high-quality logo-generation prompt.
- **Output:** Returns a polished and ready-to-use logo prompt for SDXL/DALL·E image generation to the frontend.

```python
@app.post("/api/generate-logo")
async def generate_logo_endpoint(request: dict):
    try:
        data = request
        result = await generate_logo_prompt(
            data.get("brand_name"),
            data.get("industry"),
            data.get("keywords")
        )
        return {"success": True, "data": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Activity 2.11: Voice Input Transcription Feature

Speech-to-Text (/api/transcribe-voice):

- **Input:** Receives the uploaded audio file from the user.
- **Processing:** The backend reads, converts, and processes the audio using the SpeechRecognition library to prepare it for transcription.
- **AI Call:** Calls Google's free Speech-to-Text service to convert the spoken audio into clean, usable text.
- **Output:** Returns the transcribed text back to the frontend for branding queries or content generation.

```python
@app.post("/api/transcribe-voice")
async def transcribe_voice(audio_file: UploadFile = File(...)):
    """
    Transcribe audio file using Google Speech-to-Text API
    This is a FREE fallback for browsers that block Web Speech API (like Brave)
    """
    try:
        # Read audio file
        audio_content = await audio_file.read()
        # Use Google's free Speech Recognition API (requires internet)
        # This is a workaround using pydub and SpeechRecognition library
        import speech_recognition as sr
        from io import BytesIO
        # Save to temporary file
        with open("temp_audio.wav", "wb") as f:
            f.write(audio_content)
        # Recognize speech
        recognizer = sr.Recognizer()
        with sr.AudioFile("temp_audio.wav") as source:
            audio = recognizer.record(source)
        # Use Google Speech Recognition (free)
        text = recognizer.recognize_google(audio)

        return {"success": True, "text": text}
    except Exception as e:
        raise HTTPException(status_code=400, detail=f"Transcription failed: {str(e)}")
```

Activity 2.12: System Setup & Initialization

This entry point defines how the BizForge application is launched and configured.

- Application Run (if __name__ == '__main__'):
    - **Initialization:** Starts the FastAPI application using the Uvicorn server.
    - **Configuration:** Runs the application on host `0.0.0.0` and port `8000` for local and external access.
    - **Deployment:** Makes the BizForge backend accessible at `http://localhost:8000`, enabling all API features such as brand generation, marketing content creation, sentiment analysis, and chatbot interaction.

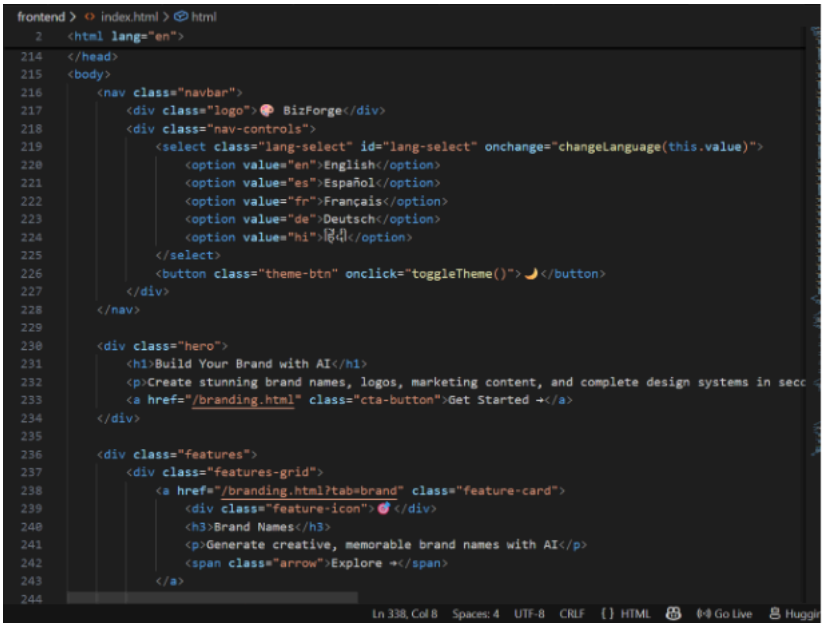
```python
if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

MILESTONE 3: Frontend Development - UI/UX Design

Create a modern, responsive, and visually appealing user interface using HTML5, CSS3, and JavaScript. The frontend should provide smooth animations, intuitive navigation, and seamless integration with backend APIs.

Activity 3.1: Base Template Structure

index.html:



Base Template Features:

- **Gradient Background:** Full-page modern gradient (purple–blue) for a premium branding look.
- **Responsive Navbar:** Sticky top navigation with language selector and theme toggle button.
- **Gradient Logo Text:** "BizForge" displayed with a bold gradient-filled text logo.
- **Hero Section:** Large headline, subtext, and a prominent CTA button directing users to the branding tools page.
- **Interactive Feature Cards:** Hover-animated cards for Brand Names, Logo Generator, Marketing Content, Design System, Sentiment Analysis, and Branding Chat.
- **Modern Grid Layout:** Auto-fit responsive grid system for showcasing features across all devices.
- **Info Section:** Clean white background section explaining "Why Choose BizForge?" with supporting feature tiles.
- **Feature Highlights:** Dedicated tiles for AI-powered tools, instant results, multilingual support, voice input, and professional output.
- **Footer:** Minimal, centered footer displaying © 2025 BizForge with model credits (IBM Granite, Groq AI, SDXL).
- **Theme Toggle:** Light/dark inversion toggle implemented with a single button for user personalization.
- **Language Selector:** Dropdown menu supporting English, Spanish, French, German, and Hindi.
- **Smooth UI Interactions:** Hover animations, shadows, transitions, and floating effects for enhanced user experience.
- **Mobile Optimization:** All layout components scale fluidly for small and large screens.

Activity 3.2: Creating Branding.html file

```
<div class="container">
    <!-- BRAND TAB -->
    <div id="brand" class="tab-content active">
        <div class="page-header">
            <h1>🎯 Brand Names Generator</h1>
            <p>Enter keywords and generate creative brand names</p>
        </div>
```

```
    <!-- LOGO TAB -->
    <div id="logo" class="tab-content">
        <div class="page-header">
            <h1>🎨 Logo Generator</h1>
            <p>Generate logo concepts and images</p>
        </div>
```

```
    <!-- CONTENT TAB -->
    <div id="content" class="tab-content">
        <div class="page-header">
            <h1>📝 Marketing Content</h1>
            <p>Generate descriptions, posts, and ads</p>
        </div>
```

```
    <!-- DESIGN TAB -->
    <div id="design" class="tab-content">
        <div class="page-header">
            <h1>🎨 Design System</h1>
            <p>Get color palettes and fonts</p>
        </div>
```

```
<!-- SENTIMENT TAB - SIMPLIFIED -->
<div id="sentiment" class="tab-content">
    <div class="page-header">
        <h1>💭 Review Analyzer</h1>
        <p>Analyze and rewrite reviews in professional tone</p>
    </div>
```

```
<!-- CHAT TAB -->
<div id="chat" class="tab-content">
    <div class="page-header">
        <h1>💬 Branding Chat</h1>
        <p>Ask questions about branding</p>
    </div>
```

- **brand tab:** This page serves as the Brand Name Generator interface, allowing users to enter keywords, choose an industry, select a tone, and generate creative brand name ideas using AI.
- **logo tab:** This page provides the Logo Generator interface where users can input their brand name, industry, and keywords to generate logo concepts and AI-assisted logo descriptions.
- **content tab:** This page is dedicated to marketing content generation, enabling users to enter a brand description, select tone and content type, and generate descriptions, captions, and ad copy.
- **design tab:** This page offers the Design System interface where users select tone and industry to receive AI-generated color palettes, style recommendations, and visual branding guidance.
- **sentiment tab:** This page provides a simple interface for analyzing customer reviews, allowing users to paste text and receive sentiment insights along with AI-improved rewrites.
- **chat tab:** This page serves as the AI Branding Chat interface, enabling users to ask branding questions and receive expert guidance from the BizForge chatbot.

MILESTONE 4: Deployment

Activity 4.1: Deployment Preparation

Local Deployment :
run the command in the terminal :

>> uvicorn main:app — reload

```
(venv) PS C:\Users\ROHITH KARTHIKEYA\Desktop\BizForgeGenAI\backend> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\ROHITH KARTHIKEYA\\Desktop\\BizForgeGe
nAI\\backend']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [20812] using StatReload
 🔹 Loading IBM Granite 4.0-h-350m...
`torch_dtype` is deprecated! Use `dtype` instead!
The fast path is not available because one of `(selective state update, causal conv1d fn, causal conv1d up
```

MILESTONE 5 : Testing & Optimization

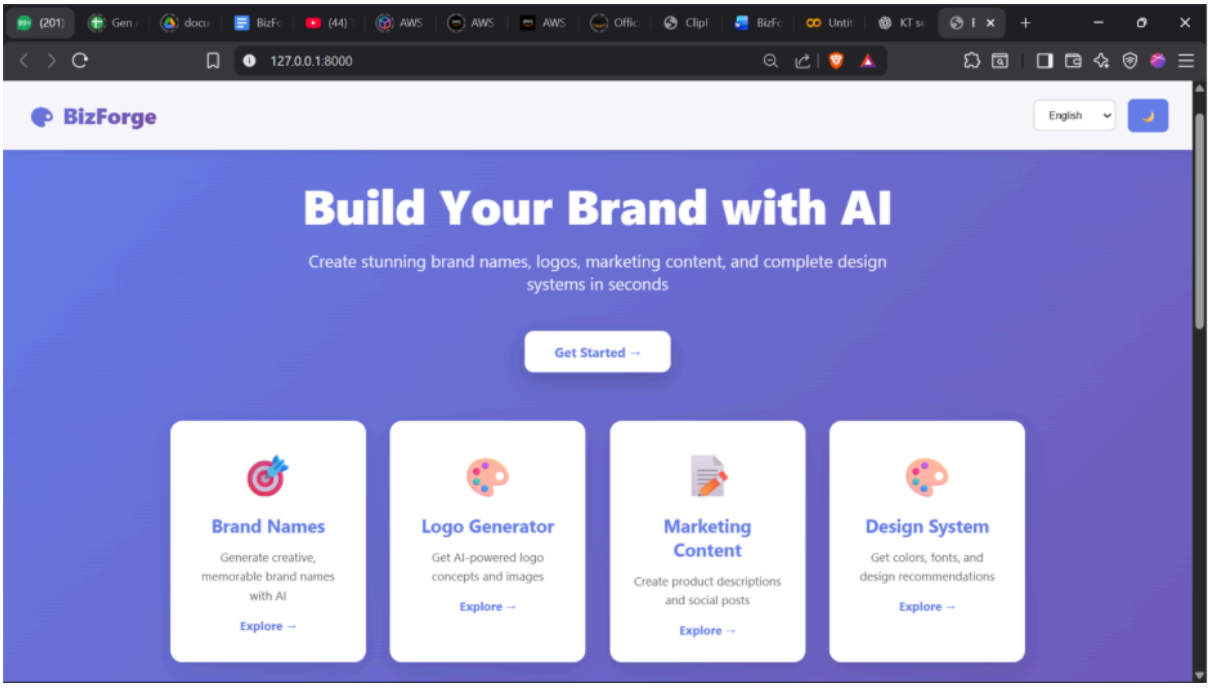
Activity 4.1: Functional Testing

Test Cases:

Test 1: Home Page

1. Navigate to landing page

2. Click "Get Started"

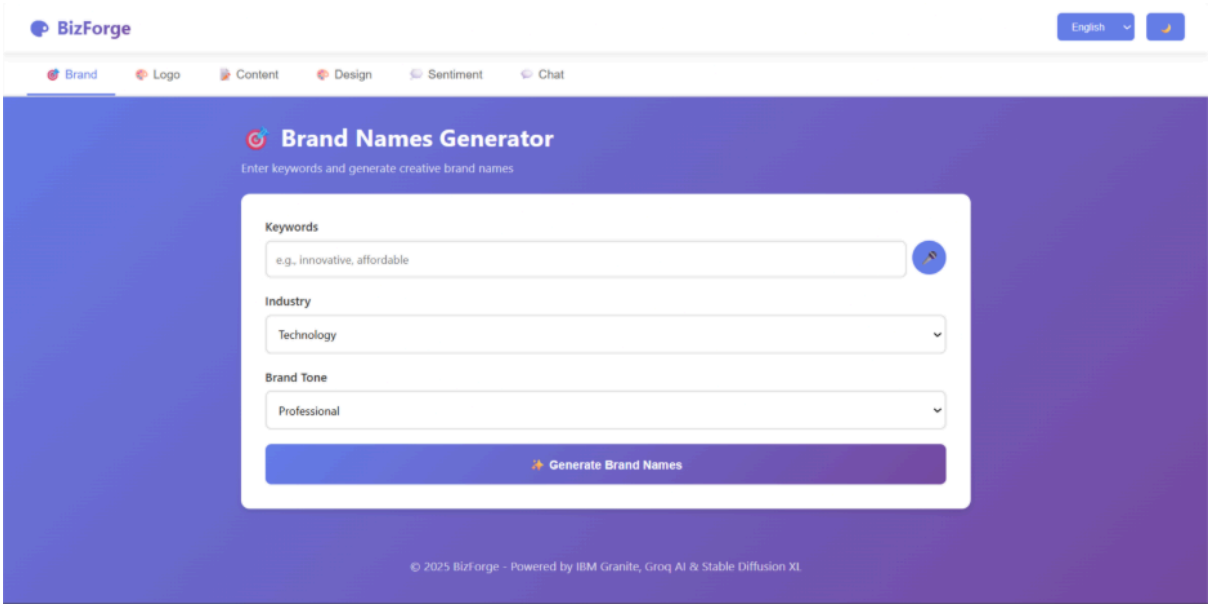


Test 2: Brand name Generator

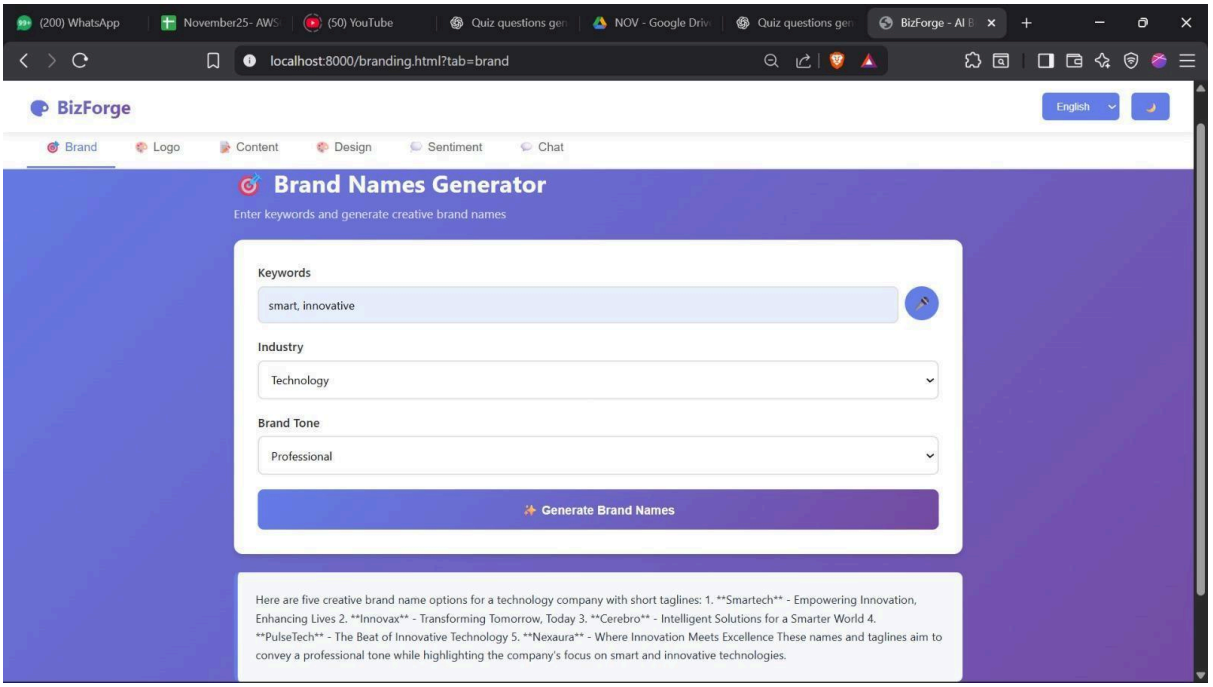
1. Click "Brand" in navigation

2. Ex→ Type: "Smart, innovative"

3. Click Generate Brand names



Output:-

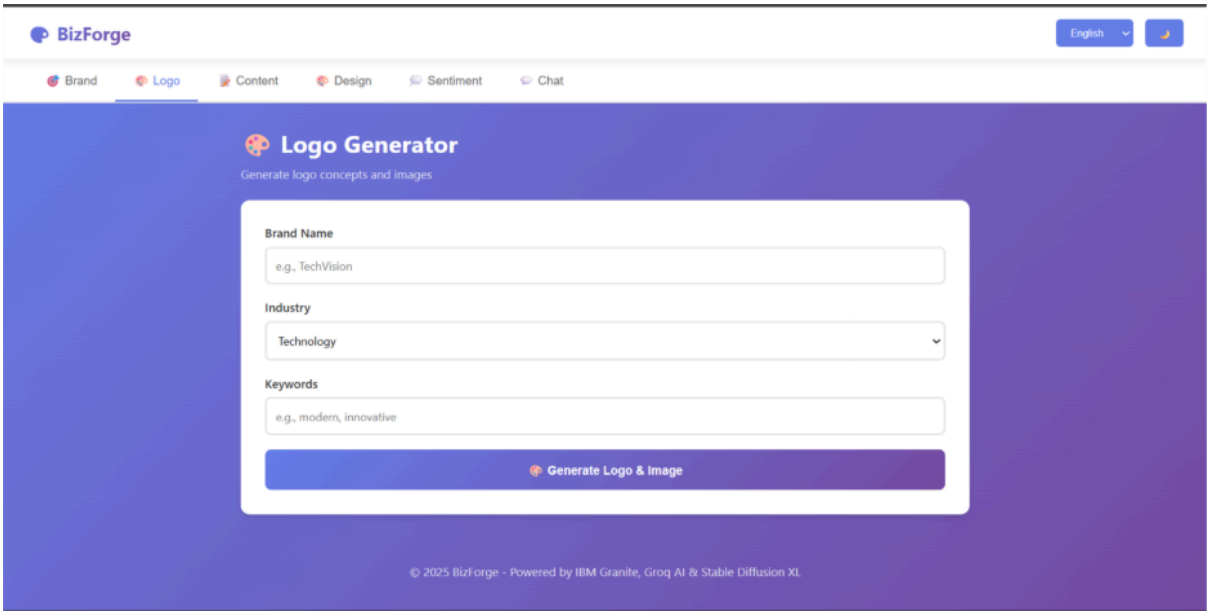


Test 3: Logo generator

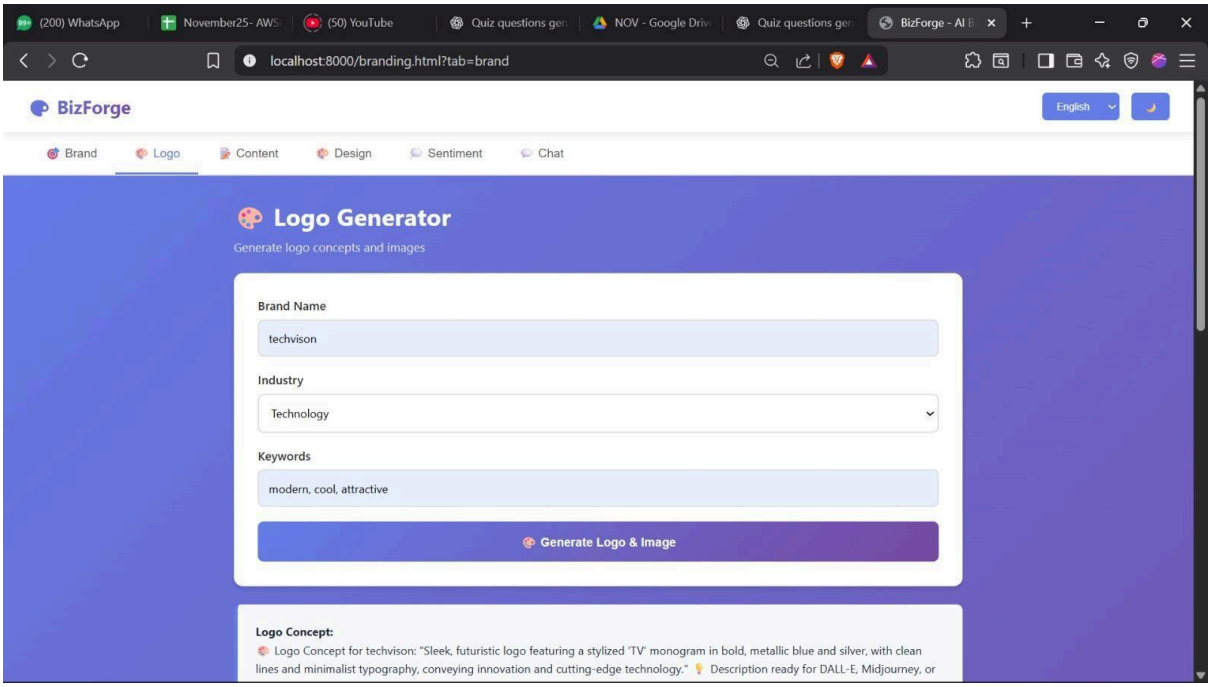
1. Navigate to Logo

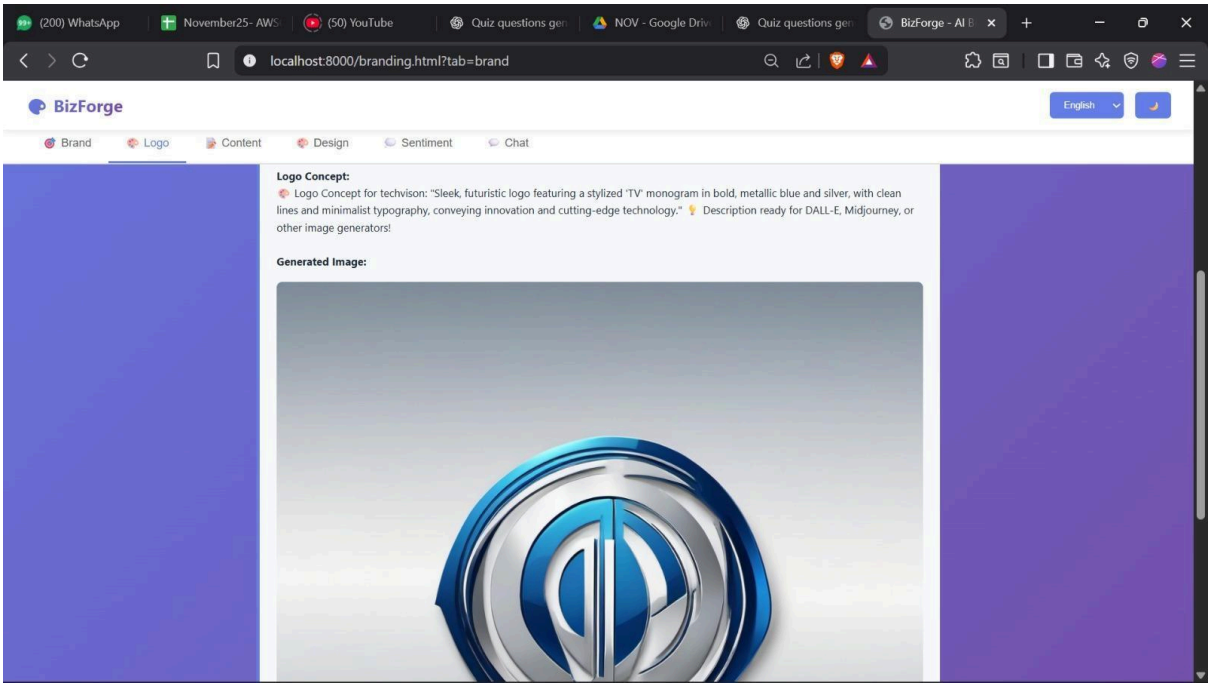
2. Enter: "technision, technology, modern cool attractive"

3. Click "Generate Logo& image"



Output:-



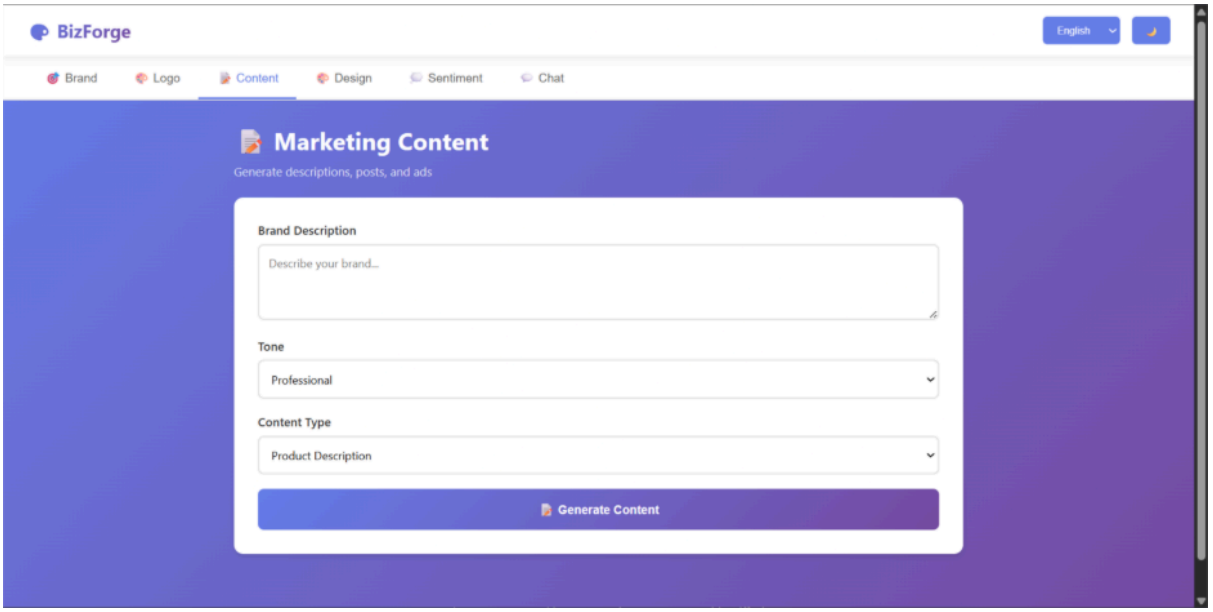


Test 4: Marketing Content

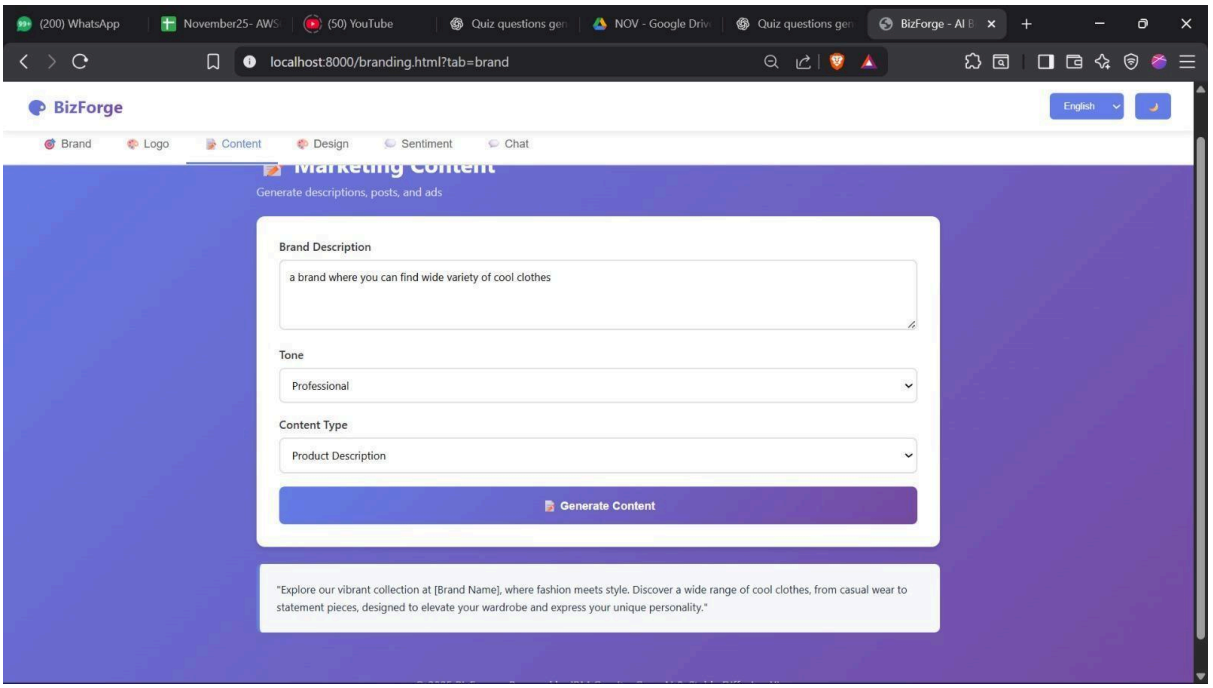
1. Go to Content

2. EX→ Enter: "A brand where you find wide variety of cool clothes"

3. Submit



Output:-

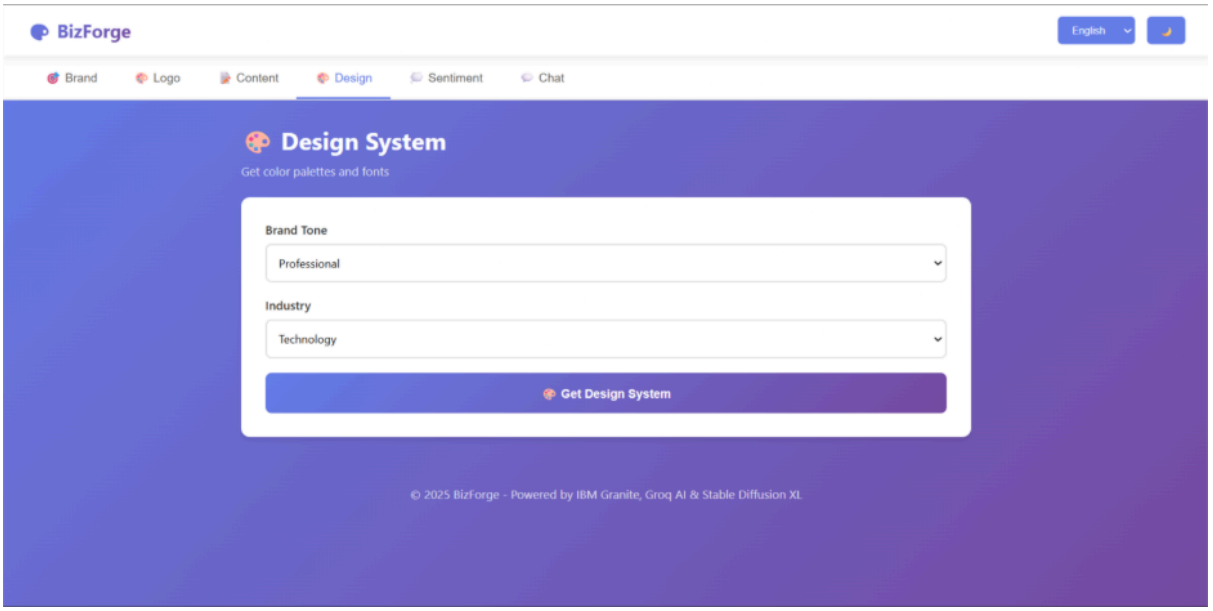


Test 5: Design System

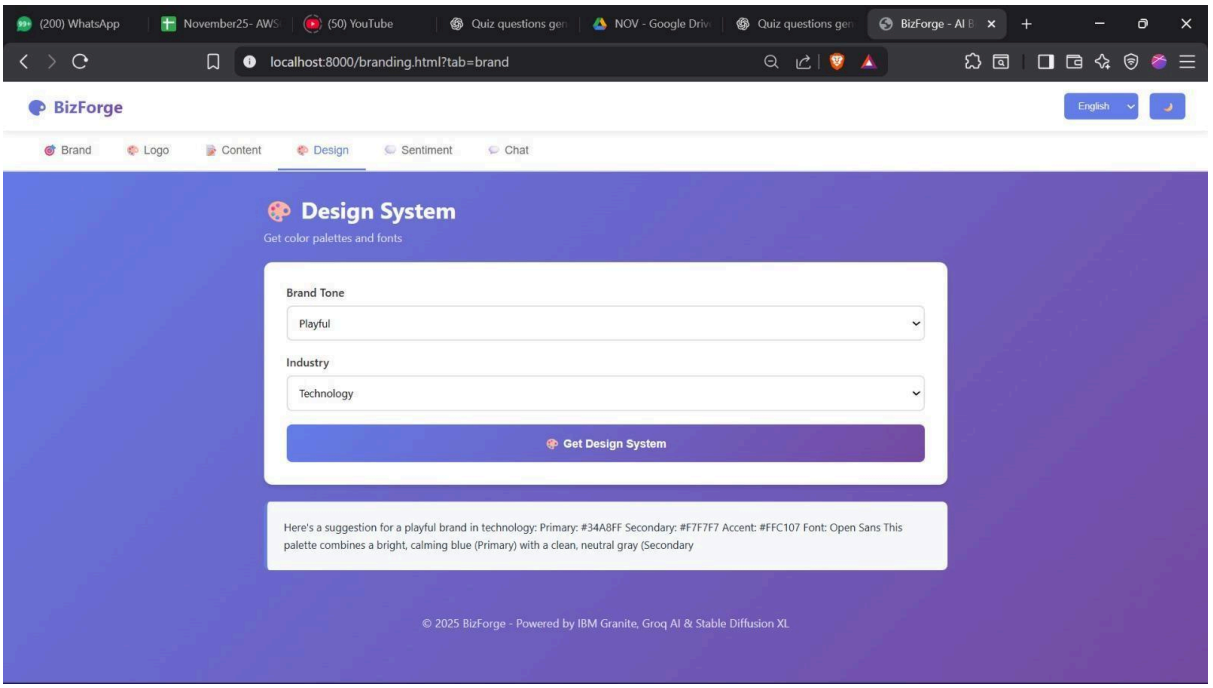
1. Navigate to Design

2. Ex→ Playful Technology

3. Click "Get design system"



Output:-

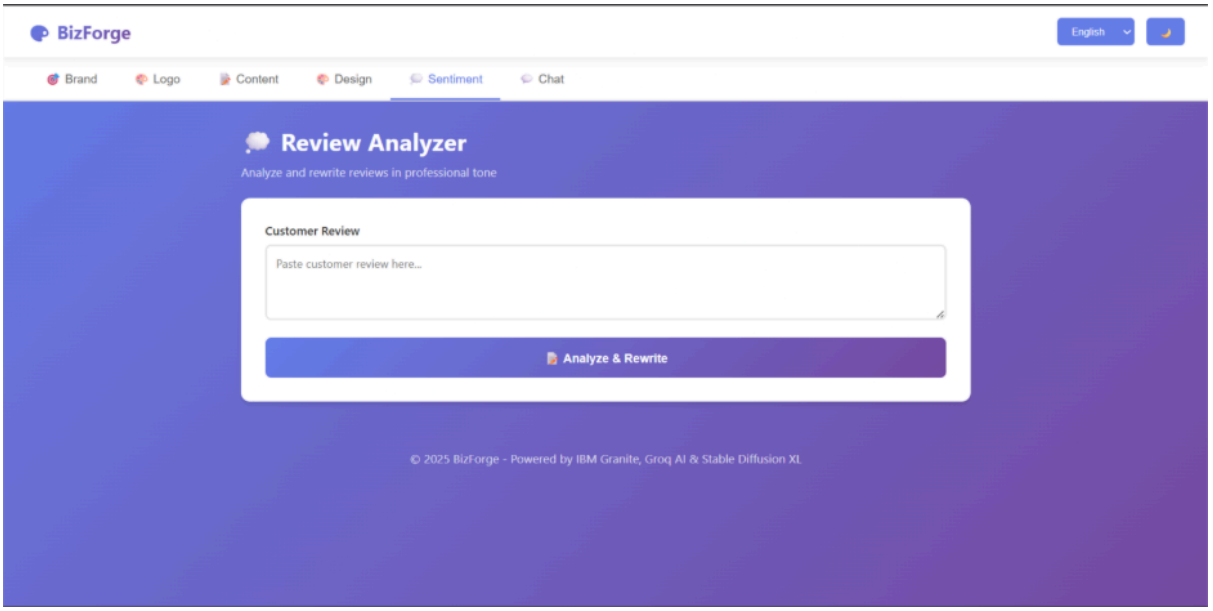


Test 6 : Review Analyzer

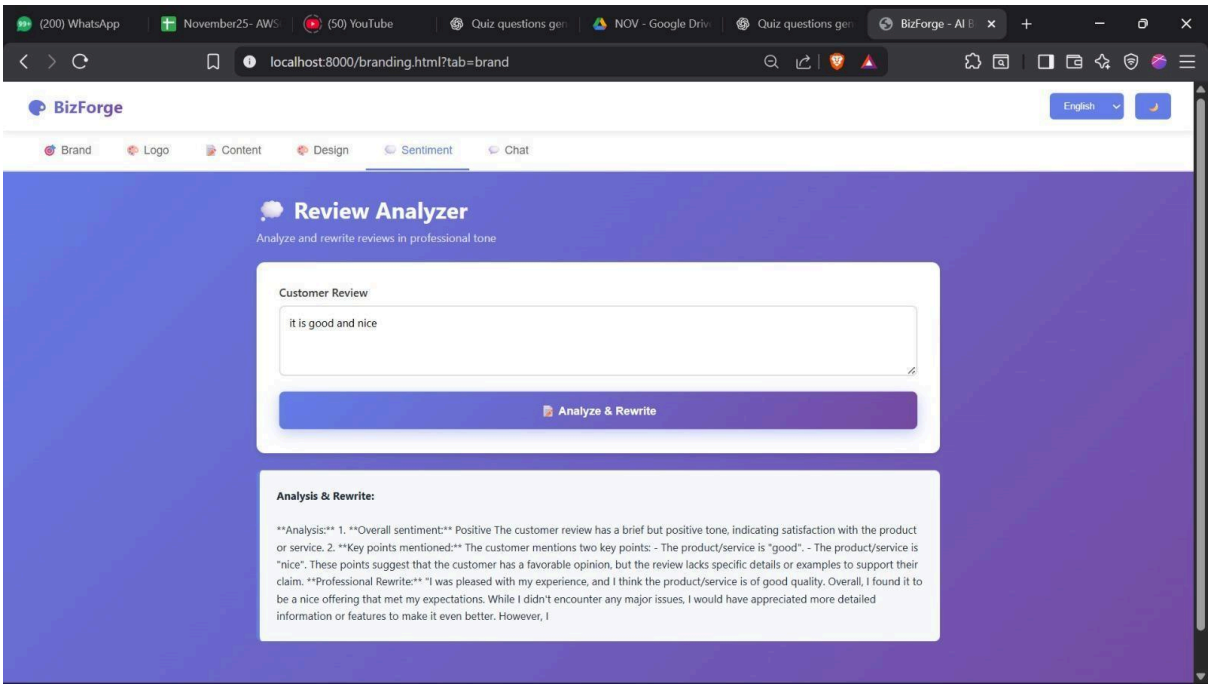
1. Navigate to Sentiment

2. Ex→ it is good and nice

3. Click "Analyze & Rewrite"



Output:-

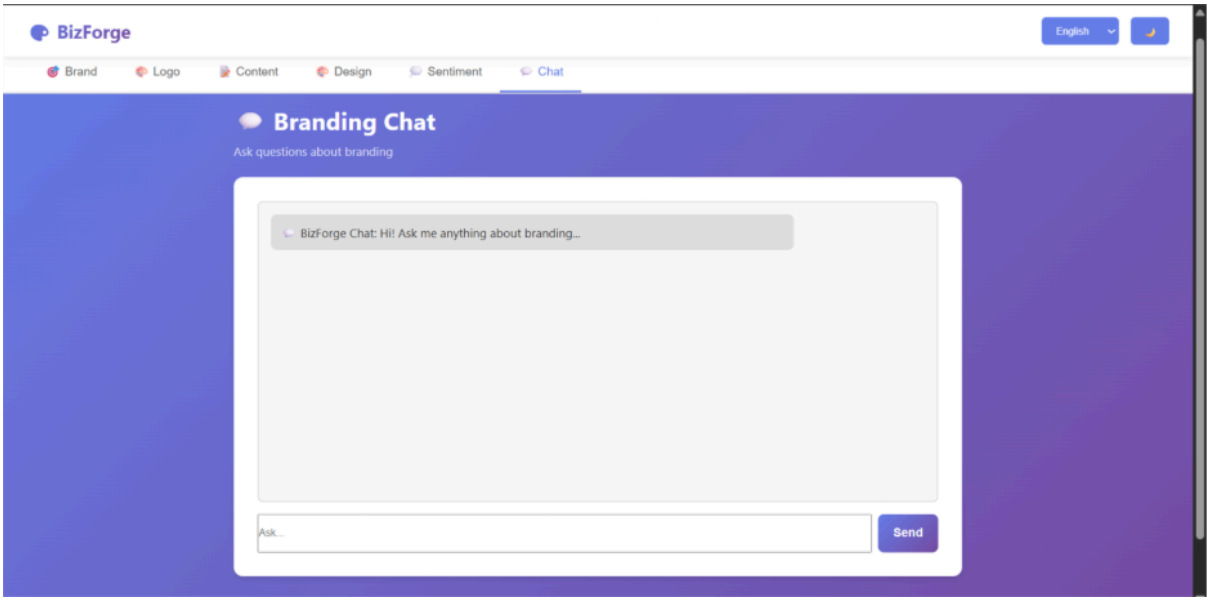


Test 6 : AI chatbot

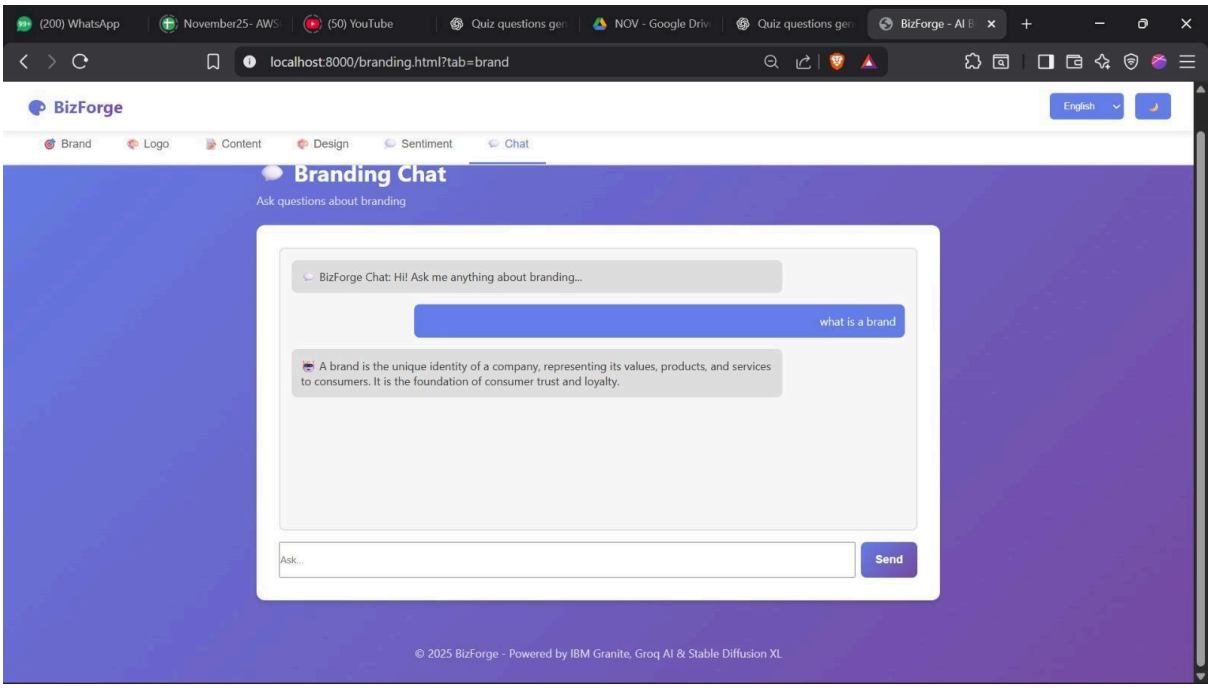
1. Navigate to chat

2. Ex→ what is a brand

3. Click "Send"



Output:-


(200) WhatsApp
November25- AWS
(50) YouTube
Quiz questions gen
NOV - Google Driv
Quiz questions gen
BizForge - AI B
localhost:8000/branding.html?tab=brand
BizForge
English
Brand
Logo
Content
Design
Sentiment
Chat
Branding Chat
Ask questions about branding
BizForge Chat: Hi! Ask me anything about branding...
what is a brand
A brand is the unique identity of a company, representing its values, products, and services to consumers. It is the foundation of consumer trust and loyalty.
Ask...
Send
© 2025 BizForge - Powered by IBM Granite, Groq AI & Stable Diffusion XL

Conclusion

BizForge successfully demonstrates the power and practicality of Generative AI in solving real-world branding challenges. By integrating Hugging Face's NLP models with a user-friendly web interface, we created a comprehensive suite of tools for startups and entrepreneurs to automate brand ideation — from naming to logo creation, social content drafting, sentiment analysis, and more. Despite the challenges faced, particularly around adapting to API deprecations and migrating from OpenAI to Hugging Face, this journey enabled a deeper understanding of AI model capabilities, backend–frontend integration, and scalable software design. The project also underscored the importance of user experience and modular architecture in building robust digital solutions. Looking ahead, BizForge has immense potential to evolve into a commercial-grade branding assistant with premium features like real-time analytics, advanced personalization, and AI-enhanced design tools. This project not only showcases the impact of modern AI but also marks a significant milestone in leveraging open-source intelligence for creative automation.