Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

Ecole Supérieure d’Informatique et du Numérique

**AI BOOK RECOMMENDER**

TEAM: **GL**(YAHYA CHERKAOUI – MOHAMED ADAM BENADDI)

**CYB**(AYMANE OUKHATOU)

**BIG DATA & AI**(RACHID SEFRIOUI)

**Encadré par:**

M. Hakim HAFIDI

M. Hamza GAMOUH

M. Yasser ADERGHAL

# Lien github: https://github.com/aymen-okt/BiblioGem\_Chatbot.git

# AI Book Recommender - Project Summary

## Abstract

This project involves the development of an AI-powered book recommender system. It leverages advanced machine learning techniques to provide context-aware book recommendations through a real-time chat interface. The system aims to enhance user engagement by offering personalized book suggestions based on user preferences and interaction history. By integrating state-of-the-art AI technologies, the project seeks to revolutionize the way users discover and explore books.

## Contents

1. Introduction
2. Architecture
3. Core Features
4. Data Flow
5. Key Components
6. Features
7. Technical Implementation
8. Security
9. Performance
10. Challenges and Solutions
11. Requirements
12. Analysis and Design
13. Implementation

## 1. Introduction

The AI Book Recommender is designed to enhance user experience by providing personalized book recommendations based on user input and context. In today’s digital age, the abundance of available books can be overwhelming for readers. This project addresses the challenge of book discovery by utilizing AI to understand user preferences and deliver tailored recommendations. The system integrates state-of-the-art AI technologies to understand and respond to user queries effectively, making book exploration more intuitive and enjoyable.

## 2. Architecture

### Frontend

* Built with HTML, CSS, bootstrap and JavaScript
* Responsive design with dark/light mode support
* Real-time chat interface
* Chat history sidebar
* Animated interactions and loading states

The frontend is designed to provide a seamless user experience, with a focus on accessibility and responsiveness. The chat interface allows users to interact with the AI in real-time, while the chat history sidebar enables easy navigation through past conversations. The use of animated interactions and loading states enhances the visual appeal and user engagement.

### Backend

* Flask server (Python)
* Gemini Pro AI integration
* FAISS for efficient similarity search
* Sentence Transformers for text embeddings
* Pandas for data management
* MongoDB

The backend architecture is built on a robust Flask server, which handles user requests and processes them using advanced AI models. The integration of Gemini Pro AI allows for natural language understanding, while FAISS and Sentence Transformers enable efficient similarity search and text embedding generation. Pandas is used for data management, ensuring smooth handling of large datasets, and MongoDB is incorporated for scalable storage and efficient data retrieval.

## 3. Core Features

### Book Recommendations

* Utilizes FAISS for fast similarity search
* Provides top k similar books based on user queries

The book recommendation feature leverages FAISS to perform fast similarity searches, identifying books that closely match user queries. By analyzing user input and context, the system provides a list of top k similar books, enhancing the discovery process.

### Conversation Management

* Stores context per chat
* Generates unique ID for each chat

Effective conversation management is achieved by storing context for each chat session. This allows the system to maintain continuity and provide relevant responses, even in follow-up interactions. Each chat is assigned a unique ID, ensuring that context is preserved across sessions.

### Context Awareness

* Maintains conversation history
* Understands follow-up questions
* Remembers previous recommendations
* Adapts responses based on context

Context awareness is a key feature of the system, enabling it to understand and respond to follow-up questions. By maintaining a history of interactions, the AI can adapt its responses based on previous recommendations and user preferences, providing a more personalized experience.

## 4. Data Flow

1. **User Input**
   * Captures user input
   * Shows thinking animation
   * Sends request to backend
   * Displays response and recommendations

The data flow begins with capturing user input through the chat interface. A thinking animation is displayed while the system processes the request, enhancing user engagement. The input is then sent to the backend for processing, and the response, along with book recommendations, is displayed to the user.

1. **Backend Processing**
   * Query preprocessing
   * Context analysis
   * Book similarity search
   * Response generation
   * Returns formatted response

Backend processing involves several steps, starting with query preprocessing to clean and format the input. Context analysis is performed to understand the user’s intent and preferences, followed by a book similarity search using FAISS. The system then generates a response, which is formatted and returned to the frontend.

1. **Response Display**
   * Formats and displays book recommendations

The final step in the data flow is the display of the response and book recommendations. The system formats the recommendations for easy readability, ensuring that users can quickly identify books of interest.

## 5. Key Components

### Chat Interface

* Real-time message display
* Thinking animation during processing
* Book title formatting
* Recommendation cards

The chat interface is a critical component of the system, providing real-time message display and interactive features. A thinking animation is shown during processing, and book titles are formatted for emphasis. Recommendation cards present book suggestions in a visually appealing manner.

### Book Database

* Stored in CSV format
* Contains titles, summaries, categories
* Vectorized for similarity search

The book database is stored in CSV format, containing essential information such as titles, summaries, and categories. The data is vectorized to facilitate efficient similarity searches, enabling the system to quickly identify relevant books.

### AI Integration

* Uses Gemini Pro for natural language understanding
* Generates contextual responses
* Maintains conversation flow

AI integration is at the heart of the system, with Gemini Pro providing natural language understanding capabilities. The AI generates contextual responses, maintaining a smooth conversation flow and enhancing user interaction.

## 6. Features

1. **Smart Recommendations**
   * Content-based similarity
   * Category matching
   * Context-aware suggestions

Smart recommendations are achieved through content-based similarity and category matching. The system provides context-aware suggestions, ensuring that recommendations align with user preferences and past interactions.

1. **Conversation Handling**
   * Natural dialogue
   * Follow-up questions
   * Casual conversation support

The system supports natural dialogue and follow-up questions, allowing for seamless conversation handling. Casual conversation support enables the AI to engage users in friendly interactions, enhancing the overall experience.

1. **User Experience**
   * Dark/light mode
   * Chat history
   * Responsive design
   * Loading animations

User experience is prioritized through features like dark/light mode, chat history, and responsive design. Loading animations provide visual feedback during processing, keeping users informed and engaged.

1. **Data Management**
   * Local storage for chat history
   * Context persistence
   * Efficient search indexing

Data management is handled through local storage for chat history and context persistence. Efficient search indexing ensures quick retrieval of relevant information, supporting smooth system operation.

## 7. Technical Implementation

### Frontend Technologies

* HTML5
* CSS3 (with custom variables)
* Vanilla JavaScript
* Web Storage API

The frontend is built using HTML5, CSS3, and Vanilla JavaScript, with custom variables for styling. The Web Storage API is used for managing local data, supporting features like chat history and context persistence.

### Backend Technologies

* Flask
* FAISS
* Sentence Transformers
* Google Generative AI
* Pandas
* MongoDB

The backend leverages Flask for server-side processing, with FAISS and Sentence Transformers for similarity search and text embedding. Google Generative AI is integrated for advanced natural language processing, while Pandas handles data management tasks, and MongoDB is used for efficient storage and retrieval of data.

### AI/ML Components

* Embedding generation
* Similarity search
* Natural language processing
* Context management

AI/ML components include embedding generation, similarity search, and natural language processing. Context management is a key focus, enabling the system to provide personalized and context-aware recommendations.

## 8. Security

* Environment variables for API keys
* Input sanitization
* Error handling
* Secure data storage

Security measures include the use of environment variables for API keys, input sanitization, and error handling. Secure data storage practices are implemented to protect user information and ensure system integrity.

## 9. Performance

* Efficient similarity search with FAISS
* Optimized response generation
* Minimal dependencies
* Client-side caching

Performance is optimized through efficient similarity search with FAISS and response generation techniques. Minimal dependencies reduce system complexity, while client-side caching improves response times and user experience.

## 10. Challenges and Solutions

### Challenges

1. **Chat Context Management**: The chat system was treating each prompt as if it were the first in the conversation, leading to a lack of continuity.
2. **Dataset Querying**: The system was overly reliant on the dataset, responding without deeper processing or context understanding.
3. **Focus on Books**: Initially, the AI was not focused solely on books, leading to irrelevant responses.

### Solutions

1. **Improved Context Management**: The chat system now stores context per chat session using a chatContexts object, which tracks the last topic, category, follow-up status, and recommendations. This allows the system to maintain conversation history and handle follow-up questions effectively.
2. **Advanced Query Processing**: The backend processing includes query preprocessing, context analysis, and book similarity search. The use of FAISS and Sentence Transformers enables efficient similarity search and natural language processing, ensuring meaningful interactions beyond simple dataset lookups.
3. **Refined Focus**: The dataset and recommendation logic have been adjusted to prioritize book-related queries and responses. The ContextAwareBookRecommender class ensures that the AI remains focused on book-related content, providing context-aware book recommendations.

## 11. Requirements

### Functional Requirements

1. The system must provide personalized book recommendations based on user input and context.
2. It should maintain a conversation history to support context-aware interactions.
3. The chat interface must be responsive and support real-time interactions.
4. The system should handle follow-up questions and adapt responses based on previous interactions.

### Non-Functional Requirements

1. The system should be scalable to handle multiple users simultaneously.
2. It must ensure data security and privacy, particularly for user interaction data.
3. The response time should be optimized for a seamless user experience.

## 12. Analysis and Design

### System Analysis

The AI Book Recommender system is designed to address the challenge of book discovery by leveraging AI to understand user preferences. The analysis phase involved identifying key user needs, such as personalized recommendations and context-aware interactions. The system architecture was designed to integrate advanced AI technologies, ensuring efficient processing and response generation.

### System Design

The system is divided into frontend and backend components. The frontend is built with HTML, CSS, and JavaScript, focusing on a responsive and accessible user interface. The backend, developed using Flask, integrates AI models for natural language processing and similarity search. Key design considerations included ensuring scalability, security, and performance optimization.

## 13. Implementation

### Frontend Implementation

The frontend was implemented using HTML5, CSS3, and Vanilla JavaScript. Key features include a real-time chat interface, dark/light mode support, and animated interactions. The Web Storage API is utilized for managing local data, such as chat history and context persistence.

### Backend Implementation

### The backend is powered by a Flask server, which handles user requests and processes them using AI models. FAISS and Sentence Transformers are employed for similarity search and text embedding, while Google Generative AI provides advanced natural language processing capabilities. Pandas is used for efficient data management, ensuring smooth handling of large datasets, and MongoDB is integrated for scalable storage and retrieval of data.

### AI Integration Implementation

AI integration is a critical component, with Gemini Pro AI enabling natural language understanding and contextual response generation. The system maintains conversation flow and adapts to user preferences, providing a personalized experience. The implementation focused on optimizing AI model performance and ensuring seamless integration with the backend architecture.

**UI&UX Demonstration**

**The principal interface** :user-friendly, and focused on natural conversation about books and reading recommendations.

A screenshot of a computer

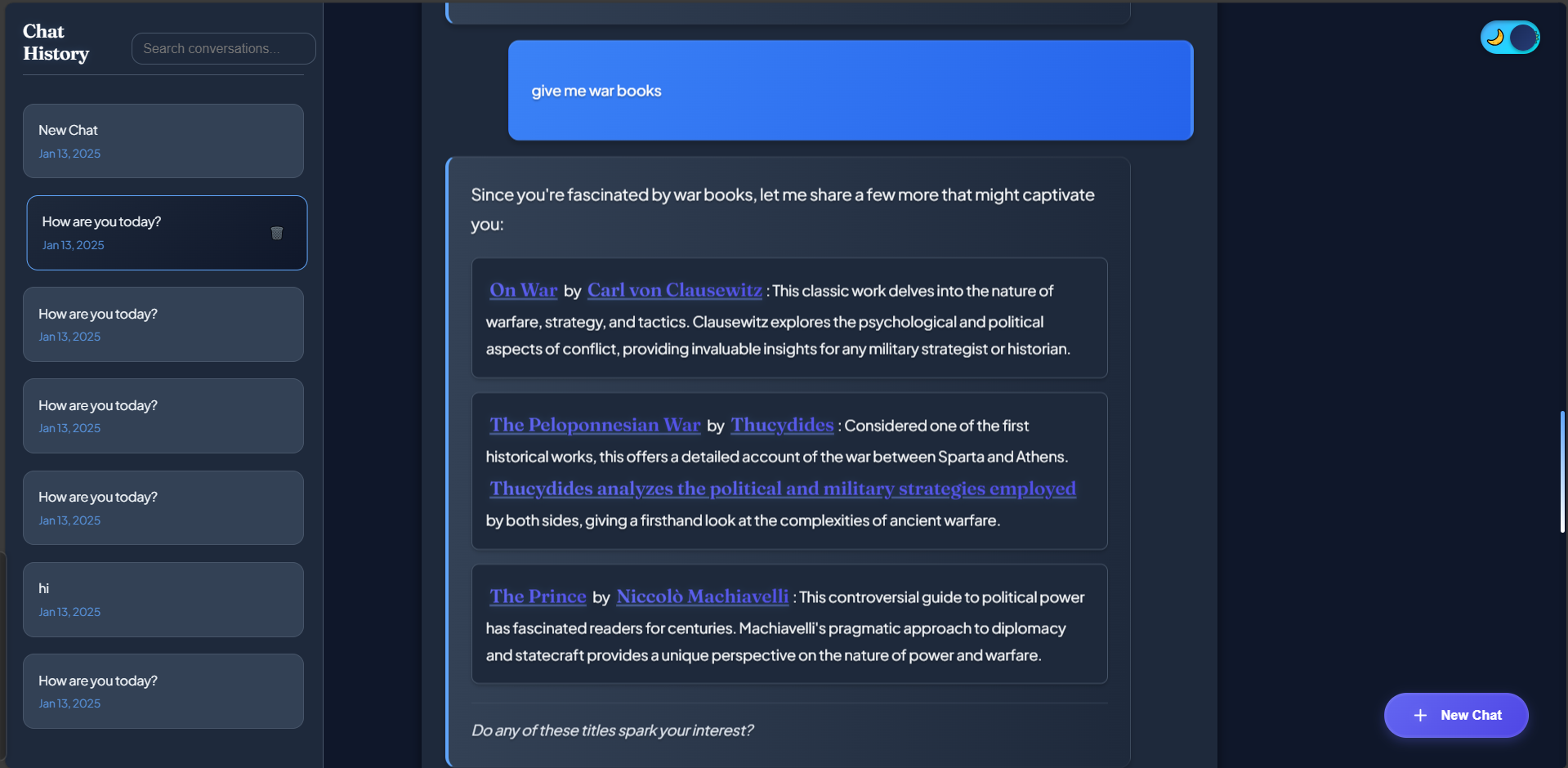
Description automatically generated

A screenshot of a chat

Description automatically generated

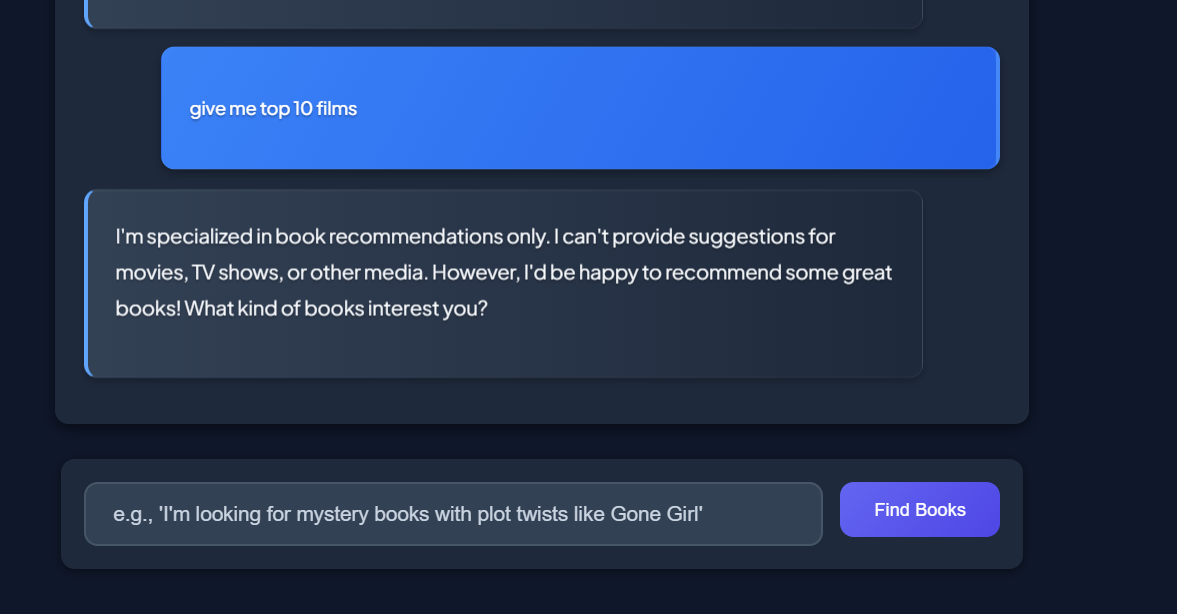
A screenshot of a computer

Description automatically generated



**Books specialization:**

When users request non-book recommendations, the chatbot responds with a friendly but clear message that establishes its limitations while redirecting the conversation towards books



The chatbot is always showing other random suggestions after the answer.

A screenshot of a chat

Description automatically generated

## Conclusion

The AI Book Recommender project successfully integrates modern AI technologies to deliver a seamless and personalized user experience. It demonstrates the potential of AI in enhancing digital interactions and providing valuable recommendations based on user context and preferences. The project sets a foundation for future developments in AI-driven book discovery, with opportunities for further enhancements and feature expansions.