Stony Brook University Chatbot

Phase 1: Data Extraction and Storage Optimization

Goal: Enhance data retrieval and storage mechanisms to improve response quality and efficiency.

1. Web Scraping Expansion:
    - Task: Extend the current web scraping functionality to cover various academic departments and majors at Stony Brook University.
    - Steps:
        - Identify all major-related URLs and sources on the university website.
        - Update the web_crawler.py to scrape information for specific departments, programs, and faculty.
        - Break down extracted data into structured document chunks for future processing.
        - Set up scripts for web scraping and run these scripts through jobs on Github Actions
2. **Explore Alternative Storage Solutions:**
    - Task: Research and implement alternatives to ChromaDB for storing embeddings.
    - Steps:
        - Compare vector databases like Pinecone, Weaviate, FAISS, or Milvus.
        - Analyze performance, scalability, cost, and integration ease for each solution.
        - Implement a prototype to test one alternative database and benchmark performance against ChromaDB.

Phase 2: User Profile System and Authentication

Goal: Build a user profile and authentication system to enable personalized interactions and data handling.

1. User Profile Creation:
    - Task: Develop a user profile system to store personalized information (e.g., major, interests).
    - Steps:
        - Design the structure of a profile that includes user preferences and previous inquiries.
        - Store these profiles in Firebase or another cloud-based backend.
        - Add dynamic responses that use this data to give more contextually relevant answers.
2. Firebase Authentication:
    - Task: Implement a user authentication system using Firebase to manage user logins, sessions, and security.
    - Steps:
        - Set up Firebase Authentication for email, Google, or social login methods.
        - Integrate authentication with the Streamlit frontend for a seamless login experience.
        - Ensure user sessions are securely handled and persisted.

Phase 3: Model Upgrades and Accuracy Improvements

Goal: Explore ways to improve the chatbot's underlying model for better response accuracy and efficiency.

1. Research on HuggingFace Models:
    - Task: Investigate HuggingFace models for potential improvements.
    - Steps:

        - Review models such as GPT-Neo, T5, or BERT that are suitable for conversational AI.
        - Compare model performance in terms of accuracy, context understanding, and response generation.
        - Experiment with fine-tuning these models to cater to the specific university information domain.

2. Evaluate Model Accuracy and Efficiency:
    - Task: Research cutting-edge models for balancing accuracy and efficiency.
    - Steps:
        - Analyze models with smaller footprints but high accuracy (e.g., LLaMA, Falcon, or Mistral).
        - Set up metrics to measure performance, latency, and resource consumption in your application.
        - Prototype and evaluate their suitability for your use case.

Phase 4: AWS Integration and Fine-Tuning Exploration

Goal: Investigate AWS services for deploying models and fine-tuning capabilities, considering cost, performance, and scalability.

1. AWS Services Research for Fine-Tuning:

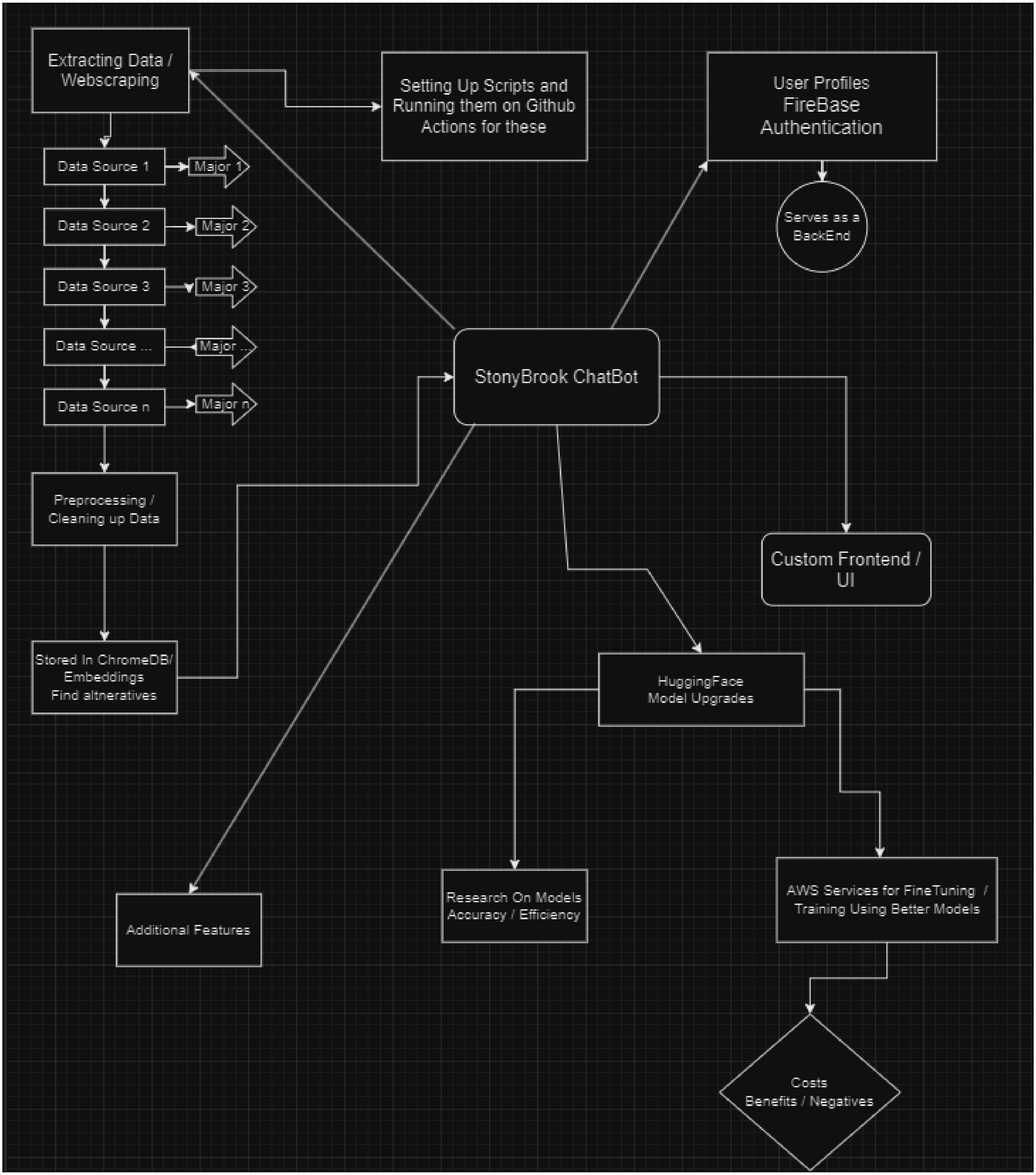
- Task: Explore the use of AWS services like SageMaker, Lambda, and EC2 for fine-tuning models.
- Steps:
  - Research the benefits and limitations of AWS SageMaker for training and deploying models.
  - Set up an instance for model fine-tuning, and evaluate its effectiveness in improving model accuracy.
  - Assess the costs of scaling the model using SageMaker, and compare it to other cloud services.

2. Benefits and Costs of AWS vs. Alternatives:
   - Task: Analyze the overall cost and efficiency trade-offs of using AWS compared to other cloud providers (e.g., Google Cloud, Azure).
   - Steps:
     - Perform a detailed comparison on infrastructure cost, model deployment time, and maintenance.
     - Weigh the pros and cons of AWS fine-tuning for scaling purposes, versus handling this on other platforms.
     - Make a final decision on the best cloud platform for future scalability and optimization.

Stretch Goal (Ongoing Maintenance and Improvements):

- Continuously evaluate and adjust scraping, user data, model performance, and cloud services to ensure that the chatbot remains efficient, accurate, and scalable.

Diagram :



Extracting Data / Webscraping
Setting Up Scripts and Running them on Github Actions for these
User Profiles FireBase Authentication
Data Source 1
Major 1
Data Source 2
Major 2
Data Source 3
Major 3
Data Source ...
Major ...
Data Source n
Major n
Serves as a BackEnd
StonyBrook ChatBot
Preprocessing / Cleaning up Data
Custom Frontend / UI
Stored In ChromeDB/ Embeddings Find altneratives
HuggingFace Model Upgrades
Research On Models Accuracy / Efficiency
AWS Services for FineTuning / Training Using Better Models
Additional Features
Costs Benefits / Negatives

Comparison of Vector Databases

Feature	Pinecone	Weaviate	FAISS	Milvus	ChromaDB
Performance	High for distributed systems; low latency.	Real-time vector search optimized for text data	Optimized for local deployments; slower at scale.	Efficient for large datasets; Slower at scale.	Best for smaller-scale use cases; user-friendly APIs.
Scalability	Cloud-native; built-in auto-scaling	Scales modularity; supports cloud and local setups.	Limited scalability; mainly used for research	High scalability with cluster support.	Suitable for moderate-scale applications.
Cost	Subscription based; charges depend on usage	Open source; cloud services may incur costs.	Free; compute-intensive for large datasets	Open source but requires cluster management.	Free and open source; low cost for prototyping.
Integration Ease	SDKs for Python, Javascript, etc; easy to use.	REST and GraphQL APIS; straightforward to integrate	Manual setup; technical expertise required.	APIs available; cluster setup and adds complexity.	Simple Python APIs; integrates seamlessly.
Indexing Speed	Fast with pre-optimized cloud indexing.	Moderately fast; customizable vector search	High speed for small-scale data	Fast for large-scale data.	Quick setup and indexing for smaller datasets.
Query Latency	Low latency for distributed queries.	Consistent low latency for semantic queries.	Low latency but struggles with large datasets.	Handles billions of vectors efficiently.	Low latency for moderate data sizes.
Target Use Cases	Large-scale commercial systems, real-time analytics	Text-based semantics searches and modular AI systems	Research, small-scaled deployments, academic use.	Enterprise-level applications requiring scalability.	Prototyping, education, and moderate-scale semantic tasks

Benchmarking Focus :

- Query Latency: Compare response times for semantic queries.
- Indexing Speed: Measure time taken to index datasets of increasing size.
- Memory Usage: Evaluate resource consumption during indexing and querying.
- Ease of Integration: Assess setup complexity and time to functional implementation.

Key Insights :

- ChromaDB is u ser-friendly, ideal for prototyping, and cost-efficient for small to medium datasets.
- Pinecone and Milvus excel in scalability and performance for large datasets but require more resources.
- Weaviate strikes a balance between performance and integration simplicity.
- FAISS is effective for local use and research but lacks cloud-native scalability.

Recommendations for Educational Chatbot Enhancement

1. Prototype: Start with ChromaDB for quick integration and moderate-scale datasets.
2. Scaling: Consider Pinecone or Milvus if the dataset size or user base grows significantly.
3. Optimization: Use sentence-transformer embeddings for vector generation to maximize semantic accuracy.

Links to Academic Datasets :

instead of adding documents manually to `store_docs()` each time, you can make the process more dynamic and modular by maintaining a configuration file (e.g., a JSON or YAML file) or a database to store the URLs or document paths. This way, you can easily update, remove, or add new documents without modifying the code.

```
https://www.stonybrook.edu/
```

https://www.stonybrook.edu/sb/bulletin/current/academicprograms/cse/degreesandrequirements.php

https://www.stonybrook.edu/academics/majors-minors-and-programs/?utm_source=chatgpt.com#UndergraduatePrograms

Extract all this data