# Software Requirement Specification

June 7, 2024

| **Project ID & Name** | **Machine Failure Prediction** | | |
|---|---|---|---|
| Project Manager | Md. Mohoiminul Islam Chowdure | | |
| Customer Request Reference | | | |
| Prepared by | Abu Darda | Date | 7 June, 2024 |
| Reviewed by | | Date | |

# Contents

# 1 Requirement Scope Summary

## 1.1 Introduction

The purpose of this document is to define the Software Requirements Specification (SRS) for the Conversational Chatbot project. This document outlines the functional and non-functional requirements, system architecture, and detailed descriptions of the various components of the chatbot system.

## 1.2 Project Overview

The Conversational Chatbot project is designed to provide a robust, interactive chatbot capable of engaging users in meaningful conversations. It leverages a variety of data sources, including SQL databases, website content, and PDF documents, to create a comprehensive knowledge base. The system uses OpenAI/Langchain/Llama index for natural language processing, FAST API for backend services, and Streamlit for the frontend interface.

## 1.3 Scope

This document covers the following aspects of the Conversational Chatbot project:

- Gathering and processing data from multiple sources
- Creating a knowledge base using a vector database
- Implementing a conversational chatbot using advanced AI models
- Developing a backend server with FAST API
- Building a user-friendly frontend with Streamlit
- Ensuring the system meets both functional and non-functional requirements

## 1.4 Stakeholders

- **Project Manager:** Oversees the project execution and ensures all milestones are met.
- **Developers:** Implement the backend, frontend, and integration of AI models.
- **Testers:** Conduct thorough testing to ensure the chatbot functions as expected.
- **End Users:** Interact with the chatbot for various information retrieval and conversational purposes.

## 1.5 Definitions and Acronyms

- **AI:** Artificial Intelligence
- **API:** Application Programming Interface
- **SRS:** Software Requirements Specification
- **SQL:** Structured Query Language
- **FAST API:** A modern, fast (high-performance), web framework for building APIs with Python
- **Streamlit:** An open-source app framework for creating and sharing beautiful, custom web apps for machine learning and data science
- **NLP:** Natural Language Processing

### 1.6 References

- Project README file
- requirements.txt
- Conversational Chatbot GitHub Repository

### 1.7 Assumptions

- Users have a basic understanding of interacting with web applications and chatbots.
- The development team has expertise in Python, FAST API, Streamlit, and AI model integration.
- The necessary data sources (SQL files, websites, PDFs) are accessible and properly formatted.

### 1.8 Constraints

- The system must handle large volumes of data efficiently.
- The chatbot should provide accurate and relevant responses to user queries.
- Integration of various data sources must be seamless and real-time updates should be supported.
- The system must be scalable and maintainable.

### 1.9 Overview of the Document

This SRS document is structured as follows:

- **Chapter 1:** Requirement Scope Summary
- **Chapter 2:** Functional Requirements Identification
- **Chapter 3:** Functional Requirements Details
- **Appendices:** Additional supporting information and documents

## 2 Functional Requirements Identification

### 2.1 System Architecture Overview

The system architecture for the Conversational Chatbot project consists of several key components, including data sources, a backend server, a frontend interface, and the AI models that drive the chatbot's conversational capabilities.

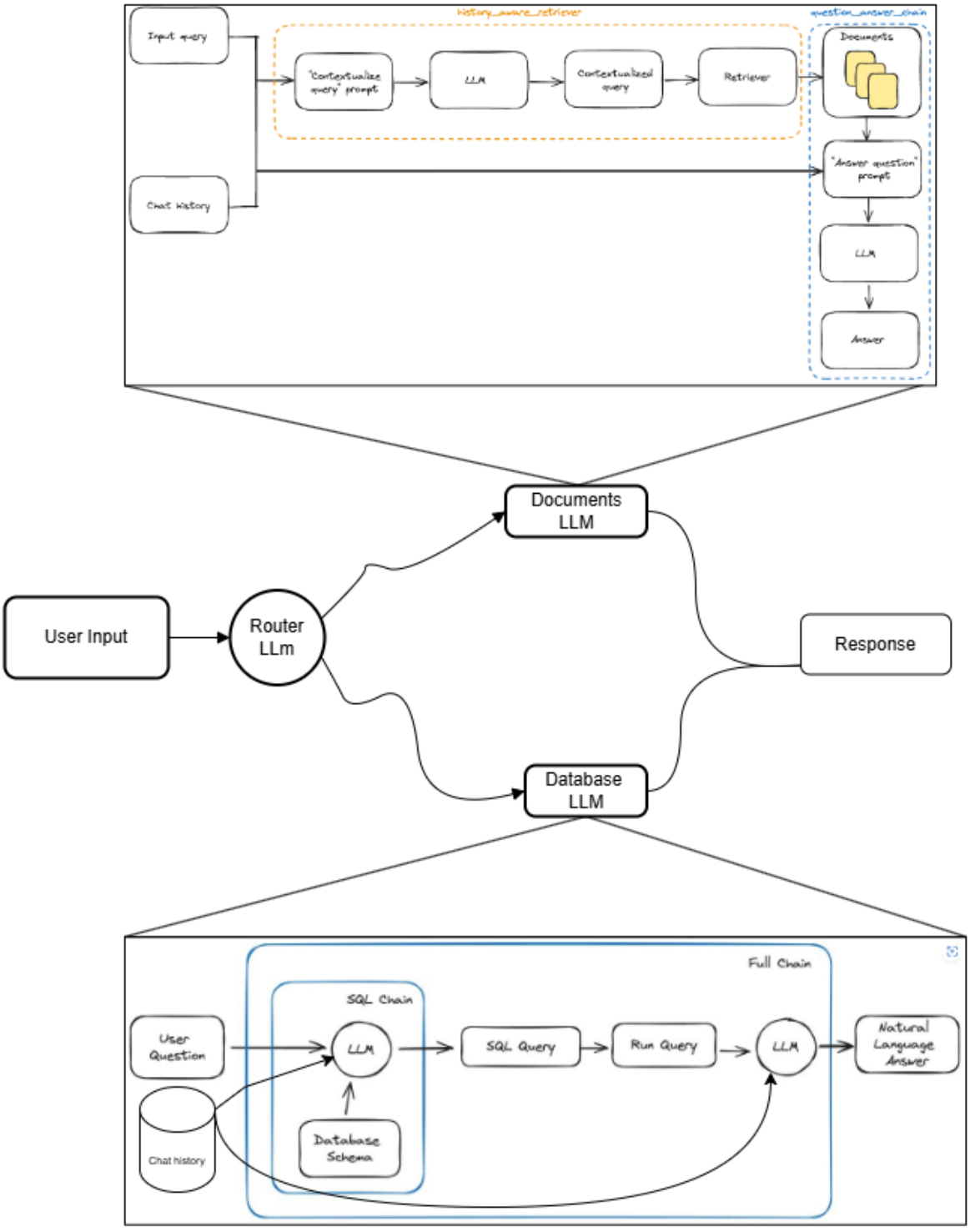


Figure 1: System Architecture Diagram

## 2.2 Data Sources

The chatbot utilizes various data sources to build its knowledge base:

- **SQL Database:** Contains product information stored in `products.sql`.
- **Web Scraping:** Gathers data from specific webpages (e.g., Nikles' about page, technologies page, luxury finishes page, and news page).
- **PDF Documents:** Includes warranty information from `Warranty.pdf`.

## 2.3 Backend Server

The backend server is implemented using FAST API and handles the following functionalities:

- Data processing and storage
- Integration with AI models
- API endpoints for frontend communication

## 2.4 Frontend Interface

The frontend interface, built using Streamlit, provides a user-friendly platform for interacting with the chatbot. It includes features such as:

- Text input for user queries
- Display of chatbot responses
- Interactive elements for enhanced user experience

## 2.5 AI Models

The chatbot leverages OpenAI/Langchain/Llama index for its natural language processing capabilities. These models enable the chatbot to understand and respond to user queries accurately.

## 2.6 Functional Requirements Summary

The following are the high-level functional requirements for the Conversational Chatbot project:

1. **Data Collection and Integration:**
    - Collect data from SQL databases, web scraping, and PDF documents.
    - Integrate collected data into a unified knowledge base.
2. **Backend Services:**
    - Implement API endpoints for data retrieval and chatbot interactions.
    - Process and store data efficiently.
3. **AI Model Integration:**
    - Utilize AI models for natural language processing.
    - Ensure the chatbot can handle various types of queries.
4. **Frontend Development:**
    - Develop a user-friendly interface using Streamlit.
    - Provide interactive elements for enhanced user engagement.

5. **Conversational Capabilities:**
    - Enable the chatbot to engage in meaningful conversations.
    - Ensure accurate and contextually relevant responses.

# 3 Functional Requirements Details

## 3.1 Data Collection and Integration

**Description:** The system must collect data from various sources including SQL databases, web scraping, and PDF documents, and integrate this data into a unified knowledge base.

**Inputs:**

- SQL files (e.g., `products.sql`)
- Webpages (e.g., `https://www.nikles.com/about/`)
- PDF documents (e.g., `Warranty.pdf`)

**Outputs:**

- A comprehensive knowledge base stored in a vector database.

**Functional Requirements:**

1. The system shall parse the SQL file `products.sql` and extract relevant product information.
2. The system shall scrape specified webpages and extract data such as company history, technology details, and news.
3. The system shall extract warranty information from the `Warranty.pdf` document.
4. The system shall integrate all collected data into a unified knowledge base.

**Dependencies:**

- Availability of SQL files, webpages, and PDF documents.
- Reliable internet connection for web scraping.

**Preconditions:**

- The system is properly configured and all necessary libraries are installed.

**Postconditions:**

- The knowledge base is updated with the latest data from all sources.

**Exception Handling:**

- If a data source is unavailable or an error occurs during data extraction, the system shall log the error and notify the user.

## 3.2 Backend Services

**Description:** The backend server shall handle data processing, storage, and API endpoints for communication with the frontend and AI models.

**Inputs:**

- User queries
- Data from the knowledge base

**Outputs:**

- Processed data
- Responses to API requests

**Functional Requirements:**

1. The backend shall provide API endpoints for retrieving product information.
2. The backend shall process user queries and fetch relevant data from the knowledge base.
3. The backend shall provide endpoints for resetting conversation.
4. The backend shall integrate with AI models to generate responses.
5. The backend shall store data efficiently and securely.

**Dependencies:**

- Proper configuration of the FAST API server.
- Integration with the vector database and AI models.

**Preconditions:**

- The backend server is running and all dependencies are met.

**Postconditions:**

- API requests are handled efficiently and responses are accurate.

**Exception Handling:**

- In case of server errors, the system shall log the error and return a meaningful error message to the user.

## 3.3 AI Model Integration

**Description:** The system shall integrate AI models for natural language processing to enable the chatbot to understand and respond to user queries.

**Inputs:**

- User queries
- Data from the knowledge base and database.

**Outputs:**

- Contextually relevant responses

**Functional Requirements:**

1. The system shall use LLM models to process and understand user queries.

2. The system shall accurately route user queries to generate responses from relevant knowledge base.
3. The system shall handle the case where it generates faulty database queries.
4. 
5. The system shall support conversational context, allowing for multi-turn interactions.
6. The model shall generate texts based on the context and knowledge base only.

**Dependencies:**

- Availability of trained AI models.
- Integration with the backend server.

**Preconditions:**

- The LLM models are properly integrated and configured.
- The routing LLM can properly route user queries to relevant chain.

**Postconditions:**

- The chatbot provides accurate and relevant responses to user queries.

**Exception Handling:**

- If the AI model fails to process a query, the system shall log the error and provide a fallback response.

## 3.4 Frontend Development

**Description:** The frontend interface shall be developed using Streamlit, providing a user-friendly platform for interacting with the chatbot.

**Inputs:**

- User inputs
- Data from the backend

**Outputs:**

- Display of chatbot responses
- Interactive elements

**Functional Requirements:**

1. The frontend shall provide a text input field for user queries.
2. The frontend shall display chatbot responses in an easy-to-read format.
3. The frontend shall include interactive elements for enhanced user engagement.
4. The frontend shall include a button to reset the chat.
5. The frontend shall handle user interactions and communicate with the backend server.

**Dependencies:**

- Proper configuration of the Streamlit application.
- Reliable communication with the backend server.

**Preconditions:**

- The frontend application is running and connected to the backend server.

**Postconditions:**

- Users can interact with the chatbot seamlessly.

**Exception Handling:**

- In case of frontend errors, the system shall log the error and notify the user with an appropriate message.

## 3.5 Conversational Capabilities

**Description:** The chatbot shall engage in meaningful conversations with users, providing accurate and contextually relevant responses.

**Inputs:**

- User queries

**Outputs:**

- Chatbot responses

**Functional Requirements:**

1. The chatbot shall understand and process natural language queries.
2. The chatbot shall maintain conversational context to handle multi-turn interactions.
3. The chatbot shall provide accurate and relevant responses based on the knowledge base.
4. The chatbot shall handle various types of queries, including product information, company details, and warranty information.

**Dependencies:**

- Availability and proper functioning of AI models.
- Availability of proper prompts to pass into the chains.
- Accurate and up-to-date knowledge base.

**Preconditions:**

- The LLM models are trained and integrated.
- The knowledge base is populated with relevant data.

**Postconditions:**

- The chatbot engages users in meaningful conversations and provides useful information.

**Exception Handling:**

- If the chatbot cannot process a query, it shall log the error and provide a fallback response indicating that it could not understand the request.

# A Glossary

- **AI (Artificial Intelligence):** The simulation of human intelligence in machines that are programmed to think like humans and mimic their actions.
- **LLM (Large Language Model):** a computational model notable for its ability to achieve general-purpose language generation and other natural language processing tasks such as text generation.
- **API (Application Programming Interface):** A set of protocols and tools for building software and applications.
- **FAST API:** A modern, fast (high-performance), web framework for building APIs with Python.
- **Streamlit:** An open-source app framework for creating and sharing beautiful, custom web apps for machine learning and data science.
- **NLP (Natural Language Processing):** A field of AI that gives machines the ability to read, understand, and derive meaning from human languages.
- **Vector Database:** A type of database designed to store and manage vectorized data, often used in machine learning and AI applications for tasks such as similarity search.

# B Data Schema

**cb_products Table Schema:**

| Column Name | Data Type | Description |
|---|---|---|
| name | Text | The name of the product |
| description | Text | A brief description of the product |
| url | Text | The URL to the product page |
| image_url | Text | The URL to the product image |
| category_names | Text | The names of categories the product belongs to |
| category_label | Text | The label of the product category |
| tag_names | Text | Tags associated with the product |

# C Installation and Configuration

## C.1 Prerequisites:

- Python installed on your machine.
- Necessary Python packages listed in `requirements.txt`.

## C.2 Installation:

1. Clone the repository.
2. Navigate to the project directory.
3. Install the required packages by running:

```
pip install -r requirements.txt
```

### C.3 Running the Application:

**Backend:** To start the backend server, execute:

```
uvicorn app.main:app --reload
```

**Frontend:** Ensure the backend server is running, then start the frontend by executing:

```
streamlit run frontend/app.py
```

**Updating the Database/URLs/Warranty Information:** To update the database, URLs, or warranty information, use the following command:

```
python Data/loader.py
```

The `loader.py` script accepts several arguments to specify paths and credentials. The default values for these arguments are:

| Argument | Flag | Default Value |
|---|---|---|
| –url | -u | Data/urls.txt |
| –pdf | -p | Data/Warranty.pdf |

## D Error Handling

**Data Collection Errors:**

- Log the error and continue with the next data source.
- Notify the user of any data sources that were not successfully processed.

**Backend Server Errors:**

- Log the error and return a meaningful error message to the user.
- Implement retry mechanisms for transient errors.

**AI Model Errors:**

- Log the error and provide a fallback response.
- Notify the user that the chatbot could not process the query.

**Frontend Errors:**

- Log the error and display an error message to the user.
- Ensure the frontend can gracefully handle communication failures with the backend.

This concludes the Software Requirements Specification for the Conversational Chatbot project. Each chapter provides detailed information to guide the development and ensure that all functional and non-functional requirements are met.