

**Author:** *Adham Said*, Software Engineer intern  
**Date:** October 2024  
**Company/Institution Name:** Bibliotheca Alexandrina

Note: This project was divided into two parts: the development of Blitz and the development of the front end for Blitz. I specifically worked on Blitz, and instructions for running Blitz are detailed in a separate document that will be provided. All the files for Blitz are included in this document, and the project does not contain a large amount of code.


 Comprehensive Development Documentation for Blitz Chatbot

Table of Contents

**Executive Summary.....2**

**Introduction.....2**

    Problem Statement..... 2

    Objectives..... 2

    Scope.....2

    Stakeholders..... 2

**Background or Literature Review..... 3**

**Requirements Analysis..... 3**

    Functional Requirements..... 3

    Non-Functional Requirements..... 4

    Use Cases/User Stories.....4

**Design.....5**

    Flow Chart.....5

    Sequence Diagram..... 6

    Use Case Diagram.....7

**Implementation..... 8**

    Language Models & Embedding Models Used.....9

**Suggestions..... 10**

Links and Resources..... 10

**Contact.....11**

# Executive Summary

This report outlines the development and implementation of the Bibliotheca Alexandrina chatbot, focusing on customer service and library support. The chatbot is designed to answer FAQs, provide library information in both English and Arabic, and assist users in navigating the library's services.

The chatbot successfully met key requirements, including real-time multilingual responses, user authentication, and support for FAQs. Recommendations for future work include expanding the chatbot's capabilities to handle more complex queries and integrating it further with internal library systems.

## Introduction

### Problem Statement

Bibliotheca Alexandrina receives numerous daily inquiries, leading to delays in response and inefficiency in serving visitors. A chatbot solution is required to provide instant assistance for common FAQs and library services to enhance visitor experience and improve operational efficiency.

### Objectives

- Develop a chatbot capable of answering FAQs about the library's services and facilities.
- Ensure the chatbot supports both English and Arabic to cater to a wide audience.

### Scope

The chatbot will be focused on providing customer service support and answering FAQs. It will not include complex administrative tasks such as book loans or renewals. Further integration with internal systems is outside the current scope but planned for future enhancements.

### Stakeholders

- **Library Visitors:** End users interacting with the chatbot for assistance and information.
- **Library Staff:** Users who benefit from the reduced load of direct customer inquiries.
- **Software Development Team:** Responsible for building and maintaining the chatbot.
- **Library IT Department:** Responsible for system integration and deployment.
- **Management:** Ensuring the chatbot aligns with the library's strategic goals for customer service improvement.

# Background or Literature Review

Chatbots have become widely used in customer service to automate responses to frequent inquiries, improving both efficiency and user satisfaction. Libraries, including Bibliotheca Alexandrina, are adopting these tools to provide quicker responses to user queries and to allow visitors to access library information without the need for direct human intervention.

Existing chatbot frameworks, including those powered by large language models (LLMs), have demonstrated effectiveness in multilingual environments, which is crucial for a library like Bibliotheca Alexandrina that serves an international audience. Research suggests that incorporating real-time conversational context and ensuring data privacy are critical for user trust in such systems.

## Requirements Analysis

### Functional Requirements

1. **Multilingual Support:** The chatbot must provide responses in both English and Arabic.
2. **FAQ Handling:** Answer frequent inquiries about services, membership, hours, and library facilities.
3. **Real-Time Responses:** Responses must be generated within 2 seconds to maintain conversation flow.
4. **Personalized Assistance:** Recognize returning users for more tailored responses (if logged in).
5. **User Authentication:** Provide authentication options (Google, email/password) for personalized experiences.
6. **Session History:** Display recent interactions for returning users.
7. **Security Against Prompt Injection:** Prevent malicious inputs and attacks.
8. **Context Awareness:** Maintain conversational context within a session to provide relevant answers.
9. **Error Handling:** Gracefully handle invalid inputs and system errors with helpful feedback and retry options.
10. **Event Inquiry:** Allow users to inquire about upcoming library events.
11. **Membership Assistance:** Assist users in inquiring and renewing memberships.
12. **Donation Information:** Provide information about making donations to the library.

## Non-Functional Requirements

1. **Performance:** Responses should be generated in under 2 seconds, with a peak load maximum of 5 seconds.
2. **Scalability:** The system should handle 100 concurrent users without performance issues.
3. **Reliability:** Ensure 99.9% uptime for chatbot and API services.
4. **Security:** Use HTTPS for all data transmissions and ensure end-to-end encryption for sensitive information.
5. **Accessibility:** Ensure the system meets WCAG 2.1 AA accessibility standards for usability by people with disabilities.
6. **Maintainability:** Modular design allowing components to be updated independently.
7. **Data Privacy:** Ensure GDPR compliance with user data and provide options to opt out.
8. **Localization:** Ensure proper handling of right-to-left (RTL) text and cultural nuances in Arabic.

## Use Cases/User Stories

1. **Use Case 1: Inquire About Membership Options**
  - a. **As a visitor**, I want to ask the chatbot about membership plans so I can choose the one that fits my needs.
2. **Use Case 2: Ask About Library Hours**
  - a. **As a visitor**, I want to inquire about the library's hours of operation so I can plan my visit.
3. **Use Case 3: Multilingual Query Handling**
  - a. **As an Arabic-speaking visitor**, I want to ask questions in Arabic so I can access information in my native language.
4. **Use Case 4: Retrieve User History**
  - a. **As a returning user**, I want to view my previous inquiries so I can continue the conversation without repeating myself.
5. **Use Case 5: Provide Feedback**
  - a. **As a user**, I want to leave feedback about the chatbot's responses to help improve its service.
6. **Use Case 6: Ask About Event Details**
  - a. **As a visitor**, I want to inquire about upcoming events so I can plan to attend.
7. **Use Case 7: Data Security Inquiry**
  - a. **As a privacy-conscious user**, I want to ask how my data is handled so I feel secure interacting with the chatbot.
8. **Use Case 8: Report a System Issue**
  - a. **As a user**, I want to report an issue with the chatbot so the support team can resolve it quickly.
9. **Use Case 9: Request Library Tour Information**
  - a. **As a first-time visitor**, I want to inquire about booking a library tour to explore the facilities.

## 10. Use Case 10: Membership Renewal Assistance

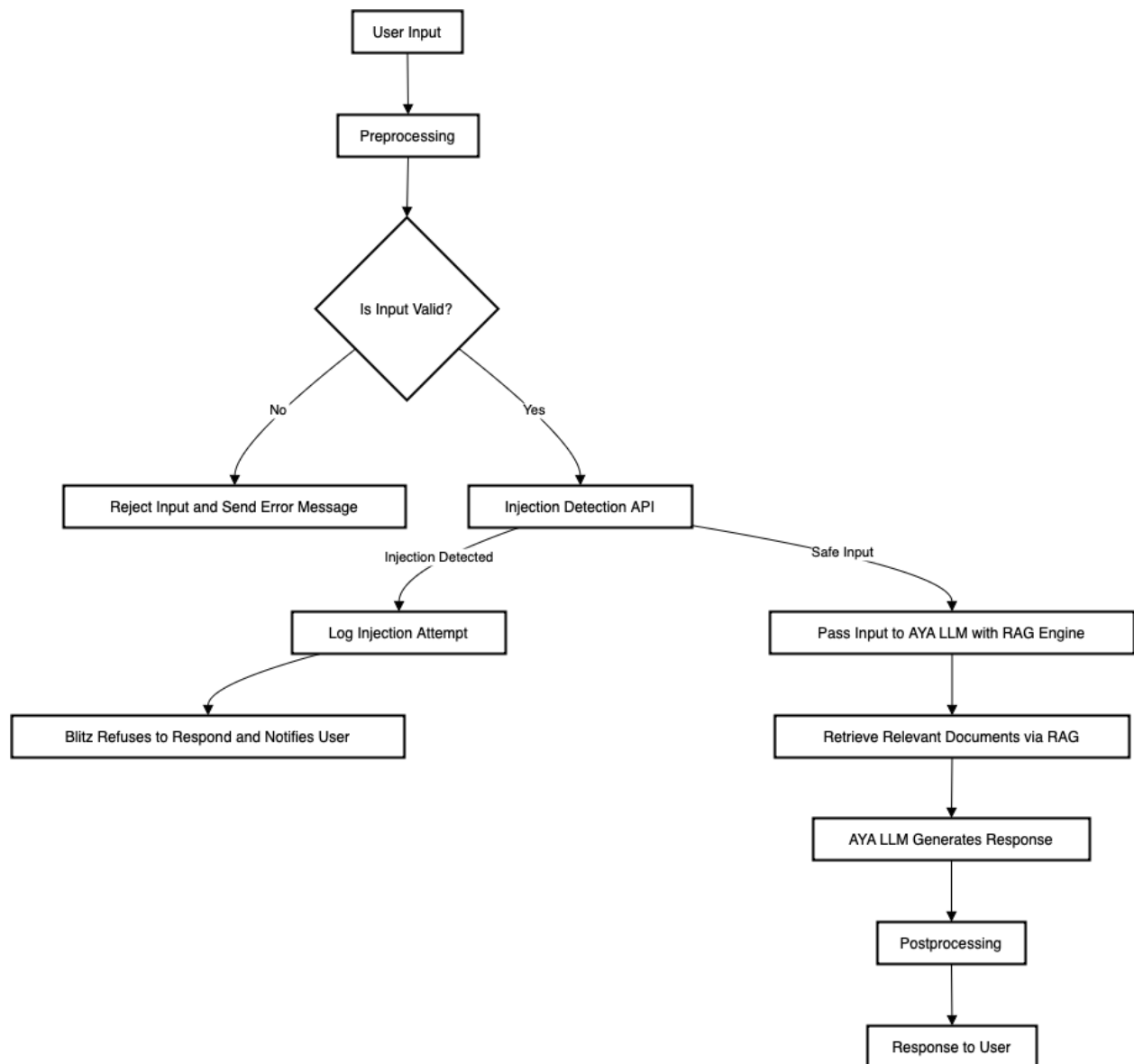
- a. **As a member**, I want to inquire about renewing my membership to continue accessing library services.

## 11. Use Case 11: Ask About Donations

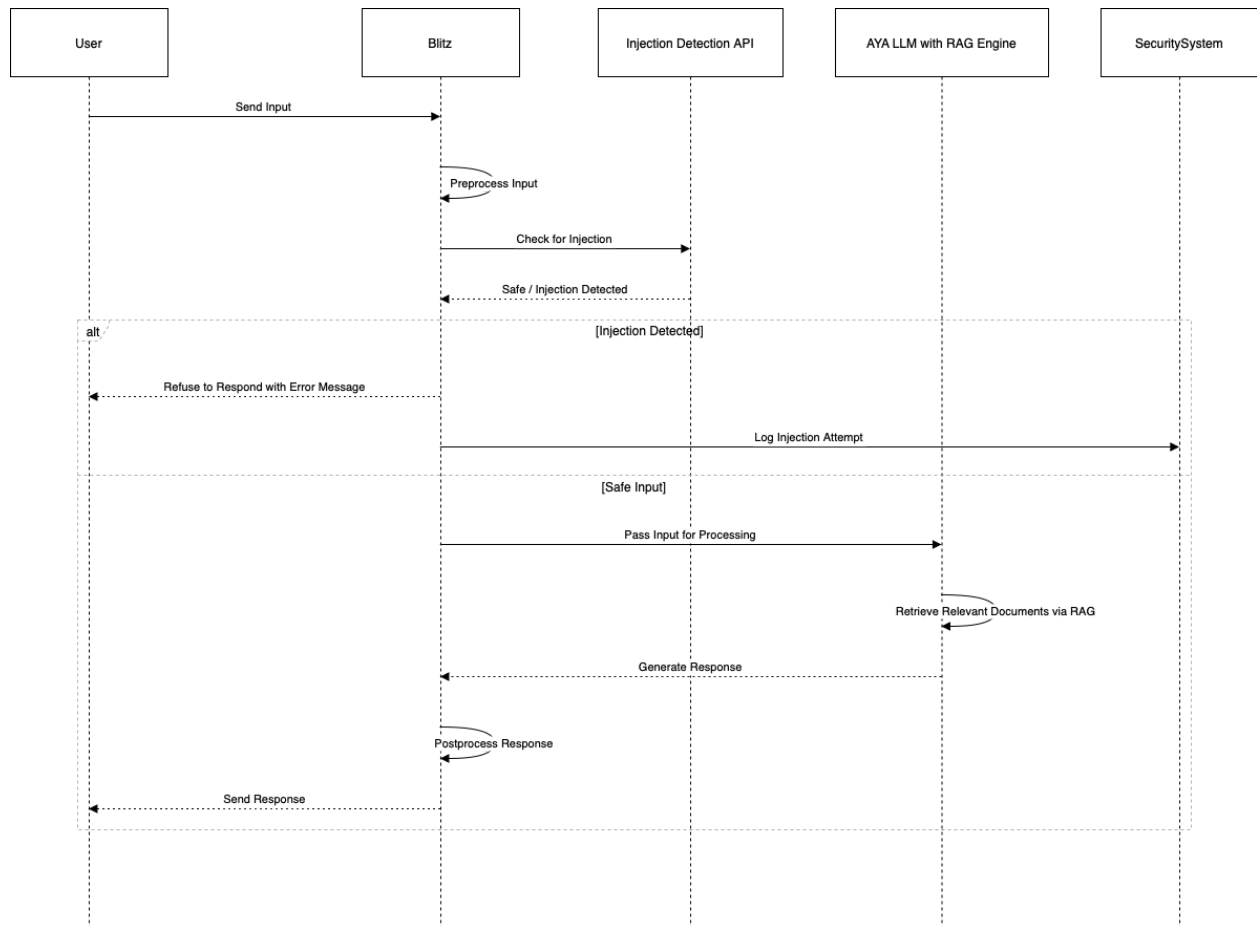
- a. **As a patron**, I want to ask the chatbot how to make a donation to the library to support its activities.

# Design

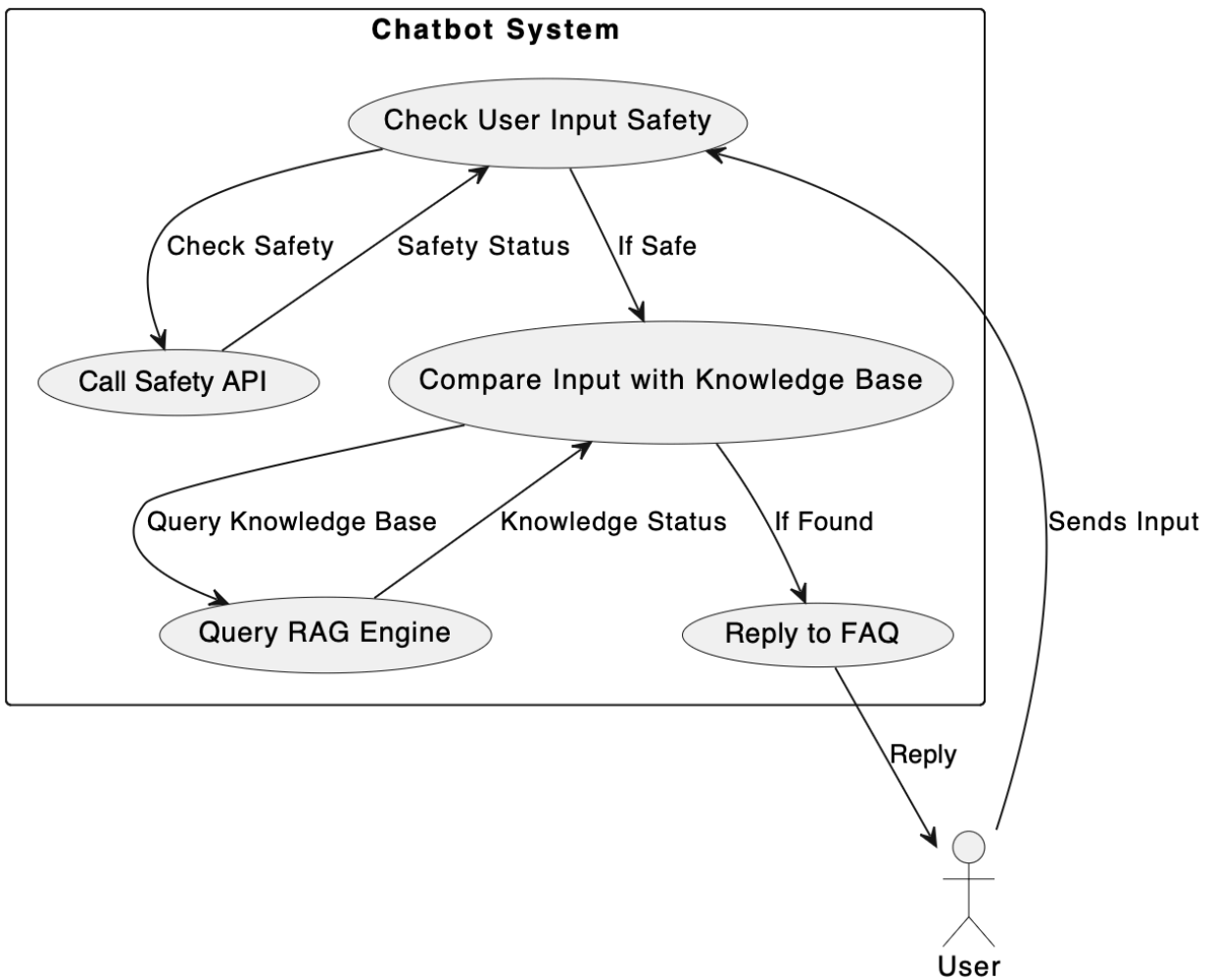
## Flow Chart



# Sequence Diagram



## Use Case Diagram



# Implementation

## Development Process

- **Methodology:** The project was developed using the **Scrum** framework, which emphasizes iterative progress, collaboration, and adaptability. Key practices included:
  - **Sprints:** Short development cycles to deliver incremental features.
  - **Daily Stand-ups:** Regular team meetings to discuss progress, blockers, and plans for the day.
  - **Sprint Reviews and Retrospectives:** Evaluation of completed work and discussion of potential improvements.

## Challenges and Solutions

- **H2oGPT:**
  - **Challenges:**
    - **Lack of RAG Support:** The platform did not support Retrieval-Augmented Generation, crucial for contextual accuracy.
    - **Prompt Injection Vulnerabilities:** There was a lack of built-in protections against prompt injection attacks.
    - **Server Performance Issues:** The Gradio server faced performance issues, and the OpenAI server did not provide full access to desired features.
  - **Solution:** H2oGPT was discontinued in favor of more advanced platforms.
- **LM Studio:**
  - **Overview:** LM Studio was selected for its superior performance and refined API endpoints.
  - **Challenges:**
    - **Lack of RAG Support:** Despite its benefits, LM Studio also lacked RAG functionality.
  - **Solution:** Further exploration of other tools was necessary due to the absence of RAG.
- **Custom RAG APIs:**
  - **Overview:** Custom RAG APIs were developed using Ollama, Langchain, LlamaIndex, and ChromaDB.
  - **Challenges:**
    - **Performance Issues:** The custom APIs provided average performance and lacked advanced features.
    - **Complexity:** The development and maintenance of these solutions were time-consuming.
  - **Solution:** Custom RAG APIs were replaced by the Ollama-Dify integration for a more robust solution.



- **AnythingLLM and OpenWebUI:**
  - **Overview:** These platforms were explored for their RAG capabilities.
  - **Challenges:**
    - **Poor Developer Support:** APIs were either non-functional or unavailable.
    - **Feature Limitations:** Despite their potential, both tools could not be effectively utilized.
  - **Solution:** Neither platform was adopted due to these limitations.
- **Ollama and Dify:**
  - **Overview:** Integrated to leverage both platforms' strengths.
  - **Key Features:**
    - **RAG Capabilities:** Enhanced the chatbot's ability to retrieve and generate contextually accurate responses.
    - **Model Customization:** Allowed extensive customization of models and integration of additional features for performance and security.
    - **Prompt Injection Protection:** Improved security against prompt injection attacks.
  - **Solution:** The integration was deemed the most effective solution for developing Blitz.

## Language Models & Embedding Models Used

- **Language Models:**
  - Initially used **Mistral 7B**, later transitioned to **LLaMA 3.1** for superior performance.
  - Ultimately adopted the **Aya model** for better handling of Arabic queries.
- **Embedding Models:**
  - Experimented with **Nomic Embed** and **Mxbai-Embed-Large**.
  - Initially used Mxbai but switched to **Nomic Embed** for its lighter footprint.
  - Currently using **Aya model** for language processing and **Nomic Embed** for embedding tasks, balancing efficiency and performance.

## Suggestions

- **Try Larger Models:**

- Consider experimenting with larger models such as aya 35B. These models may provide improved performance and accuracy.
- **Refine the System Prompt:**
  - Fine-tuning the system prompt can help minimize hallucinations. Ensuring the prompt is clear and well-defined will improve the model's reliability.
- **Adjust Model Parameters:**
  - **Top-K and Top-P Sampling:** Modify parameters like Top-K and Top-P to reduce randomness in the model's responses. This adjustment can make the model's outputs more focused and conservative.
- **Implement a Rerank Model:**
  - Utilize a rerank model to improve the relevance and accuracy of search results. Reranking involves initially retrieving a set of candidate results and then applying a more sophisticated model to reorder these results based on their relevance to the query.
- **Adopt Hybrid Search:**
  - Consider using hybrid search, which combines both vector search and traditional keyword-based search. This approach leverages the strengths of both methods to enhance search effectiveness, providing more precise and contextually relevant results.

## Links and Resources

- [Blitz Files, Knowledge Base & Demos](#)
- Mistral 7b ([HuggingFace](#)) ([Ollama](#))
- Aya 8b ([HuggingFace](#)) ([Ollama](#))
- Llama 3.1 ([HuggingFace](#)) ([Ollama](#))
- [Ollama](#)
- Nomic Embeddings Model ([HuggingFace](#)) ([Ollama](#))
- MixedBread Embeddings Model ([HuggingFace](#)) ([Ollama](#))
- [Dify](#) ([Documentation](#))
- Langchain ([Documentation](#))
- LlamaIndex ([Documentation](#))
- ChromaDB ([Documentation](#))
- [ZenGuard](#) ([Documentation](#))
- ProtectAi Debra v3 ([HuggingFace](#))
- [Rebuff](#)

## Contact

If you have any questions or need assistance, I am available to help. Please feel free to reach out to me at [Adham Afis](#) .