**DXFactor Chat Agent - RAG-based Solution using Google Gemini**

Contents

[1. Introduction 1](#_Toc244519333)

[2. System Architecture 2](#_Toc244519334)

[3. Technology Used 4](#_Toc244519335)

[4. Installation and Setup 4](#_Toc244519336)

[5. Directory Structure 5](#_Toc244519341)

[6. System Workflow 5](#_Toc244519342)

[7. Error Handling and Optimizations 6](#_Toc244519354)

[8. Security Consideration 6](#_Toc244519355)

[9. Testing Strategy 6](#_Toc244519358)

[11. Example Queries 6](#_Toc244519360)

**Introduction**

The DXFactor Chat Agent is an advanced Retrieval-Augmented Generation (RAG) chatbot designed to provide accurate responses based on information retrieved from DXFactor's website. The system uses a combination of web scraping, vector database storage, and generative AI to fetch and synthesize the most relevant answers.

The chatbot leverages Google’s Gemini API, ChromaDB, and Tavily Search API to ensure that responses are contextual, precise, and up-to-date. The entire system is wrapped in a user-friendly Streamlit web interface for seamless interaction.

**System Architecture**

1. **Web Scraping**
   * Uses **selenium** to extract content from DXFactor's website.
   * Stores extracted content in dxfactor.txt for processing.
2. **Vector Storage (ChromaDB)**
   * Stores and manages **vector embeddings** of the website’s content.
   * Uses **RecursiveCharacterTextSplitter** with:
     + **Chunk size:** 500 characters
     + **Overlap:** 50 characters
   * This enables fast and efficient retrieval of relevant documents based on user queries.
3. **Embedding Model (Gemini API)**
   * Converts user queries and website content into embeddings using Google’s Gemini embedding model.
   * Ensures semantic similarity between user queries and stored data.
4. **RAG Pipeline**
   * **Retrieves relevant documents** from ChromaDB using **similarity search**.
   * Validates retrieved documents using gemini-2.0-flash-001.
   * Generates responses using gemini-2.0-flash-001.
5. **Fallback Mechanism (Tavily Search API)**
   * If no relevant documents are found in ChromaDB, the system queries Tavily Search API.
   * The search results are processed and summarized using gemini-2.0-flash-001.
6. **Response Generation**
   * Generates **final responses** after analyzing retrieved documents.
   * Displays results via the **Streamlit UI**.

**Technologies Used**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| Python | Core programming language |
| Selenium | Extract content from Dxfactor webiste URLs |
| ChromaDB | Vector storage for document retrieval |
| Google Gemini API | Embeddings & LLM-based answer generation |
| Tavily Search API | Web search fallback for out of box query |
| LangChain | LLM pipeline integration |
| Streamlit | Web-based User Interface |

**Installation & Setup**

**Prerequisites**

* Python 3.8+ installed on your system.
* Google API key with access to Gemini API.
* Tavily Search API key.
* Chrome WebDriver for Selenium-based web scraping.

**Installation Steps**

1. Clone the repository:
   * git clone <https://github.com/your-username/dxfactor.git>
   * cd dxfactor
2. make environment in python using command below in terminal:
   * python -m venv rag\_venv
   * in windows activate enviroment – rag\_venv\Scripts\activate
   * in linux activate environment – source rag\_venv/bin/activate
3. Install dependencies:
   * pip install -r requirements.txt
4. Set up environment variables gemini and tavily api key in .env:
   * GOOGLE\_API\_KEY=your\_google\_api\_key
   * TAVILY\_API\_KEY=your\_tavily\_api\_key
5. scrape data from website and store data in chromadb vector store then run
   * python dxfactor\_scraper.py
   * it will generate dxfactor\_data.txt file then run
   * give path of txt file and then run python text\_to\_chromadb.py
   * it will convert and store dxfactor\_data.txt data into chromadb vector store
6. Run the application:
   * streamlit run streamlit\_run.py

## Directory Structure

## DXFactor

|  |  |
| --- | --- |
| **Filename** | **Working of filename** |
| chromadb\_store/ | Persistent vector storage |
| .env | API keys & config |
| config.py | Configuration settings like prompt |
| gemini\_embedding.py | Gemini embedding model integration |
| gemini\_model.py | Gemini LLM integration |
| rag\_agent.py | Core RAG implementation |
| requirements.txt | Required dependencies |
| streamlit\_run.py | Streamlit UI script |
| tavily\_search.py | Tavily search API integration |
| vector\_database.py | Vector database management |
| dxfactor\_scraper.py | Scrape Website Data |
| text\_to\_chromadb.py | Store txt data into vector database |

**System Workflow**

1. **User Query Input**
   * The user enters a question into the Streamlit interface.
2. **Embedding & Search**
   * The query is converted into an embedding via Gemini API.
   * The system retrieves relevant documents from ChromaDB.
3. **Relevance Check & Answer Generation**
   * gemini-2.0-flash-001 use to check whether the retrieve data is relevant or not.
   * If retrieved documents are relevant gemini-2.0-flash-001 generates a response.
   * If no relevant documents are found, the Tavily Search API fetches external information.
4. **Response Display**
   * The chatbot presents the final answer in the Streamlit UI.

**Error Handling & Performance Optimizations**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| Ensuring relevant responses from scraped data | Used Gemini embeddings and ChromaDB retrieval and Google llm model |
| Handling out-of-domain queries | Integrated Tavily Search API as a fallback |
| Making the UI intuitive | Used Streamlit for a simple and interactive experience |

## ****Security Considerations****

* **API Keys:** Stored securely in .env file.
* **Data Privacy:** No sensitive user data is stored.

## ****Testing Strategy****

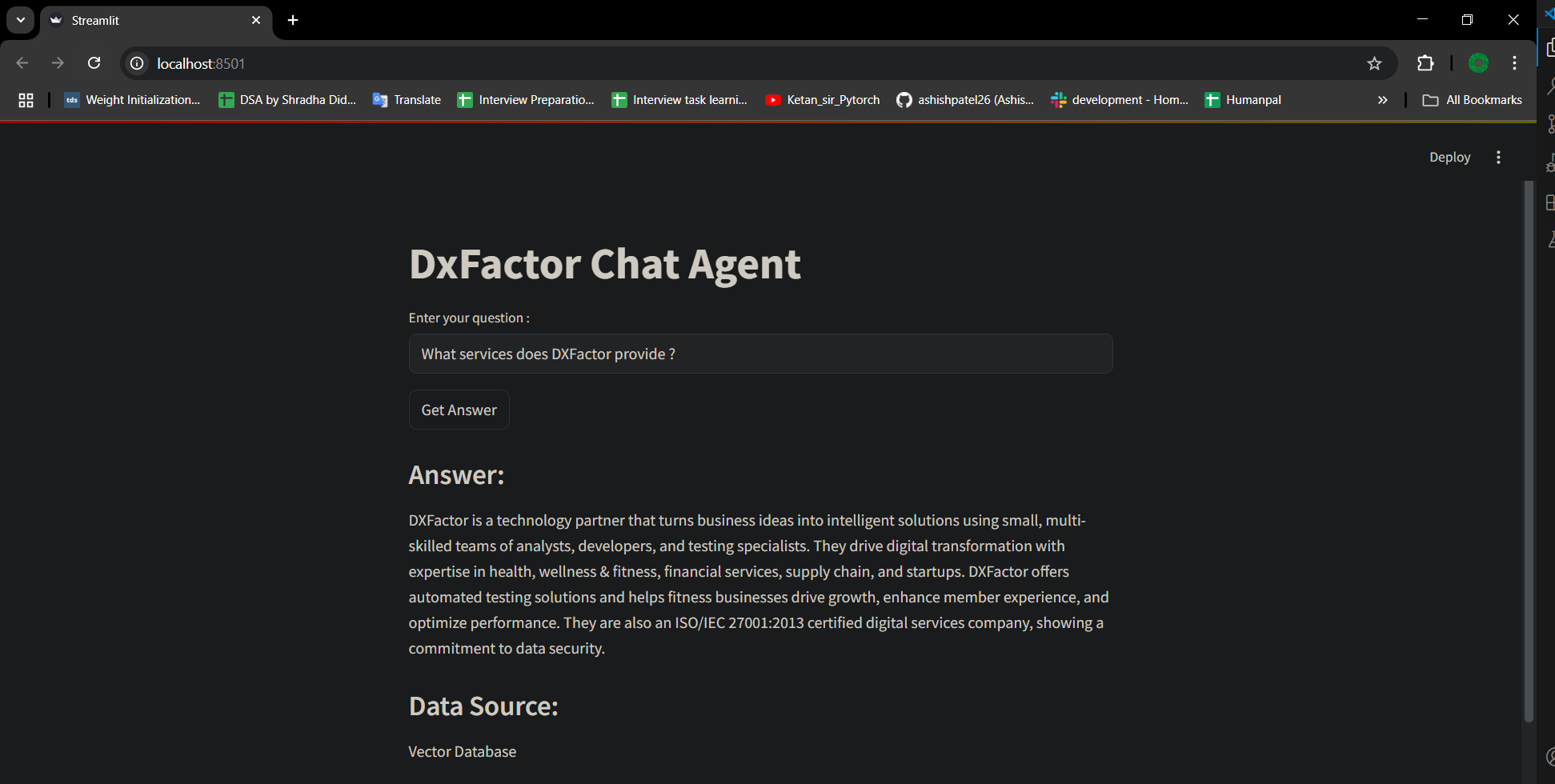
* **Unit Tests:**
  + Verify **ChromaDB retrieval**.
  + Validate **API responses** (Gemini & Tavily).
* **Integration Tests:**
  + Check **full RAG pipeline** (query → retrieval → response).

**Example Queries**

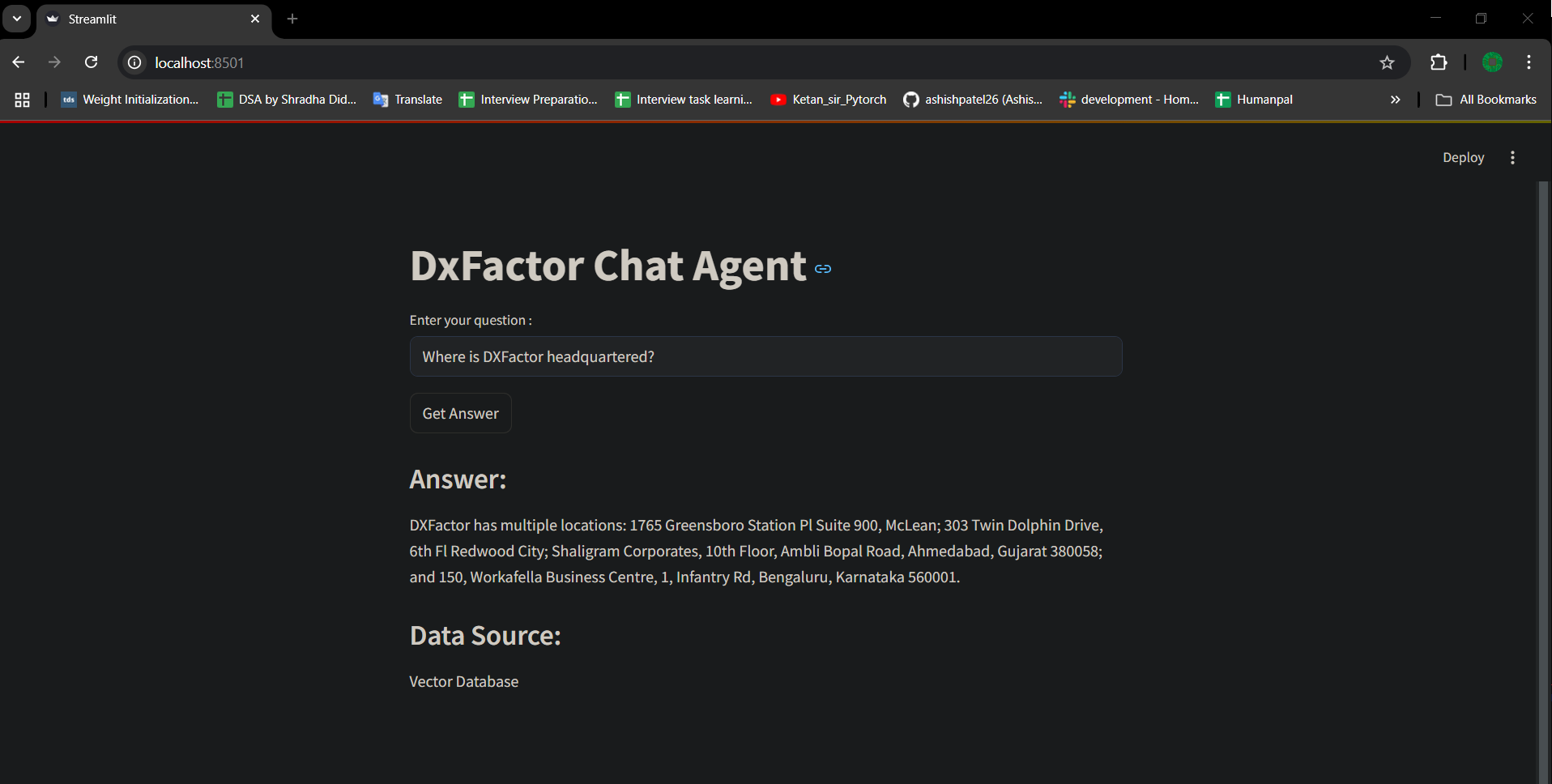
* What services does DXFactor provide?
* Where is DXFactor headquartered?
* Who are some key clients or partners mentioned on the website?
* Provide a brief overview of DXFactor's main industry.

**Screenshot of output from rag application**

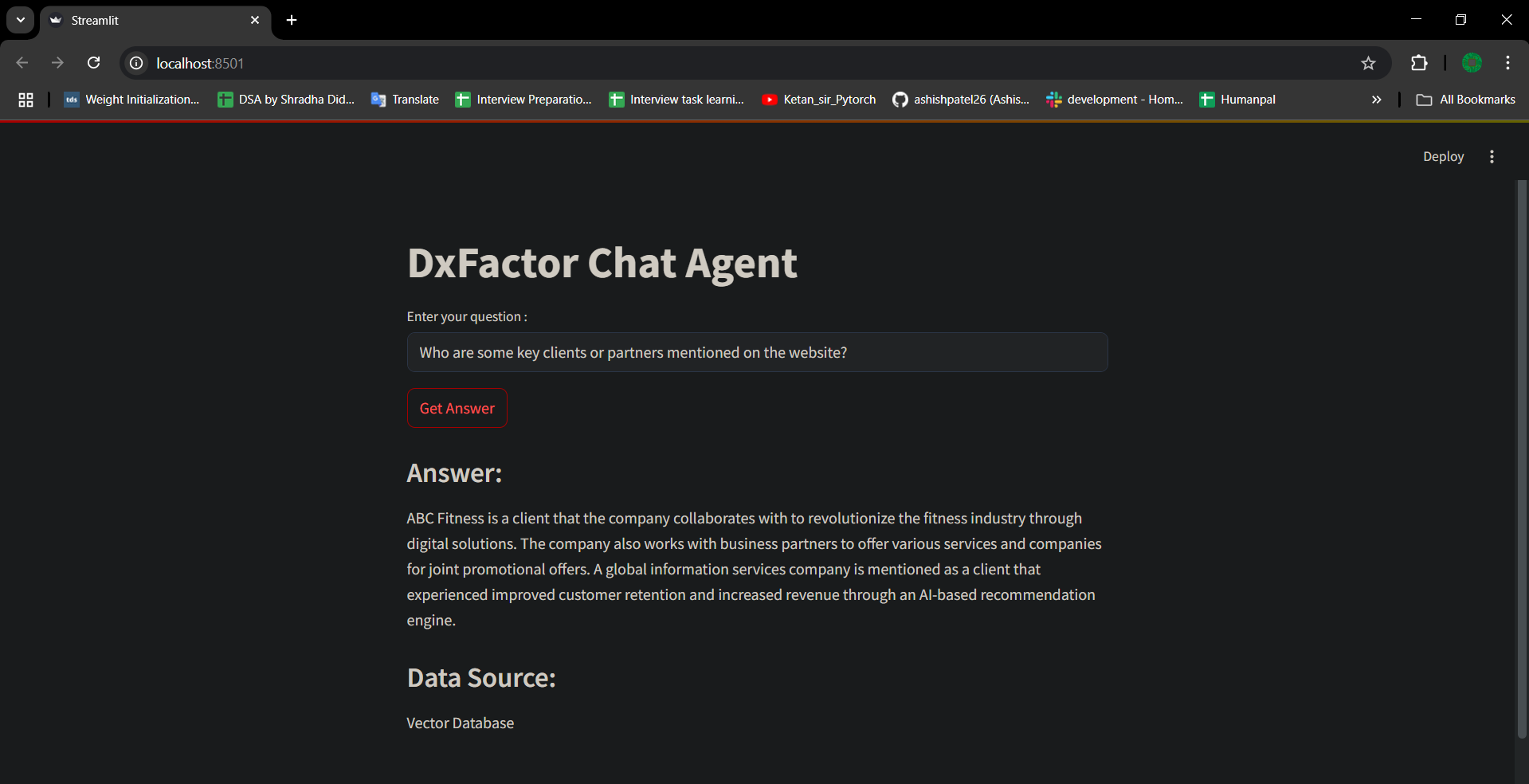
Query 1 : What services does DXFactor provide ?

****

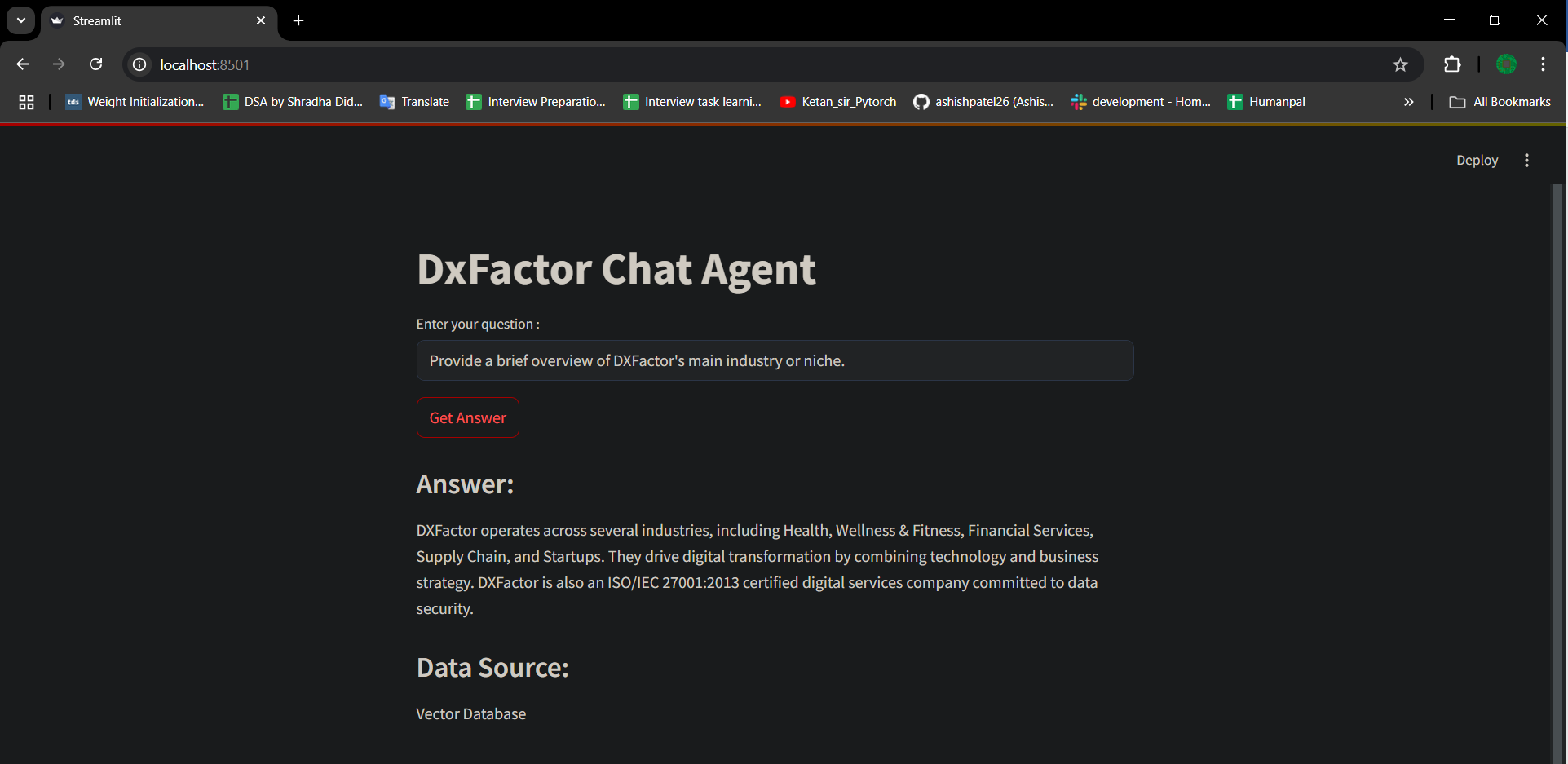
Query 2 : Where is DXFactor headquartered ?



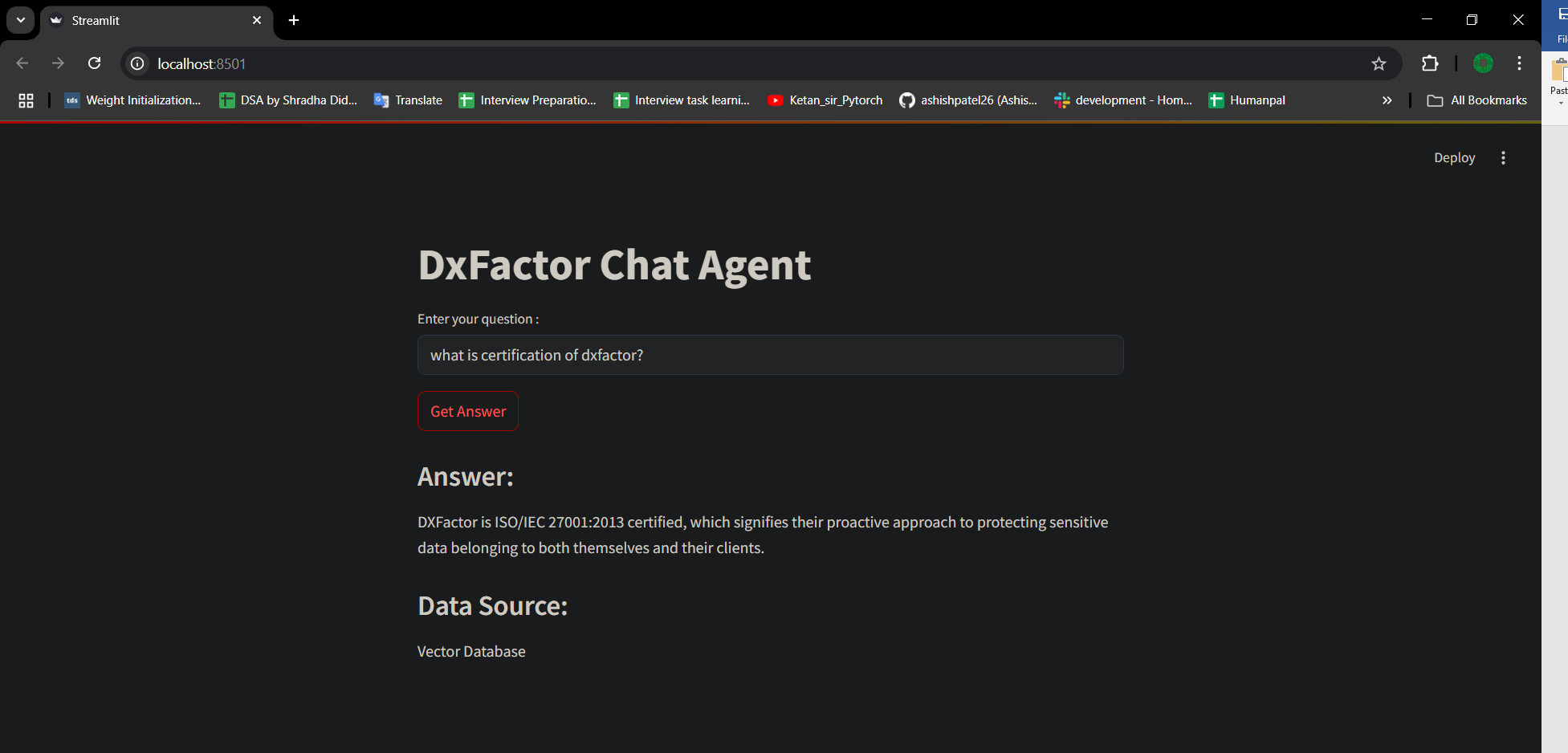
Query 3 : Who are some key clients or partners mentioned on the website?



Query 4 : Provide a brief overview of DXFactor's main industry or niche.



Query 5 : What is certification of dxfactor ?



Query 6 : who is caption of india cricket team ? (outside scope)

