

END OF SEMESTER PROJECT

Computer Engineering
Module: Java programming

Development of an E-commerce Question-Answering Chatbot



Realized by :

- Hajar Bousaken
- Fatima-el zahra Flifla

Supervised by:

Pr. BAHRI ABDELKHALEK

Année Universitaire 2024/2025

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ أَنْتَ الْعَلِيمُ الْحَكِيمُ ﴿٣٢﴾
سورة البقرة

صدق الله العظيم

Dedication

First, we thank God Almighty for giving us this opportunity in life to be part of this class, this batch, and this school. Thank God. We also express our highest gratitude to our parents who have supported us and always support us with their prayers and their desire to see us succeed and reach a position we are proud of. Thank God for the blessing of having parents in our lives.

We are pleased to extend our sincere respect and gratitude to everyone who played a role, even if small, from the beginning of our educational journey in successfully completing this project.

We also extend our sincere thanks to our respected professor, Mr. Bahri Abdel Khaleq, for his tremendous efforts and his keenness to teach us useful knowledge and skills that contributed to the completion of a project that we are proud of. We also extend our thanks to all the professors of the Information Engineering Department for the continuous support they provide to us.

To our parents, brothers and sisters, our words are not enough to express our deep love and boundless gratitude to you. Your many sacrifices for our education represent to us the essence of perseverance, creativity and unlimited dedication.

الإهداء

أولاً، نشكر الله سبحانه وتعالى على منحنا هذه الفرصة في الحياة لنكون جزءاً من هذا الفصل وهذه الدفعة وهذه المدرسة. الحمد لله. كما نعبر عن أسمى عبارات الامتنان لوالدينا الذين دعمونا ويدعموننا دائماً بدعواتهم ورغبتهم في رؤيتنا ننجح ونصل إلى مكانة نفخر بها. الحمد لله على نعمة وجود الوالدين في حياتنا.

يسرنا أن نتقدم بخالص احترامنا وامتناننا لكل من كان له دور ولو صغير منذ بداية مشوارنا في حياة الدراسة في إتمام هذا المشروع بنجاح.

نتوجه أيضاً بالشكر الجزيل لأستاذنا المحترم أ.بحري عبد الخالق على جهوده الجبارة وحرصه على تلقيننا ما ينفعا من علم ومهارات ساهمت في إنجاز مشروع نفتخر به. كما نقدم شكرنا لجميع أساتذة شعبة الهندسة المعلوماتية على دعمهم المستمر الذي يقدمونه لنا.

إلى والدينا وإخواننا وأخواتنا، كلماتنا لا تكفي للتعبير عن حبنا العميق وامتناننا اللامحدود لكم. تضحياتكم العديدة من أجل تعليمنا تمثل لنا جوهر المثابرة والإبداع والتفاني اللامحدود.

Dédicace

Tout d'abord, nous remercions Dieu Tout-Puissant de nous avoir donné cette opportunité dans la vie de faire partie de cette classe, de cette promotion et de cette école. Dieu merci. Nous exprimons également notre plus grande gratitude à nos parents qui nous ont soutenus et nous soutiennent toujours par leurs prières et leur désir de nous voir réussir et atteindre une position dont nous sommes fiers. Remercions Dieu pour la bénédiction d'avoir des parents dans nos vies.

Nous sommes heureux d'exprimer notre sincère respect et notre gratitude à tous ceux qui ont joué un rôle, même minime, depuis le début de notre parcours éducatif dans la réussite de ce projet.

Nous adressons également nos sincères remerciements à notre professeur respecté, M. Bahri Abdel Khaleq, pour ses efforts considérables et son souci de nous enseigner des connaissances et des compétences utiles qui ont contribué à la réalisation d'un projet dont nous sommes fiers. Nous remercions également tous les professeurs du Département de génie de l'information pour le soutien continu qu'ils nous apportent.

À nos parents, frères et sœurs, nos paroles ne suffisent pas à vous exprimer notre profond amour et notre gratitude sans limites. Vos nombreux sacrifices pour notre éducation représentent pour nous l'essence de la persévérance, de la créativité et d'un dévouement sans limites.

Abstract

This report covers the development process of a question-and-answer chatbot designed for e-commerce platforms. Our project leverages cutting-edge technologies including Kafka for real-time message streaming, MongoDB for efficient data storage, NLP (natural language processing) to understand and generate responses, and JavaFX to provide an interactive user interface .

The system is designed to manage customer questions regarding products available in a store, store policies, store working hours, promotions that may exist and be enjoyed by customers, and recommendations on specific products desired by customers. customers to evolve the customer experience and attract more customers by understanding their desires.

By amalgamating advanced technologies with real-time streaming principles, this project aims to redefine the landscape of real-time chatbot development. Through continuous iteration and refinement, this initiative aspires to set new standards for efficiency and innovation in backend systems engineering.

Keywords: Chatbot, E-commerce, Question-answering, Support client, streaming, real-time.

Résumé

Ce rapport porte le processus de développement d'un chatbot de questions-réponses conçu pour les plateformes de e-commerce. Notre projet exploite des technologies de pointe, notamment Kafka pour le streaming de messages en temps réel, MongoDB pour un stockage de données efficace, le NLP (traitement du langage naturel) pour comprendre et générer des réponses, et JavaFX pour offrir une interface utilisateur interactive.

Le système est conçu pour gérer les questions des clients concernant les produits disponibles dans un magasin, les politiques du magasin, les horaires de travail du magasin, les promotions qui peuvent exister et bénéficiées par les clients et les recommandations sur des produits spécifiques désirés par les clients pour ainsi évoluer l'expérience client et attirer plus de clients en comprenant leur désir.

En amalgamant des technologies avancées avec des principes streaming en temps réel, ce projet vise à redéfinir le paysage du développement d'un chatbot en temps réel. À travers une itération et un affinage continus, cette initiative aspire à établir de nouvelles normes d'efficacité et d'innovation dans l'ingénierie des systèmes backend.

Mots clés : E-commerce, Chatbot, Questions-Réponses, Messagerie en Temps Réel, Support Client, Automatisation.

Table De Matières

Dedication.....	4
Dedicace.....	5
Abstract.....	6
Resume.....	7
Table of content.....	8
General introduction.....	9
Chapter1 :Project Description.....	10
Introduction.....	10
Project Presentation.....	10
Motivation And problems of project.....	10
Objectives of project.....	11
Specifications.....	11
Work Methodology.....	12
Methodological Approach used.....	13
Planning.....	14
QOOQCP method.....	15
Gantt chart.....	15
Conclusion.....	16
Chapter 2 : The state of the art.....	16
Introduction.....	16
History des Chatbots.....	17
The first Chatbots.....	17
Advances in Artificielle Intelligence.....	17
Modern Chatbots.....	17
The Role of Chatbots and NLP in E-commerce.....	18
Examples of NLP-based Chatbots in E-commerce.....	18
Conclusion.....	19
Chapter 3 : Methodology and Outils.....	19
Introduction.....	19
Methodology for developing our personalized chatbot.....	20
Requirements Analysis.....	20
System Architecture Design.....	20
Development Workflow.....	21
Collection, preparation of data and generation of responses.....	22
NLP Methodology Using Stanford NLP.....	22
NLP Methodology Using Keyword Extraction and Question Normalization.....	23
Kafka.....	24
MongoDB.....	29
JavaFX.....	31
IntelliJ.....	32
GitHub.....	35
Conclusion.....	36
Chapter 4 : Results and Analysis.....	37
Introduction.....	37
Start the Application Environment.....	38
II. Test the Application Workflow.....	39
III. Test End-to-End Functionality.....	40
Conclusion.....	49
Conclusion and Perspectives.....	50
References.....	52

General Introduction

In today's fast-paced digital world, e-commerce platforms have become an essential part of the global economy, providing customers with the convenience of shopping anytime and anywhere. As the number of users and their demands grow, customer service has become a critical factor in maintaining customer satisfaction and loyalty. To address this need, intelligent solutions such as chatbots are being increasingly adopted to provide instant and accurate responses to user queries.

This project focuses on the development of a question-answering chatbot tailored for e-commerce platforms. The chatbot is designed to handle a wide range of customer inquiries, including product availability, store policies, promotions, and personalized recommendations. By leveraging advanced technologies such as Kafka, Natural Language Processing (NLP), MongoDB, JavaFX, and JSON, the chatbot ensures real-time interaction, efficient data management, and an intuitive user experience.

The primary objective of this project is to build a robust and scalable system that enhances customer engagement while reducing the workload on human support teams. The integration of Kafka enables seamless communication between system components, while MongoDB provides flexible and efficient storage of customer conversations and data. NLP techniques are employed to comprehend and generate meaningful responses, ensuring the chatbot can understand user intentions and respond accurately.

This report provides a detailed analysis of the project, including the technologies used, the architecture of the system, and the functionalities of the chatbot. It highlights the importance of each technology and its role in creating an intelligent and responsive system. Furthermore, the report demonstrates how this solution can transform customer support in the e-commerce sector, delivering a fast, reliable, and user-centric experience.

Chapter 1 : Project Description

Introduction:

This chapter outlines the general context of the project. The first section is dedicated to the presentation of the project, and then demonstrates the Methodological **Approach and QOOQCP Method** that we are working with, offering a visual timeline of tasks and milestones. The chapter concludes by summarizing the main points and setting the stage for subsequent chapters.

I. Project Presentation :

1.Motivation And problems of project

In today's rapidly evolving e-commerce landscape, providing seamless and efficient customer support is critical to ensuring customer satisfaction and improving business outcomes. Traditional methods of customer service often struggle to keep up with the growing volume of customer inquiries. As a result, there is a strong demand for automated systems that can handle customer queries in real-time, offering quick and accurate responses.

Our project addresses this challenge by developing an intelligent question-answering chatbot for e-commerce platforms. The chatbot uses Natural Language Processing (NLP) to understand and respond to customer questions, Kafka for real-time message streaming, and MongoDB for storing data. This system aims to provide an efficient solution to handle frequent customer inquiries, such as product details, store policies, promotions, and more.

The problem we aim to solve is the gap between customer expectations for fast, automated support and the limitations of traditional customer service systems. By implementing an AI-powered chatbot, we intend to create a scalable and user-friendly solution to enhance the customer experience.

2. Objectives

The main objective of this project is to develop an intelligent question-answering chatbot for e-commerce platforms, designed to enhance the customer experience by providing real-time, automated responses. The chatbot will integrate advanced technologies such as Kafka for real-time message streaming, MongoDB for efficient data storage, NLP (Natural Language Processing) for understanding and generating responses, and JavaFX for an interactive user interface. Key objectives include:

1. **Real-time Interaction:** Enable customers to interact with the chatbot seamlessly and receive accurate answers to their queries.
2. **Data Management:** Implement MongoDB to efficiently store product information, customer inquiries, and chatbot responses.
3. **Natural Language Processing:** Utilize NLP techniques to understand customer questions and generate appropriate responses.
4. **User Interface:** Create a user-friendly, engaging interface using JavaFX, making the chatbot accessible and easy to use.
5. **Scalability and Efficiency:** Ensure the chatbot system can handle a large number of queries simultaneously, offering a smooth user experience even under heavy load.
6. **Integration with E-commerce Features:** Enable the chatbot to access product details, store policies, and recommendations, enhancing its ability to assist customers.

3. Specifications

The chatbot project is designed to provide real-time, accurate responses to user queries in the context of an online fashion store. Below are the key specifications:

1. Functionality

- Answer frequently asked questions (FAQs) about stock availability, return policies, store hours, and promotions.
- Provide product recommendations based on user preferences and past interactions.
- Support multiple languages (e.g., English, French, Arabic) for diverse user bases.
- Handle real-time customer queries using Natural Language Processing (NLP).
- Allow retrieval of conversation history based on user ID or email.

2. Technical Features

- Backend Framework: Java (with integration of Kafka for messaging).
- Natural Language Processing: Stanford CoreNLP for tokenization, lemmatization, and intent detection.
- Database: Mongo database for storing product information.
- Integration: JSON-based communication for seamless integration between components.
- Message Broker: Apache Kafka for producer-consumer communication between chatbot modules.

3. User Interface

- Simple and intuitive web-based chat interface.
- JavaFX-based desktop client for managing the chatbot's responses and interactions.

4. Performance Requirements

- Low latency for real-time responses (< 2 seconds).
- Scalability to handle concurrent users during peak times.

5. Security Features

- Secure data storage for user queries and personal information.
- Encrypted communication between client and server to protect user privacy.

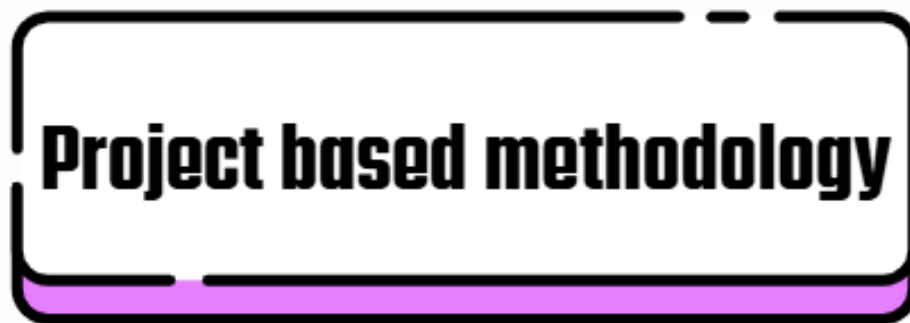
6. Additional Features

- Support for learning from new questions by storing and analyzing them for future responses.
- Gantt Chart planning to ensure clear project timeline and task allocation.

4. Work Methodology

1. Methodological Approach used

We adopted a **Project-Based Methodology**. This approach focuses on achieving a specific goal by dividing the project into clear, collaborative tasks based on individual expertise. It allowed us to ensure structured progress while maintaining flexibility to adapt to challenges encountered during development.



❖ Requirement Analysis and Task Distribution:

- We began by defining the core objectives of the project: building a responsive and efficient chatbot leveraging Kafka, NLP, MongoDB, and JavaFX.
- Tasks were divided according to each team member's strengths:
 - **Hajar's Role:** Focused on integrating Kafka into the project to ensure seamless message flow and running the system without manual intervention (e.g., Zookeeper and Kafka running automatically). Additionally, worked on an NLP module utilizing Stanford CoreNLP.
 - **Fatima zahra's Role:** Worked on database management, converting CSV files to JSON format for MongoDB integration. Also developed an alternative NLP module based on "Keyword Extraction and Question Normalization" and designed the GUI with JavaFX.

❖ **Design and Development:**

- Each component was designed separately and developed using a modular approach to ensure smooth integration.
- Kafka was configured to automate its startup, eliminating the need to manually start Zookeeper and Kafka servers.
- The NLP module was first implemented using Stanford CoreNLP but replaced with a faster, keyword-based approach to improve response times.
- Fatima created a functional GUI using JavaFX and connected it to the database for real-time interaction.

❖ **Integration and Testing:**

- Once individual components were developed, they were integrated into a single system.
- Testing focused on ensuring data consistency between MongoDB and the GUI, validating NLP response accuracy, and confirming Kafka's ability to handle messaging efficiently.

❖ **Iteration and Optimization:**

- We iteratively improved the system, such as optimizing the NLP response time and ensuring the user interface met project expectations.

❖ **Final Deployment:**

- The chatbot was finalized with all components fully integrated and tested.

I. Planification :

1. Methodology: QQQQCP Approach

For our chatbot project, we applied the **QQQQCP method** to clearly define all aspects of our work. This method, which stands for *Who, What, Where, When, Why, How*, allowed us to organize our project efficiently and ensure every aspect was addressed systematically.

Question	Réponse
What	To develop an efficient, real-time question-answering chatbot using Kafka, NLP, MongoDB, JSON, and JavaFX, capable of enhancing user satisfaction in an e-commerce setting.
Who	Team Members: Hajar Bousaken & Fatima zahra flifla Target Audience: E-commerce users seeking quick answers to FAQs and real-time assistance.
Where	Ensah
When	Début : 19/10/2024 Fin : 19/12/2024
Why	★ To automate and improve the customer service experience for e-commerce users by providing instant responses to their queries. ★ To reduce the workload on human customer service agents and improve response times for users.
How	Project-Based Methodology: ★ Kafka is used for message brokering, ensuring smooth communication between different components of the chatbot system. ★ MongoDB serves as the database for storing and retrieving information. ★ NLP processes user queries, extracting keywords and normalizing questions for accurate responses. ★ JSON is used as an intermediary format for data exchange between components. ★ JavaFX provides a user-friendly GUI for chatbot interactions.

Conclusion:

Thus, after presenting the problem, contextualizing our project, and defining our objectives, we adopted a rigorous methodology throughout our work. The presentation of the Gantt chart provided a clear visualization of the planned steps. These tools not only facilitated project management but also contributed to achieving the expected results within the allocated time frame.

Chapter 2: State of the Art

Introduction:

In this chapter, we will explore the evolution of chatbots through history, their recent transformation thanks to transformer-based technologies. Chatbots, which started as simple rule-based systems, have undergone a significant metamorphosis with the advent of transformers, enabling a more nuanced understanding and context-aware responses to users.

I. History of Chatbots

Chatbots, or conversational agents, have a rich and fascinating history spanning decades. Their development has been marked by significant advances in Artificial Intelligence (AI) and Natural Language Processing (NLP), leading to increasingly sophisticated and useful applications

Early Chatbots:

1. 1966: ELIZA

- Creator: Joseph Weizenbaum at MIT
- Description: Often considered the first chatbot, ELIZA used simple scripts to simulate a conversation with a psychotherapist. Despite its basic rule-based design,

ELIZA demonstrated that computers could simulate human

- conversation at a rudimentary level.

2. 1972: PARRY

- Creator: Kenneth Colby at Stanford University
- Description: PARRY was an advancement over ELIZA, designed to simulate a person with paranoid schizophrenia. It used more complex heuristic-based models to generate responses.

Advances in Artificial Intelligence:

1. 1980s: Racter

- Description: A text-generating program capable of producing random text. While not strictly a chatbot, it showcased AI's potential for generating coherent text.

2. 1990s: A.L.I.C.E. (Artificial Linguistic Internet Computer Entity)

- Creator: Richard Wallace
- Description: A.L.I.C.E. used rule-based and pattern-matching approaches to engage in conversations, winning multiple Loebner Prizes for AI advancements.

Modern Chatbots:

1. 2000s: SmarterChild

- Creator: ActiveBuddy
- Description: An interactive chatbot available on instant messaging platforms like AIM and MSN, providing simple responses and information such as weather and sports updates.

2. 2010s:

- Siri (2011): Integrated into Apple smartphones, Siri used advanced voice recognition and NLP technologies.
- Alexa (2014): Amazon's voice assistant for Echo devices, which popularized home assistant usage.
- Google Assistant (2016): A highly contextual virtual assistant integrated into

II. The Role of Chatbots and NLP in E-commerce

In the context of e-commerce, chatbots leveraging Natural Language Processing (NLP) play a pivotal role in enhancing customer experiences. Unlike systems powered by Large Language Models (LLMs), NLP-based chatbots are designed to efficiently handle specific tasks with structured data, offering reliable and quick solutions for user needs.

Key Contributions of NLP-based Chatbots in E-commerce:

1. Real-time Assistance:

- Providing instant responses to user inquiries about product availability, return policies, promotions, and store details.

2. Keyword-based Recommendations:

- Extracting keywords from user queries to offer targeted product suggestions.

3. Multilingual Support:

- Facilitating interactions in multiple languages to accommodate a global and diverse customer base.

4. System Efficiency:

- Reducing reliance on expensive computing resources compared to LLMs, with faster response times tailored to specific domains.

III. Examples of NLP-based Chatbots in E-commerce:

→ H&M Chatbot:

- ◆ Guides customers through product searches using keywords and categories, ensuring a smooth shopping experience.

→ Zalando's Chatbot:

- ◆ Assists users in finding outfits based on preferences, leveraging keyword extraction to refine recommendations.

→ Sephora's Virtual Assistant:

- ◆ Offers makeup tutorials, product availability, and recommendations based on user questions.

Conclusion

In conclusion, the integration of chatbots powered by NLP technologies has become indispensable in modern e-commerce. These chatbots offer tailored and efficient solutions to address customer needs, from real-time assistance to personalized recommendations. By leveraging multilingual capabilities and focusing on domain-specific tasks, NLP-based chatbots enhance user satisfaction while reducing operational costs. Unlike LLMs, which require significant computational resources, NLP systems are optimized for targeted interactions, making them an ideal choice for e-commerce platforms seeking to improve their customer experience efficiently.

Chapitre 3 : Méthodologie et Outils

Introduction:

In this chapter, we will undertake a detailed exploration of each critical step necessary to develop an effective chatbot. From initial planning to the final deployment, every phase of the development process will be examined in depth. This will provide a comprehensive understanding of the methodologies and tools used to create a robust chatbot, emphasizing their role in ensuring the chatbot's functionality, performance, and integration within the larger system. Through this analysis, we aim to highlight the key considerations and best practices that guided our project, allowing for a clearer insight into the workflow and decision-making processes involved in building a successful chatbot.

I. A Detailed Methodology for Developing Our Custom Chatbot

1. Requirements Analysis

To develop an effective chatbot, the key functionalities include: question-answering to provide accurate information, product recommendations based on user preferences, order tracking for real-time updates, and customer issue resolution for quick support.

Defining user personas and interaction scenarios helps tailor the chatbot to diverse needs, such as providing instant answers or recommending relevant products. Finally, the performance metrics focus on ensuring fast response times, high accuracy of answers, and system reliability, delivering an optimal user experience.

2. System Architecture Design

- ❖ **NLP module:** the chatbot project uses a modular approach for efficient development and functionality. The NLP module processes user inputs by performing tasks such as tokenization, intent detection, and response generation. Tools like Stanford NLP or Apache OpenNLP are used to ensure the chatbot understands user queries and provides relevant responses.
- ❖ **Kafka messaging module:** the Kafka messaging module facilitates real-time communication between different components. By leveraging Kafka Producer and Consumer APIs, the system handles multiple requests concurrently, ensuring fast and scalable processing for a seamless user experience.

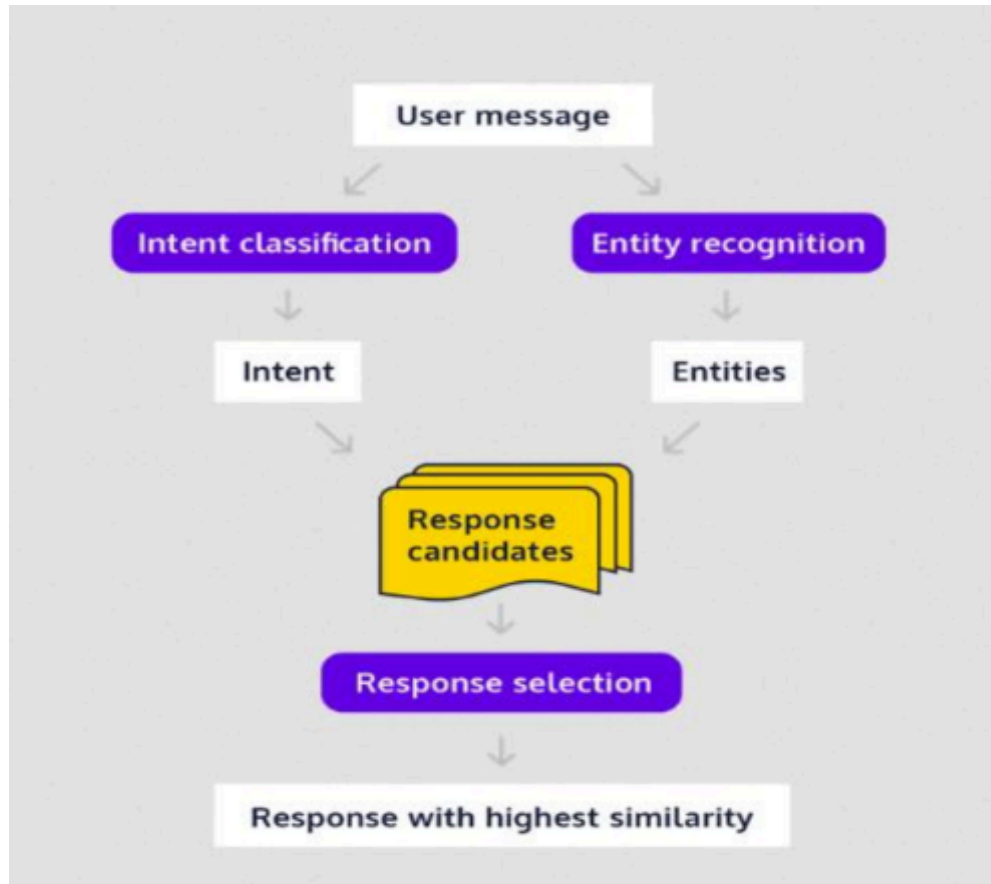
- ❖ **Database layer:** the database layer, built with MongoDB, stores essential data such as product information, user history, and FAQs. Its flexible document-based structure allows for quick data retrieval and efficient storage, making it well-suited for the chatbot's dynamic requirements.
- ❖ **User interface:** the user interface, developed using JavaFX, delivers an interactive and intuitive experience for users. It enables clear communication, displays responses and options, and provides real-time updates for a smooth interaction process.

3.Development Workflow:

The development of the chatbot project will follow an Agile methodology, organized into iterative cycles. Sprint 1 ; will focus on creating the foundational chatbot framework and setting up the database.

Sprint 2; will integrate Natural Language Processing (NLP) for generating responses. Sprint 3 will introduce real-time messaging using Kafka for effective communication. Sprint 4 will shift focus to designing and implementing the user interface with JavaFX.

Additionally, regular testing and user feedback will be incorporated throughout the process to ensure continuous improvement and alignment with user needs.



II. Collection, preparation of data and generation of responses

In our project, we employed two distinct NLP methodologies to develop a robust chatbot for answering questions in English. Below is a detailed description of each approach:

1. NLP Methodology Using Stanford NLP

For the NLP component of the project, **Stanford CoreNLP** is a powerful suite of natural language processing tools that provides various functions for text analysis. The primary goal was to

process text, tokenize it, and extract relevant features for question answering. The methodology followed involved the following steps:

- **Preprocessing:**
 - User question was first cleaned and preprocessed. This included removing noise (such as unnecessary symbols or punctuation) and normalizing the text.
 - Tokenization was applied to split the text into smaller units (tokens), which are essential for analyzing and understanding the text.
- **Question Classification and Answer Retrieval:**
 - The system classified user questions based on predefined categories, allowing the chatbot to identify the type of response required (e.g., product-related, order status, etc.).
 - Finally, using keyword extraction and semantic analysis, the system matched the questions to the most relevant answers from the database, enhancing the chatbot's ability to provide contextually appropriate responses.
- For each intent, the method compares the user input with the patterns using **cosine similarity** to measure the closeness between the input and each pattern.
 - **Cosine similarity** is used to quantify the similarity between two vectors (representing the tokenized user input and the tokenized pattern). The higher the similarity score, the more relevant the pattern is to the user input.
- The response with the highest similarity score is selected, and a random response is chosen from the list of possible responses in the json file for that intent.

➤ **Challenges:**

- One of the main challenges we encountered was the processing time for generating responses, which often took around 40 seconds. This was primarily due to the large amount of data we have in our json file and load them to processing with Stanford CoreNLP.

2. NLP Methodology Using Keyword Extraction and Question Normalization

The NLP methodology in the chatbot project focuses on analyzing user questions to provide accurate responses based on product information stored in the MongoDB database.

★ Initially, when a user asks a question, the chatbot attempts to directly match the query with an existing product by identifying specific keywords related to the product in the database. If a match is found, the chatbot quickly retrieves and presents the relevant product details. However, if the question is more general or the exact product is not found

★ The chatbot proceeds to the next step: keyword extraction and question normalization. This involves breaking down the user's query into key components, such as product categories or attributes, and using NLP techniques to extract the relevant keywords. The chatbot then compares these keywords to a predefined set of product descriptions or responses stored in its resources, calculating the similarity between the question and available answers. By evaluating which response contains the most matching keywords, the chatbot selects and provides the best possible answer, ensuring that even imprecise or unclear queries are handled effectively. This methodology enables the chatbot to deliver relevant responses regardless of the user's phrasing, improving the overall user experience.

III. Kafka



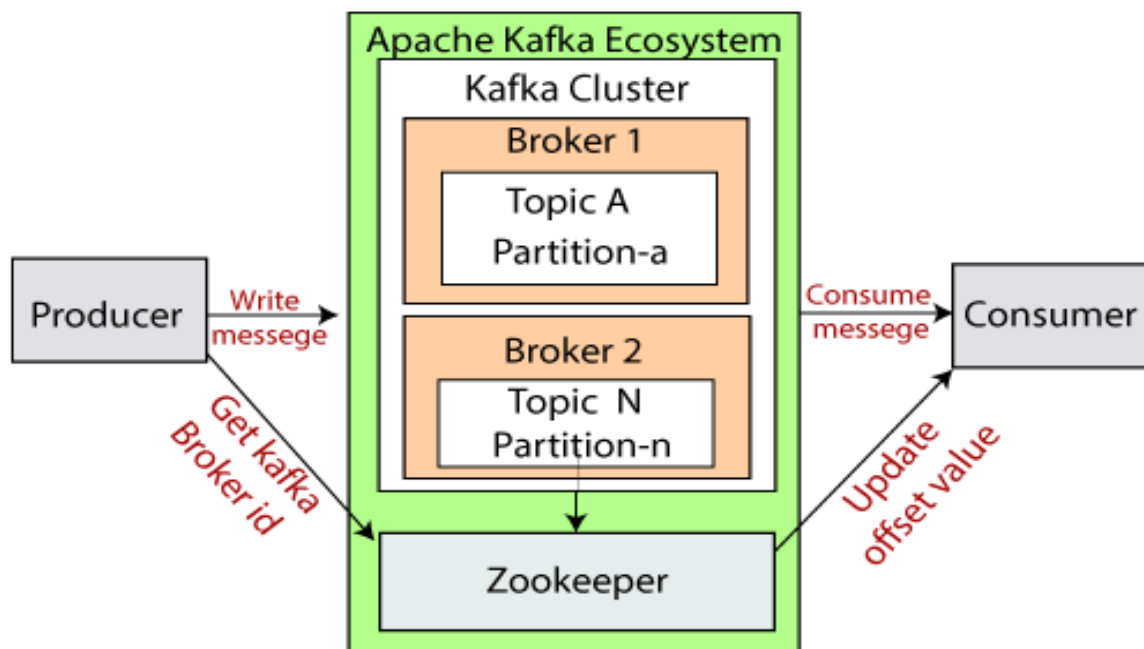
1. Definition:

Apache Kafka is an open-source distributed event streaming platform used to handle real-time data feeds. It is widely used for building real-time data pipelines and streaming applications. Kafka is capable of handling large volumes of data with high throughput, scalability, fault tolerance, and low latency, making it ideal for managing the flow of messages in real-time applications.

Kafka's architecture is based on the following key components:

- ❖ **Producers:** Producers are responsible for publishing messages (events) to Kafka topics. They push data into Kafka clusters.
- ❖ **Topics:** Kafka topics are the channels through which data flows. Topics are partitioned for scalability, with each partition storing a subset of the messages.
- ❖ **Partitions:** Each topic is divided into partitions for parallel processing. Partitions allow distributed storage and load balancing across Kafka brokers.
- ❖ **Brokers:** Brokers are Kafka servers that store data and serve client requests. A Kafka cluster consists of multiple brokers, ensuring redundancy and scalability.
- ❖ **Consumers:** Consumers read messages from topics. They can subscribe to specific topics and process data either independently or as part of a consumer group.
- ❖ **Consumer Groups:** A consumer group allows multiple consumers to read from the same topic in parallel. Each consumer in the group processes data from one partition, enabling efficient workload distribution.

- ❖ **ZooKeeper/Controller:** Originally, ZooKeeper managed Kafka's metadata (such as leader election for partitions). However, newer versions use Kafka's own KRaft protocol for metadata management, reducing dependency on ZooKeeper.
- ❖ **Message Storage:** Kafka persists messages on disk and allows consumers to replay messages from a specific offset, enabling reliable delivery and fault tolerance.



Apache Kafka Architecture

Kafka plays a critical role in ensuring smooth, real-time communication in the chatbot system. Here's how it helps:

1. Real-Time Messaging

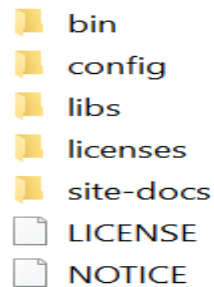
- Kafka ensures low-latency communication between the chatbot's producer (user queries) and consumer (chatbot responses), enabling near-instantaneous interactions.
- 2. **Scalability**
 - Its distributed architecture allows horizontal scaling, which is essential for handling multiple concurrent user interactions during peak times.
- 3. **Message Persistence**
 - Kafka stores messages durably, enabling the chatbot to process user queries reliably even in cases of system downtime or failure.
- 4. **Asynchronous Processing**
 - The producer and consumer are decoupled, meaning the chatbot can process messages asynchronously without blocking operations, ensuring a smooth user experience.
- 5. **Fault Tolerance**
 - Kafka replicates data across brokers, ensuring fault tolerance and availability. This is crucial for maintaining consistent communication in the chatbot system.
- 6. **Flexible Integration**
 - Kafka easily integrates with other components like NLP modules, databases, and user interfaces, streamlining the flow of data through the system.
- 7. **Load Balancing**
 - With consumer groups and topic partitioning, Kafka distributes the processing workload effectively, preventing bottlenecks in the chatbot's message handling.

2. Installation steps and Problems :

After we install kafka zip , we must unzip and then do those configuration to

not have problems after :

1. Go to kafka folder :



Those are the most important folder in kafka , now we will enter in config then in server and run it with notepad to change to the right configuration

We have to add path for folder which will save the logs of kafka , here I'am choosing this :

```
>>>>log.dirs=c:/kafka/kafka-logs
```

Then we have to fix the port for kafka :

```
listeners=PLAINTEXT://localhost:9092
```

```
advertised.listeners=PLAINTEXT://localhost:9092
```

Then go to zookeeper script and change the path : **dataDir=C:/kafka/zookeeper-data**

After we solve the configuration of our files , we can now run zookeeper and kafka without problems with those commands :

- `.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties`
- `.\bin\windows\kafka-server-start.bat .\config\server.properties`
- `kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partition 1 --topic test`
- `kafka-console-producer.bat --broker-list localhost:9092 --topic test`
- `kafka-console-consumer.bat --topic test --bootstrap-server localhost:9092 --from-beginning`

Now if we don't delete the folders below, we will have those problems :

```
at kafka.Kafka$.main(Kafka.scala:112)
at kafka.Kafka.main(Kafka.scala)
[2024-12-30 08:59:01,946] INFO [KafkaServer id=0] shutting down (kafka.server.KafkaServer)
C:\Users\PC\IdeaProjects\ChatBot-using-Kafka-and-Java>
```

>>>>Stopping impressively in Kafka, the questions and responses will not be received by the Kafka cluster.

Solution:

To not have problems with deleting folders creating by running kafka , kafka-logs and zookeeper-data , we have to stop them after closing our application with:

- `.\bin\windows\zookeeper-server-stop.bat .\config\zookeeper.properties`
- `.\bin\windows\kafka-server-stop.bat .\config\server.properties`

Benefits of Kafka:

- ❖ **Stockage et reprise** : Kafka stocke les messages durablement, permettant la reprise en cas de panne.
- ❖ **Faible latence** : Kafka est optimisé pour des opérations à haute vitesse avec des délais minimes

IV. MongoDB



MongoDB is an open-source, NoSQL (Not Only SQL) database management system that is designed to handle large volumes of unstructured or semi-structured data. Unlike traditional relational databases, which use tables and rows to organize data, MongoDB stores data in flexible, JSON-like documents called BSON (Binary JSON). This document-oriented approach allows MongoDB to scale easily and handle complex, high-volume data with varying structures.

MongoDB plays a crucial role in the development of our personalized chatbot, especially in handling large volumes of data and ensuring flexibility in storing and retrieving information. Here's why MongoDB is important for our project:

Flexible Schema Design:

MongoDB's document-oriented data model allows us to store data in a flexible format using JSON-like documents (BSON). This is particularly useful for a chatbot that handles diverse user queries and responses, which may not fit neatly into a traditional relational schema. The schema-less structure allows for easy modifications as the project evolves.

Scalability:

MongoDB is designed to scale horizontally, meaning that it can handle increasing data volumes without compromising performance. As the chatbot system grows, whether in terms of users or data complexity, MongoDB can scale effortlessly to accommodate these changes, ensuring consistent performance.

High Performance:

MongoDB's high performance makes it an ideal choice for our chatbot project. Its support for various indexing types significantly enhances query efficiency, enabling faster retrieval of user data, contextual history, and real-time suggestions. The robust aggregation framework facilitates efficient data analysis and transformation, which are essential for refining the chatbot's responses and improving its overall intelligence. These features ensure the chatbot can deliver a responsive and intelligent user experience.

In addition, MongoDB's rich ecosystem ensures seamless integration with multiple programming languages, allowing smooth backend operations. Its real-time analytics capabilities enable monitoring of chatbot interactions and the derivation of actionable insights to enhance functionality. By leveraging MongoDB's flexibility, scalability, and performance, the chatbot is equipped to handle complex user interactions and adapt to future growth, solidifying its role as a cornerstone of our infrastructure.

V. JavaFx:



JavaFX joue un rôle clé dans la création de l'interface utilisateur graphique (GUI) de notre chatbot, offrant un design intuitif, dynamique et engageant. Voici les principales fonctionnalités exploitées :

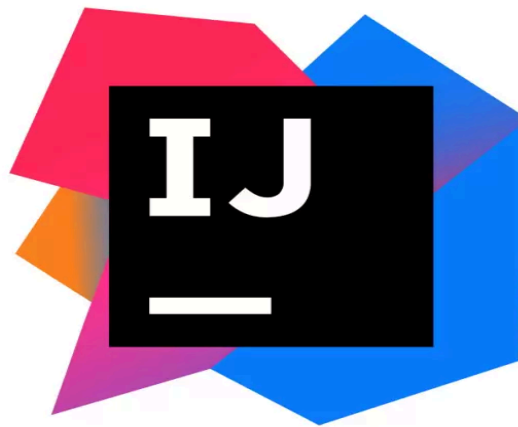
- **Composants Dynamiques**
 - JavaFX permet l'intégration dynamique des données issues de MongoDB.
 - Les cartes affichant les détails des produits sont mises à jour en temps réel.
 - L'interface reste fluide et réactive, même lors des changements de données.
- **Mise en page Conversationnelle**
 - Les messages utilisateur et bot sont présentés sous forme de bulles de discussion.
 - L'organisation conversationnelle rend l'interaction intuitive.
 - Les styles CSS personnalisés assurent une présentation claire et cohérente.
- **Animations Fluides**
 - Les transitions douces ajoutent une touche d'interactivité.
 - Les bulles glissent à l'écran et les cartes s'élargissent pour un effet captivant.
- **Design Modulable et Évolutif**
 - JavaFX offre une grande flexibilité pour adapter l'interface aux besoins.
 - De nouveaux composants peuvent être ajoutés sans modifier l'existant.

JavaFX supports a high degree of customization, allowing the chatbot's interface to be tailored to specific design requirements or user preferences. Through CSS styling and the use of Scene Builder, the application can easily adopt new themes or branding. Moreover, the framework's modular design makes it simple to scale the interface as new features or components are

introduced.

By leveraging JavaFX, the chatbot's interface is both functional and visually compelling. Its dynamic capabilities, combined with smooth animations and customizable design, create an application that not only meets the current needs of users but is also prepared to evolve with future demands.

VI. IntelliJ :



IntelliJ IDEA est l'environnement de développement intégré (IDE) choisi pour le développement du chatbot. Voici les principales fonctionnalités et avantages de cet IDE pour notre projet :

- **Productivité améliorée :**
 - IntelliJ offre un support complet pour Java et JavaFX, ce qui simplifie le processus de développement.
 - Les fonctionnalités comme l'auto-complétion du code, la navigation rapide et les suggestions intelligentes accélèrent le développement.
- **Débogage et Tests :**
 - L'outil de débogage d'IntelliJ permet de détecter rapidement les erreurs dans le code.
 - IntelliJ prend en charge les tests unitaires et les tests d'intégration, assurant une

validation continue du projet.

- **Intégration avec Git :**

- IntelliJ facilite la gestion de version en intégrant Git directement dans l'IDE.
- Les commits, branches et fusions peuvent être réalisés directement depuis l'interface.

- **Personnalisation de l'IDE :**

- IntelliJ permet de personnaliser l'interface selon les préférences du développeur.
- Des plugins et extensions sont disponibles pour ajouter de nouvelles fonctionnalités adaptées aux besoins du projet.

Dependencies Used in the Chatbot Project (maven)

In the development of the chatbot project for a retail environment, we used a set of key dependencies to build a robust, efficient, and scalable application. Below are the main dependencies we relied on, along with examples:

- **Java:**

Java serves as the primary programming language for the backend of the chatbot, providing a stable environment for logic, processing, and integrations.

- Example Dependency:

- `com.fasterxml.jackson.core:jackson-databind`: Used for JSON processing to handle data exchange between the chatbot and other systems.

- **Kafka:**

Apache Kafka is used for real-time data streaming and message brokering. It allows for decoupling the components of the system, ensuring efficient real-time message processing between the user interface, database, and NLP systems. Kafka is essential for handling a large number of concurrent user interactions.

- Example Dependency:

- `org.apache.kafka:kafka-clients`: The core Kafka client library used to produce and consume messages in real-time.

- **JavaFX:**

JavaFX is employed for the graphical user interface (GUI) of the chatbot, allowing for dynamic and interactive elements. JavaFX provides the framework to build engaging chat layouts, including real-time data updates and smooth animations.

- Example Dependency:

- **org.openjfx:javafx-controls**: This library provides the essential controls and layout containers for building the chatbot's user interface.

- **MongoDB:**

MongoDB is the NoSQL database used to store and manage the data for the chatbot, such as product details, user preferences, and conversation logs. MongoDB's flexibility in handling unstructured data allows for easy updates and scalability.

- Example Dependency:

- **org.mongodb:mongo-java-driver**: The MongoDB Java driver used to interact with the database, perform CRUD operations, and manage the data.

- **Natural Language Processing (NLP):**

NLP libraries are essential for processing and understanding user input. These libraries enable the chatbot to analyze and interpret natural language text, extract meaningful information, and generate appropriate responses.

- Example Dependency:

- **edu.stanford.nlp:stanford-corenlp**: A popular NLP library for tokenization, part-of-speech tagging, and named entity recognition, which helps the chatbot understand and process user queries effectively.

Maven and IntelliJ IDEA are essential tools used in the development of the chatbot project. **Maven** is a build automation tool that manages project dependencies, builds, and packaging. It simplifies the process of integrating libraries and ensures that the correct versions of dependencies are used. **IntelliJ IDEA** is the integrated development environment (IDE) chosen for the project, providing a powerful interface for coding, debugging, and testing. With features like code

completion, version control integration, and seamless Maven support, IntelliJ IDEA enhances productivity and streamlines the development workflow.

VII. GitHub



GitHub integration with **IntelliJ IDEA** streamlined the version control and collaboration process for the chatbot project. IntelliJ IDEA allows developers to manage Git repositories directly within the IDE, making it easier to work with GitHub features such as cloning, committing, pushing, and merging code.

Key features include:

- **Version Control:** Developers could commit, push, and pull changes directly from IntelliJ IDEA, making it easy to track and manage code changes.
- **Branching and Merging:** Developers created and managed branches for different tasks, ensuring parallel development without interfering with each other.
- **Pull Requests:** GitHub pull requests could be reviewed and merged within IntelliJ, simplifying code reviews and integration.
- **Task Tracking:** GitHub issues could be viewed and managed in IntelliJ, helping track bugs and tasks.

In summary, GitHub's integration with IntelliJ IDEA simplified the development workflow, improved collaboration, and ensured smooth version control, ultimately enhancing the project's

efficiency and success.

Conclusion:

In conclusion, the development of the personalized chatbot relies on an Agile methodological approach, allowing for flexible and iterative project management. The integration of advanced technologies such as Kafka for real-time streaming, JavaFX for the user interface, MongoDB for data management, and Natural Language Processing (NLP) for understanding user queries has enabled the creation of a system that is both powerful and responsive.

The use of Maven and IntelliJ was essential for efficiently managing dependencies and automating the build process, thus simplifying project management and allowing for a greater focus on feature development. Each technology, from Kafka to MongoDB, through JavaFX and NLP, contributes to building a robust chatbot capable of responding to user needs in a smooth and relevant manner.

By applying an Agile methodology, we also ensured continuous product evolution through regular user feedback and frequent testing. This approach guarantees not only rapid and adaptive development but also a final product that meets user expectations while remaining scalable for future improvements.

In summary, the personalized chatbot project demonstrates the effectiveness of a well-designed modular architecture, where each component, from real-time message processing to the user interface, works harmoniously to provide an optimal user experience and a reliable tool for customer interactions.

Chapter 4: Results and Analysis

Introduction

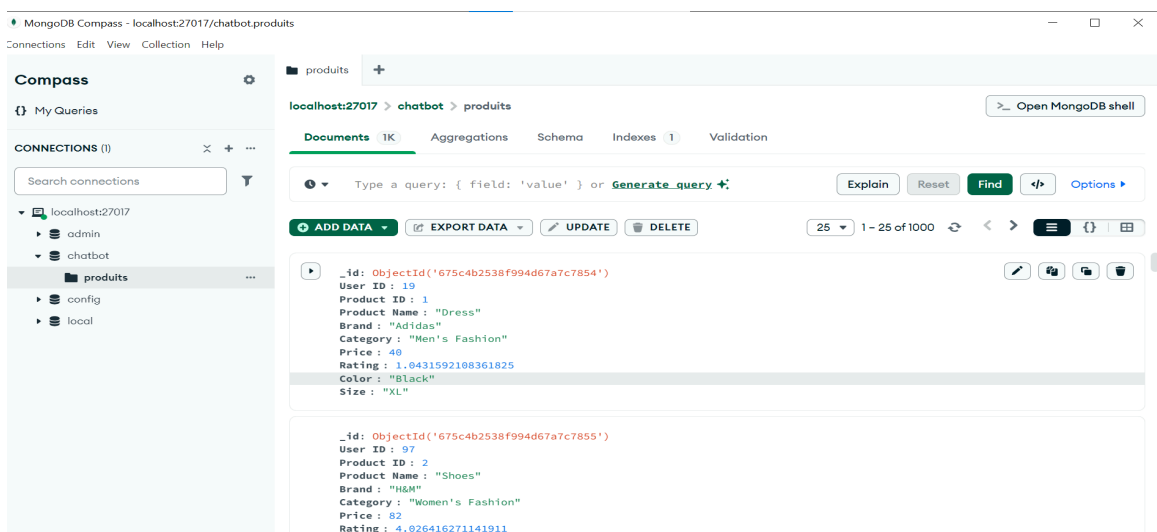
The introduction to this chapter outlines the methods used to evaluate the chatbot's responses, including testing scenarios that simulate real user interactions. We will discuss the criteria for measuring success, such as response accuracy, user satisfaction, and the system's ability to handle different types of queries. Additionally, we will highlight the importance of continuous improvement in natural language processing and how the insights gained from testing can be used to refine and optimize the chatbot for better performance in future use cases.

I. Start the Application Environment

1.1 Start MongoDB

- Open the terminal or command prompt.
- Start the MongoDB server by running the following command

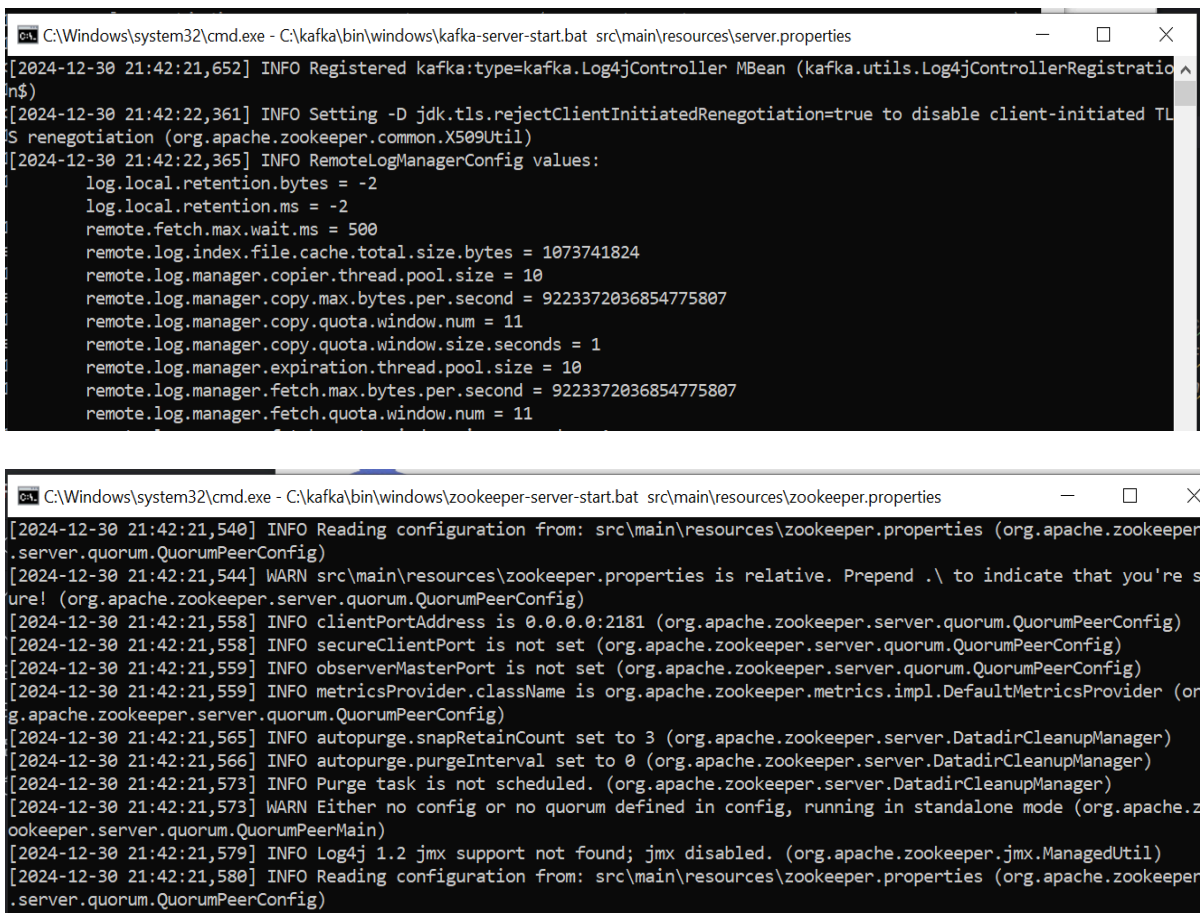
```
>>> mongod --dbpath
```



Here we have our data storage after converting our CSV file of product information into a JSON format compatible with MongoDB, the data is stored in a structured manner within a collection. When we query the collection, MongoDB will display the stored documents in a JSON-like format.

1.2 Start Kafka and Zookeeper

When we run our Application , the kafka and zookeeper start automatically and give :



The image contains two screenshots of Windows command prompt windows. The top window shows the Kafka server startup logs, and the bottom window shows the Zookeeper server startup logs. Both windows have a title bar indicating the command being executed: 'C:\Windows\system32\cmd.exe - C:\kafka\bin\windows\kafka-server-start.bat src\main\resources\server.properties' for the top and 'C:\Windows\system32\cmd.exe - C:\kafka\bin\windows\zookeeper-server-start.bat src\main\resources\zookeeper.properties' for the bottom.

```
C:\Windows\system32\cmd.exe - C:\kafka\bin\windows\kafka-server-start.bat src\main\resources\server.properties
[2024-12-30 21:42:21,652] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$1)
[2024-12-30 21:42:22,361] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-12-30 21:42:22,365] INFO RemoteLogManagerConfig values:
    log.local.retention.bytes = -2
    log.local.retention.ms = -2
    remote.fetch.max.wait.ms = 500
    remote.log.index.file.cache.total.size.bytes = 1073741824
    remote.log.manager.copier.thread.pool.size = 10
    remote.log.manager.copy.max.bytes.per.second = 9223372036854775807
    remote.log.manager.copy.quota.window.num = 11
    remote.log.manager.copy.quota.window.size.seconds = 1
    remote.log.manager.expiration.thread.pool.size = 10
    remote.log.manager.fetch.max.bytes.per.second = 9223372036854775807
    remote.log.manager.fetch.quota.window.num = 11

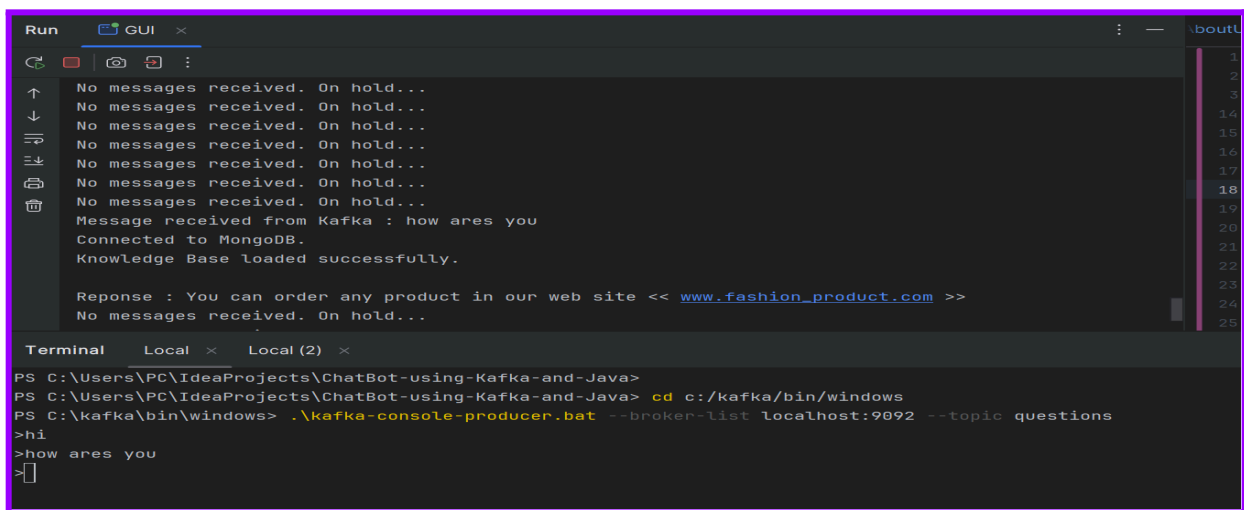
C:\Windows\system32\cmd.exe - C:\kafka\bin\windows\zookeeper-server-start.bat src\main\resources\zookeeper.properties
[2024-12-30 21:42:21,540] INFO Reading configuration from: src\main\resources\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,544] WARN src\main\resources\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,558] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,558] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,559] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,559] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-30 21:42:21,565] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-12-30 21:42:21,566] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-12-30 21:42:21,573] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2024-12-30 21:42:21,573] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2024-12-30 21:42:21,579] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-12-30 21:42:21,580] INFO Reading configuration from: src\main\resources\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

Those logs display critical details, including startup progress, connection statuses, configurations loaded, and any errors or warnings encountered. These logs are essential for monitoring, troubleshooting, and ensuring that the services are running correctly and ready

for use.

II. Test the Application Workflow

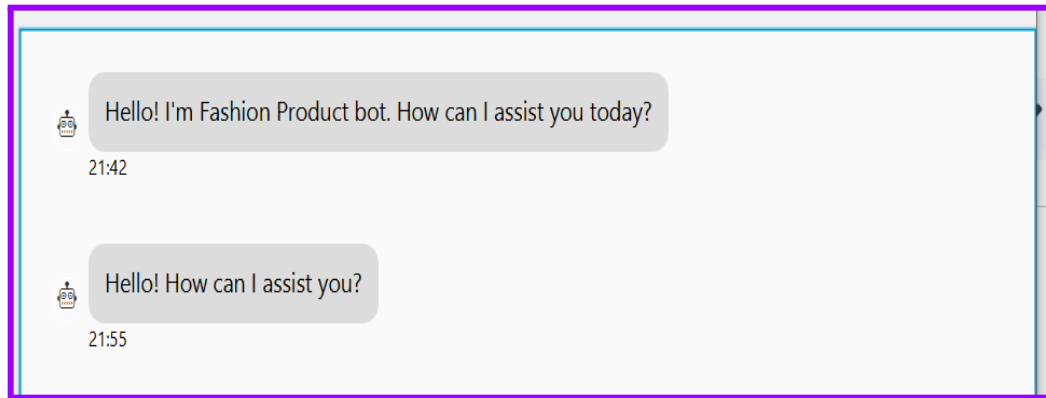
Now we can test with terminal our kafka server , and test if the message produced with producer will receiving by consumer :



```
Run GUI x
No messages received. On hold...
No messages received. On hold...
No messages received. On hold...
No messages received. On hold...
No messages received. On hold...
No messages received. On hold...
No messages received. On hold...
Message received from Kafka : how ares you
Connected to MongoDB.
Knowledge Base loaded successfully.
Reponse : You can order any product in our web site << www.fashion-product.com >>
No messages received. On hold...

Terminal Local x Local (2) x
PS C:\Users\PC\IdeaProjects\ChatBot-using-Kafka-and-Java>
PS C:\Users\PC\IdeaProjects\ChatBot-using-Kafka-and-Java> cd c:/kafka/bin/windows
PS C:\kafka\bin\windows> .\kafka-console-producer.bat --broker-list localhost:9092 --topic questions
>hi
>how ares you
>
```

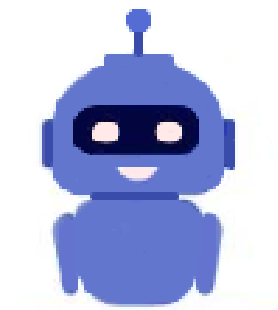
And in the application we can see the message is nice receiving and the chatbot respond for it :



And if we run the consumer we will see this :

```
PS C:\Users\PC\IdeaProjects\ChatBot-using-Kafka-and-Java> cd c:/kafka/bin/windows
PS C:\kafka\bin\windows> .\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic questions --from-beginning
hi
how are you
|
```

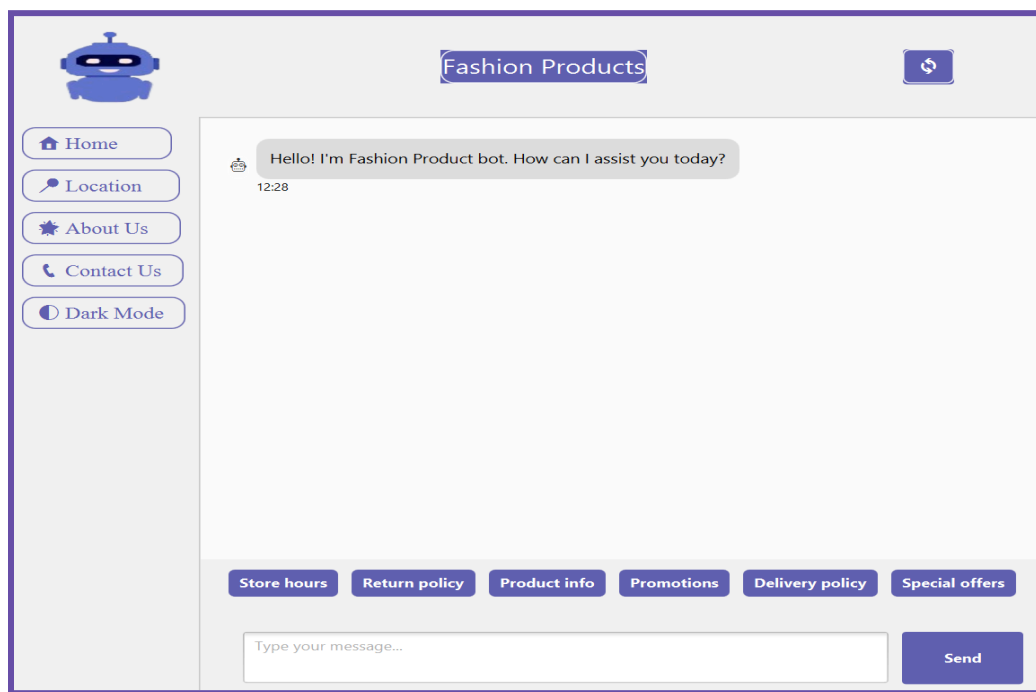
III. Test End-to-End Functionality



To ensure seamless functionality of the chatbot, an end-to-end test is conducted by integrating all components of the system.

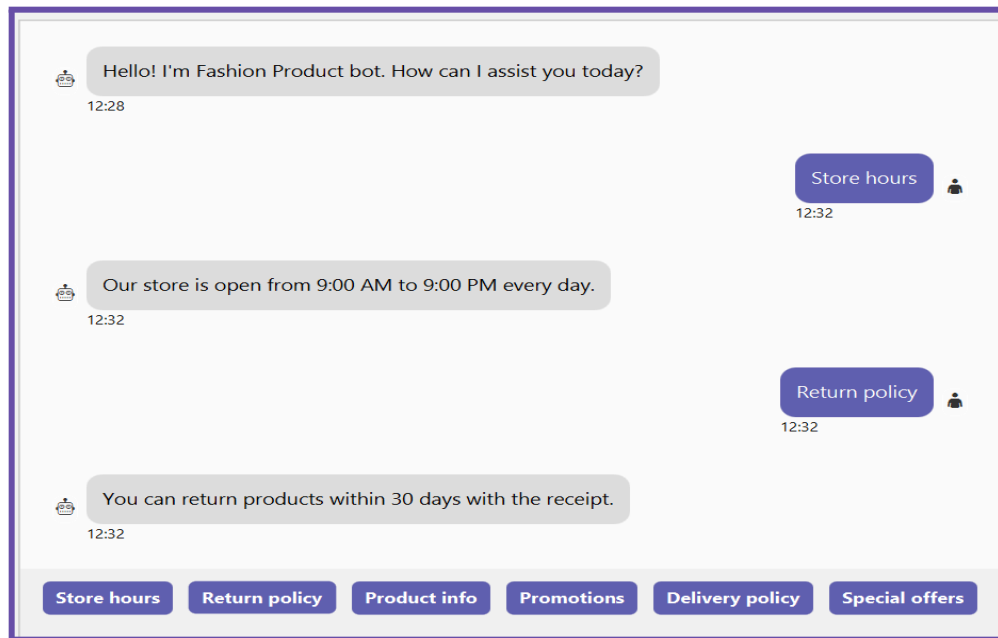
The process begins with launching the chatbot's JavaFX interface, where a user enters a question. This input is sent to Kafka for message handling, then processed by the NLP module to analyze and interpret the query.

The system retrieves the corresponding response from MongoDB or generates a new one, storing it for future use. Finally, the response is displayed in the JavaFX interface, completing the interaction cycle and confirming the proper operation of all interconnected modules.



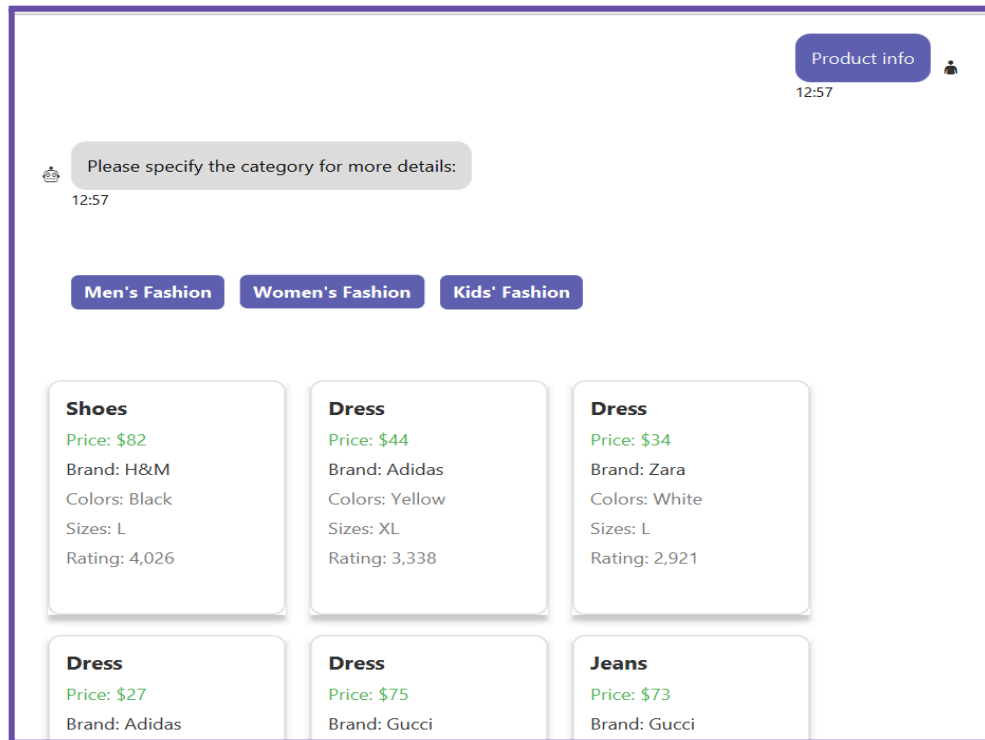
Here we have two modes of chatting : By choices , By chatting in input label.

- ❖ **By choices :** In this mode, users interact with the chatbot by selecting predefined buttons such as "Store Hours," "Return Policy," or "Promotions." This ensures quick and accurate responses to frequently asked questions.

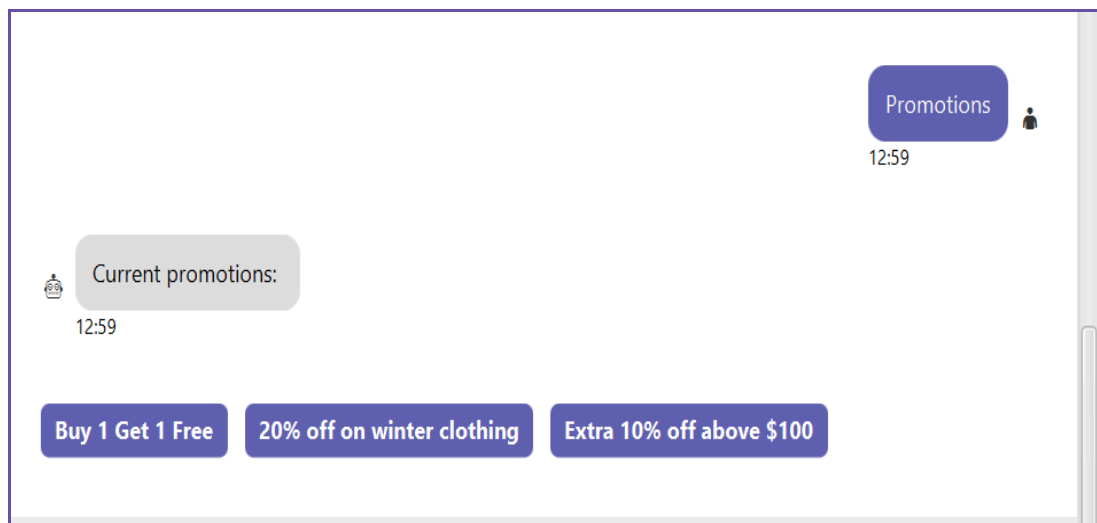


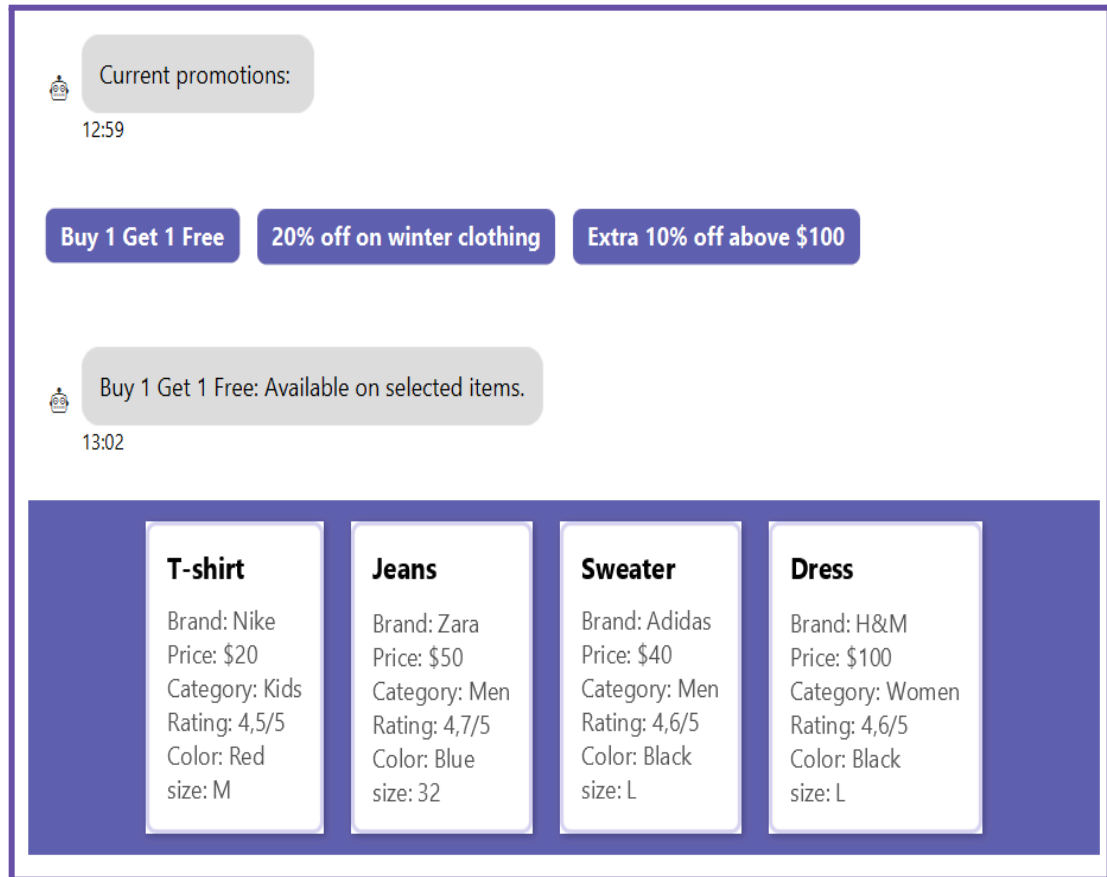
We can choose from the list we have there :

- If we want to see products information , just click **Product info**

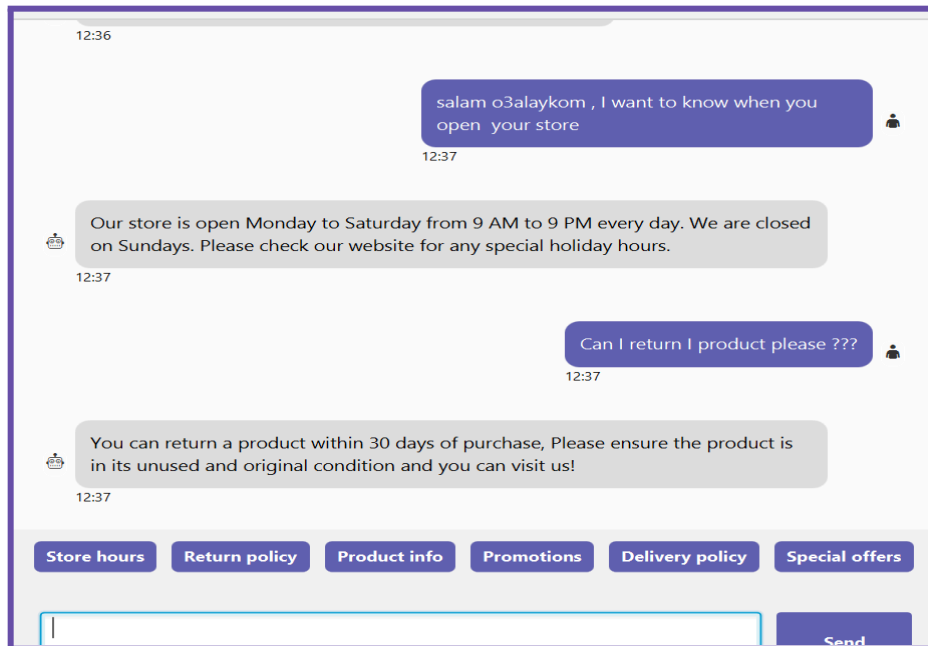


- And if you want to know about promotions that are available in the store just click in the button **promotions**, It will show the products that are available in the promotions



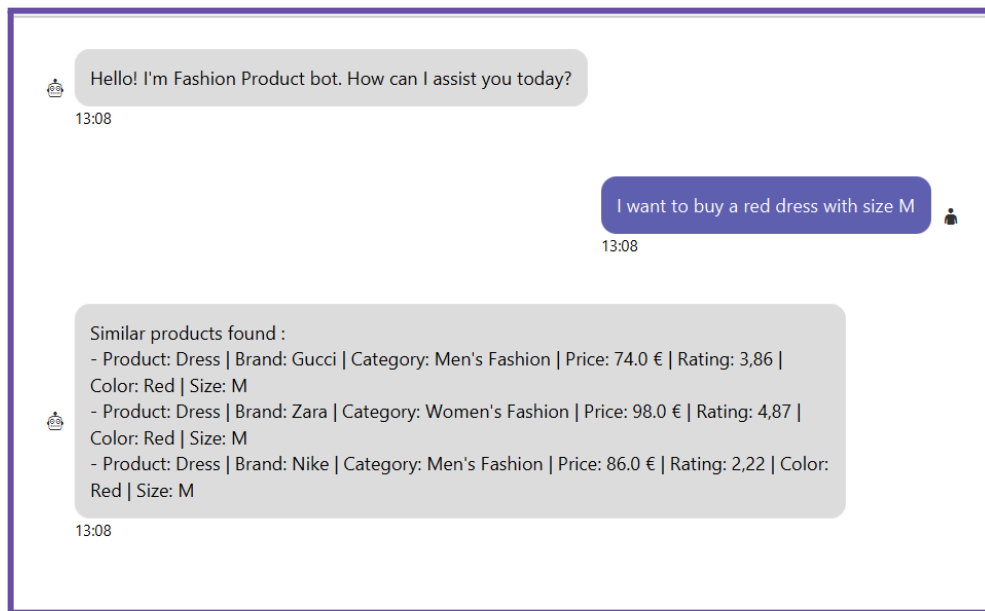


- ❖ **By chatting** : This mode allows users to type their questions directly into the input field. The chatbot processes the query using natural language processing (NLP) and provides a personalized response. This feature offers flexibility for users seeking detailed or specific information.



➤ We can also ask for a specific product in chatting :

When a user searches for a product, the chatbot intelligently filters the inventory to match the specified criteria. For example, if the user requests all available products in size "M," the chatbot retrieves and displays the complete list of items with this size. Similarly, if the user is looking for a red dress, the chatbot narrows the search results to show only dresses in the color red. This feature leverages the product database to deliver precise and relevant information, enhancing the user experience by simplifying the search process.



(Which means that yes we have the product that you seek for and just come to the store to buy it)

But if we don't specify the size , it will show all products for red dress for example.

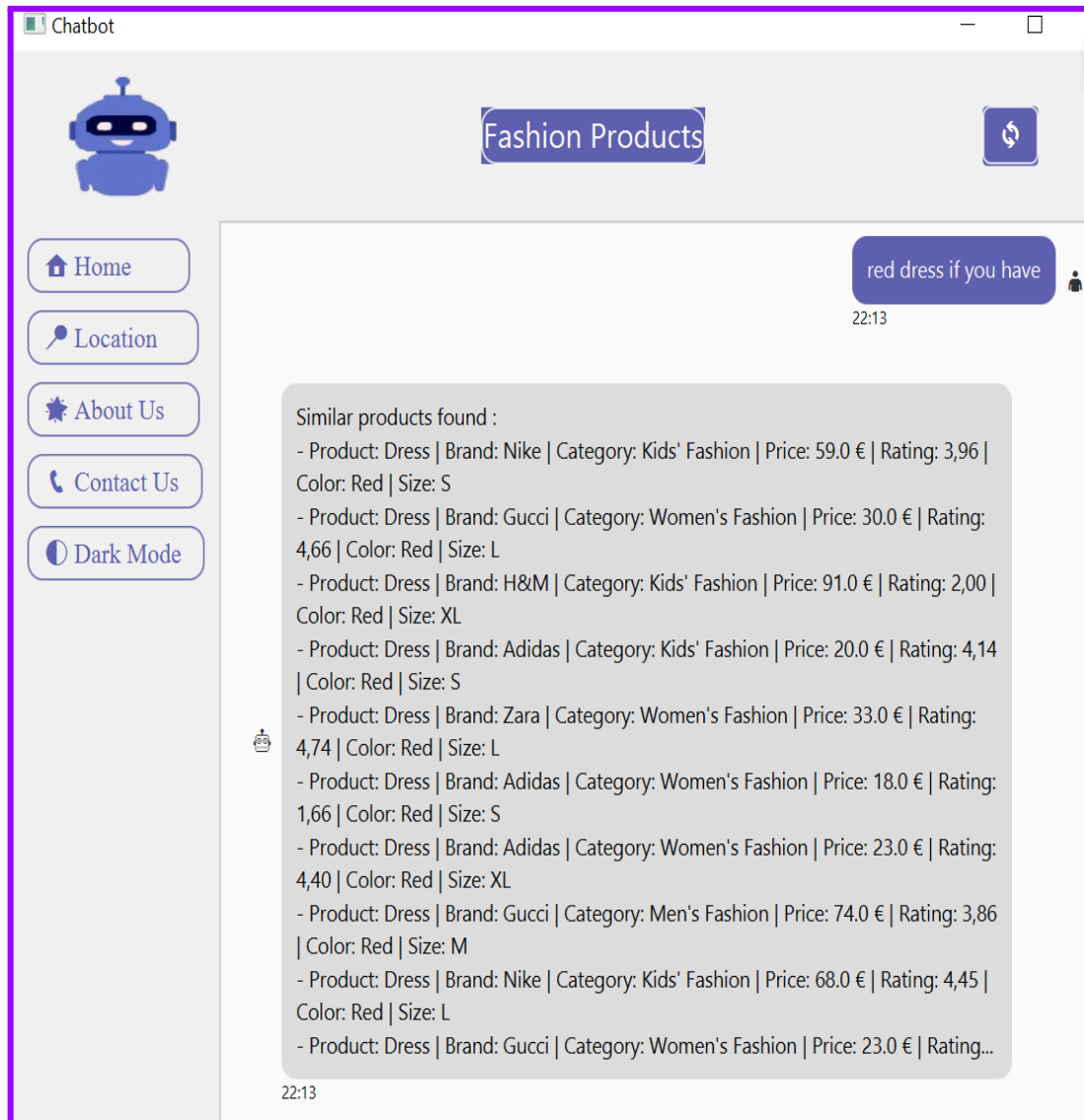
```
No messages received. On hold...
Message envoye au topic avec succes
Message received from Kafka : red dress if you have
Connected to MongoDB.
Knowledge Base loaded successfully.
Message envoyé au topic : questions, Partition: 0, Offset: 2

Reponse : Similar products found :
- Product: Dress | Brand: Nike | Category: Kids' Fashion | Price: 59.0 € | Rating: 3,96 | Color: Red | Size: S
- Product: Dress | Brand: Gucci | Category: Women's Fashion | Price: 30.0 € | Rating: 4,66 | Color: Red | Size: L
- Product: Dress | Brand: H&M | Category: Kids' Fashion | Price: 91.0 € | Rating: 2,00 | Color: Red | Size: XL
- Product: Dress | Brand: Adidas | Category: Kids' Fashion | Price: 20.0 € | Rating: 4,14 | Color: Red | Size: S
- Product: Dress | Brand: Zara | Category: Women's Fashion | Price: 33.0 € | Rating: 4,74 | Color: Red | Size: L
- Product: Dress | Brand: Adidas | Category: Women's Fashion | Price: 18.0 € | Rating: 1,66 | Color: Red | Size: S
- Product: Dress | Brand: Adidas | Category: Women's Fashion | Price: 23.0 € | Rating: 4,40 | Color: Red | Size: XL
- Product: Dress | Brand: Gucci | Category: Men's Fashion | Price: 74.0 € | Rating: 3,86 | Color: Red | Size: M
- Product: Dress | Brand: Nike | Category: Kids' Fashion | Price: 68.0 € | Rating: 4,45 | Color: Red | Size: L
- Product: Dress | Brand: Gucci | Category: Women's Fashion | Price: 23.0 € | Rating: 1,20 | Color: Red | Size: XL

No messages received. On hold...
```

Here's a brief explanation of each functionality in the left menu:

❖ **Home:**

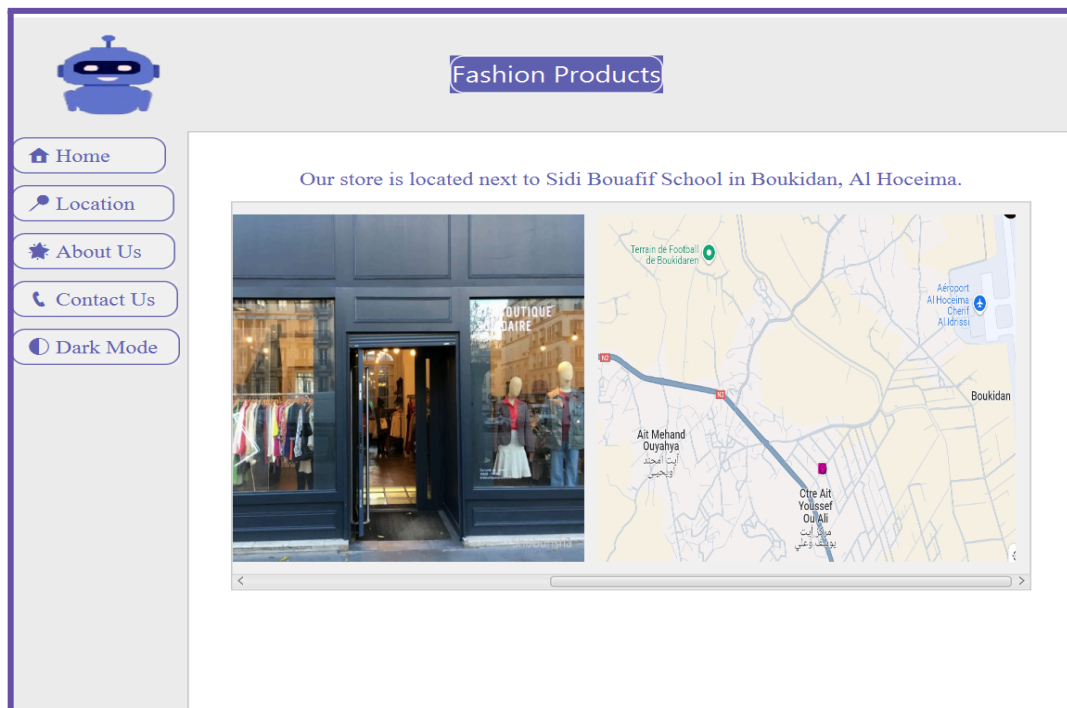


This button redirects users to the chatbot's main page, serving as the starting point for interactions and providing easy navigation back to the main interface.

(and a button in the top right to refresh the chatbot and clear old messages)

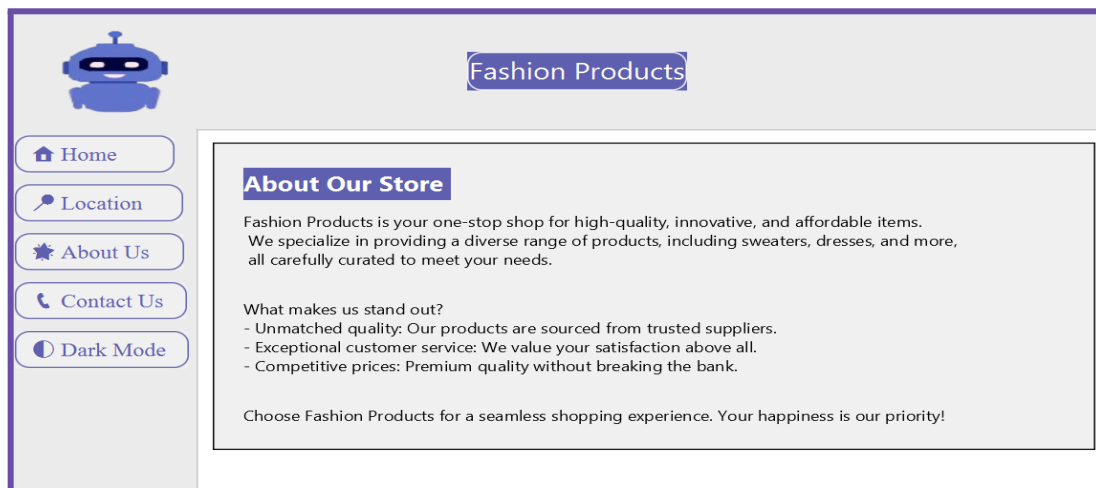
❖ Location:

Displays information about the physical location of the store or business. Users can find the address, directions, or even a map to visit the store.



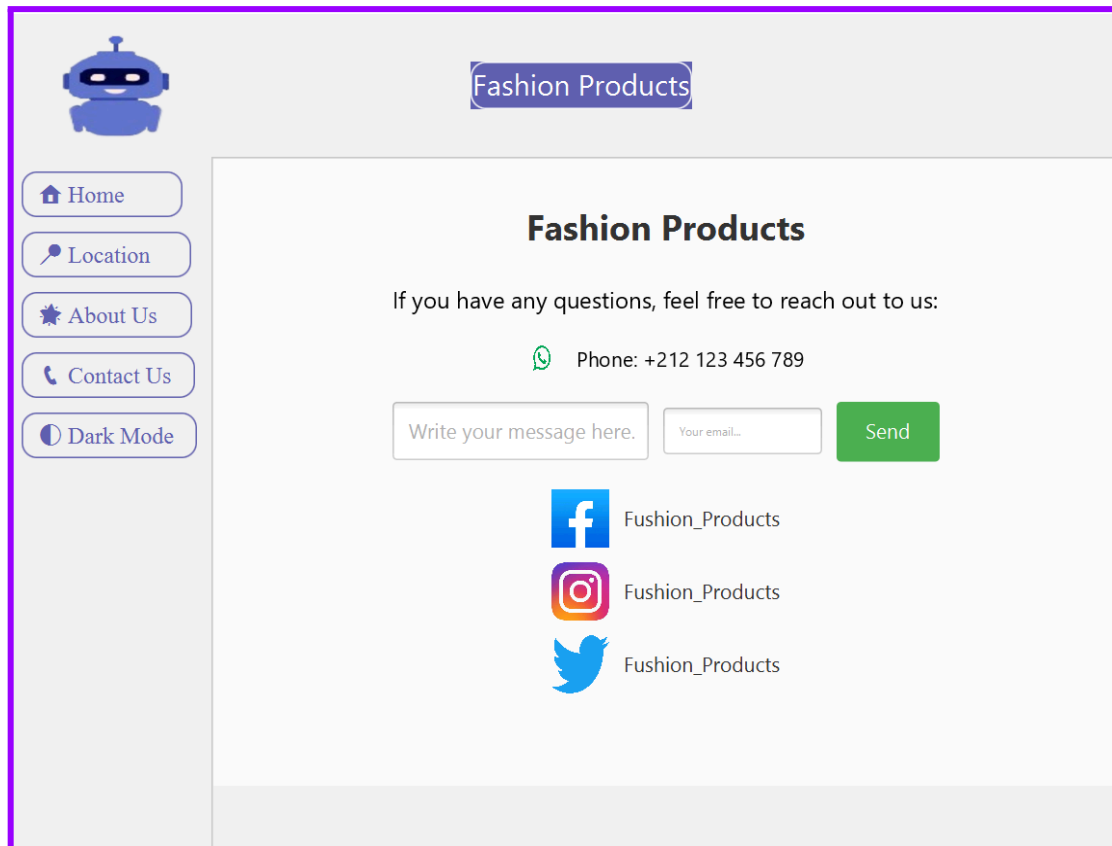
❖ About Us:

Provides details about the business, including its mission, vision, history, and values. It helps users understand the brand better and build trust.



❖ **Contact Us:**

Allows users to access contact details such as phone numbers, email addresses, or links to customer support for further assistance.



Here we can send a message to the manager of the store if we have problems or we want to know something that's not mentioned in the application , and after sending a message , it will appear in the email's box (we will work from this perspective in the future).

❖ **Dark Mode:**

Enables users to switch between light and dark themes for better accessibility and visual comfort, especially in low-light environments.

Conclusion

This project aimed to develop a question-answering chatbot for e-commerce platforms, with a focus on providing customers with a seamless and efficient way to inquire about product availability, store policies, promotions, and personalized recommendations. By integrating advanced technologies such as Kafka for real-time data streaming, Natural Language Processing (NLP) for understanding and processing customer queries, MongoDB for scalable data storage, JavaFX for an interactive user interface, and JSON for efficient data interchange, the chatbot is equipped to handle diverse customer inquiries effectively. The implementation of these technologies has ensured that the chatbot is not only able to respond promptly but also offers an intuitive and dynamic user experience, making it a valuable tool for enhancing customer satisfaction and engagement on e-commerce platforms.

Key components of the system include:

- **Kafka:** Used for real-time data streaming, ensuring that user interactions are processed promptly and effectively. Kafka allows for efficient message brokering and ensures smooth communication between the chatbot's different services, enabling the chatbot to scale and manage a growing volume of requests.
- **Natural Language Processing (NLP):** NLP techniques are used to understand and interpret user inputs in natural language. The chatbot is able to process questions related to specific products, as well as general inquiries about the store, by extracting relevant keywords and matching them with stored data in the system.
- **MongoDB:** The NoSQL database is used to store dynamic, unstructured data such as product details, customer preferences, and chatbot logs. MongoDB's flexible schema design ensures that the database can scale as new products and features are added to the system.
- **JavaFX:** JavaFX enables the creation of a responsive and interactive user interface. It is used to present the chatbot's responses in a conversational format, providing a smooth and engaging user experience. Through JavaFX, real-time updates, animations, and a dynamic layout are seamlessly integrated into the chatbot.
- **JSON:** The data format used for structuring messages exchanged between the backend and the user interface. JSON ensures that data is transmitted efficiently and in a human-readable format, allowing for easy integration between different parts of the system.

By integrating these technologies, the chatbot provides real-time interaction with users, efficiently manages large amounts of data, and delivers personalized responses that enhance the customer experience. This project not only demonstrates the feasibility of using modern

technologies to create a scalable and user-friendly chatbot but also sets the stage for future improvements in AI-driven customer service.

Perspectives

Looking ahead, there are numerous opportunities to enhance the chatbot's functionality and performance. Future advancements may involve the integration of machine learning algorithms, enabling the chatbot to learn from previous interactions and offer more accurate responses over time. Additionally, adding voice recognition for voice-based queries and extending the chatbot's support to multiple languages could expand its reach. There is also potential to integrate the chatbot with other customer service platforms, creating a more comprehensive support experience. By continuing to enhance the chatbot's intelligence and feature set, it can better adapt to evolving customer needs and trends in e-commerce.

Another potential improvement is the integration of the chatbot with CRM systems, enabling it to access and update customer profiles, track order statuses, and provide post-purchase support. This would make the chatbot an even more powerful tool in enhancing the overall e-commerce experience.

Store Manager Communication via Email:

This feature will enable users to send direct messages to the store manager for inquiries or issues not covered by the chatbot. These messages will be forwarded to the store's email inbox for further assistance.

Google Maps API Integration for Store Location:

This feature will allow users to search for their location and receive step-by-step directions to the store, making it easier to find the fashion store.

References

- ❖ **Kafka Documentation.** (n.d.). Apache Kafka. Retrieved from <https://kafka.apache.org/documentation/>
- ❖ **MongoDB Documentation.** (n.d.). MongoDB. Retrieved from <https://www.mongodb.com/docs/>
- ❖ **JavaFX Documentation.** (n.d.). Oracle. Retrieved from <https://openjfx.io/>
- ❖ **Stanford NLP Group.** (n.d.). Stanford CoreNLP. Retrieved from <https://stanfordnlp.github.io/CoreNLP/>
- ❖ **Apache Maven.** (n.d.). Apache Maven. Retrieved from <https://maven.apache.org/>
- ❖ **IntelliJ IDEA Documentation.** (n.d.). JetBrains. Retrieved from <https://www.jetbrains.com/idea/documentation/>
- ❖ **Dataset for products fashion**
<https://www.kaggle.com/datasets/bhanupratapbiswas/fashion-products/discussion/416907>