INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

AI shopping Assistant

Amr Abd Alrahman - a49324

Thesis presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

Prof. Rui Pedro Lopes

Bragança

2024-2025

AI shopping Assistant

Amr Abd Alrahman - a49324

Thesis presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

Prof. Rui Pedro Lopes

Bragança

2024-2025

Abstract

This project addresses the challenge of online shoppers feeling overwhelmed by the vast array of options available on e-commerce platforms, a problem that often leads to cart abandonment and revenue losses for store owners. The proposed solution is the development of an AI-powered virtual fashion assistant capable of understanding customer preferences in real-time and suggesting curated product recommendations. By offering personalized, streamlined guidance, the assistant simplifies the journey from browsing to checkout while preserving the customer's freedom to design a custom look. This approach not only enhances user satisfaction but also increases conversion rates and customer retention for online retailers.

Keywords: Fashion Assistant, Virtual Assistant, Artificial Intelligence, E-commerce, Personalized Shopping

Contents

List of Figures

Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

DB Data Base.

ESTiG Escola Superior de Tecnologia e Gestão.

HTTP HyperText Transfer Protocol.

IPB Instituto Politécnico de Bragança.

KB Knowledge Base.

NLP Natural Language Processing.

SEO Search Engine Optimization.

SSG Static Site Generation.

SSR Server Side Rendering.

UI User Interface.

UX User Experience.

Chapter 1

Introduction

The rapid growth of e-commerce, particularly in the fashion industry, has transformed how consumers shop. With the convenience of online shopping, customers now have access to a vast array of products from the comfort of their homes. However, while variety is often seen as a strength, the sheer abundance of choices can lead to decision fatigue, overwhelming shoppers. This cognitive overload not only reduces the overall shopping experience but also results in cart abandonment, a common issue for many online retailers. As such, there is a pressing need for solutions that streamline the shopping journey and help users make more informed, confident purchasing decisions.

In response to these challenges, Artificial Intelligence (AI) has emerged as a powerful tool in the e-commerce sector. AI technologies, including machine learning, natural language processing, and data analytics, are transforming the way online stores interact with their customers. By offering personalized product recommendations and anticipating customer preferences, AI helps to reduce decision fatigue and enhance the shopping experience. Virtual assistants and chatbots, in particular, are gaining popularity in online retail, offering customers real-time, tailored support throughout their shopping journey. In the fashion industry, where trends are ever-changing and consumer preferences vary widely, AI-driven solutions provide a unique opportunity to curate personalized shopping experiences that truly resonate with individual customers.

Despite the advancements in AI, many online clothing retailers still face significant

challenges related to low conversion rates and high cart abandonment. A key issue is the lack of personalized guidance, which often leaves shoppers struggling to find products that align with their tastes or fit specific occasions. This gap in the shopping experience presents an exciting opportunity for innovation, particularly through the use of AI-powered virtual assistants. These systems can engage customers in a more interactive and personalized manner, guiding them through product selection, suggesting complete outfits, and ultimately simplifying the decision-making process. By incorporating conversational AI, retailers have the chance to not only improve user satisfaction but also drive higher conversion rates and customer loyalty.

1.1 Objective

The project aims to develop an AI-powered shopping assistant designed to assist customers when they feel overwhelmed, lost, or are searching for a specific style. Through a conversational chat interface, customers will be able to describe the type of style they are looking for — including preferred colors, themes, occasions, and other personal preferences.

One of the main reasons customers abandon online clothing stores before reaching the checkout stage is the overwhelming number of product options, which can lead to decision fatigue, frustration, or boredom. This project embraces the growing trend of using AI technologies to streamline and personalize user experiences, making shopping faster, more efficient, and more engaging. By offering intelligent product suggestions tailored to the customer's desires, the AI assistant aims to transform the shopping journey into an interactive and enjoyable experience, ultimately increasing the likelihood of customers reaching the checkout stage and completing their purchases.

1.2 Structure of the document

This thesis is organized into six chapters, each addressing a key aspect of the development and implementation of the AI-powered shopping assistant. The structure of the document is as follows:

- **Chapter 1: Introduction** - This chapter introduces the project, outlining the objectives and the motivation behind the development of the AI-powered shopping assistant. It highlights the issues faced by customers in online shopping and the role of AI in improving the user experience.

- **Chapter 2: Context and Technologies** - This chapter discusses the challenges in online shopping, such as decision fatigue, and presents the AI-powered virtual fashion assistant as a solution. It reviews key concepts in conversational AI and explores the technologies used in the project, including Voiceflow, Google Sheets, n8n, and Next.js, justifying their selection for rapid development and integration.

- **Chapter 3: System Analysis** - This chapter outlines the challenges of online shopping, particularly decision fatigue and high cart abandonment rates. It presents the AI-powered fashion assistant as a solution, describing its core functionalities such as entity recognition, personalized recommendations, and dynamic response generation. The chapter also details the system architecture, including the integration of Google Sheets, n8n, Voiceflow, and Next.js, and provides an overview of the modeling and interaction design, ensuring a seamless and personalized user experience.

- **Chapter 4: Development** - This chapter details the development of the AI-powered virtual fashion assistant, focusing on system architecture, integration, and the technologies used. It covers the setup of Google Sheets as the product catalog, the use of n8n for workflow automation, and the implementation of Voiceflow for conversational AI design. The chapter also discusses the frontend integration with Next.js, ensuring smooth user interactions, and addresses key technical challenges,

such as maintaining data synchronization, ensuring structured AI responses, and generating cohesive outfit suggestions.

- **Chapter 5: Tests and Discussion** - This chapter presents the testing phase of the AI-powered fashion assistant, focusing on both functional verification and user experience validation. It outlines the methodology, key test cases, and results, ensuring that the system performs as expected. The chapter also analyzes the outcomes, discussing successes such as accurate entity recognition, seamless system integration, and a positive user experience, as well as challenges encountered, like handling ambiguous inputs and edge cases. Finally, it offers insights for future improvements to enhance the system's accuracy, scalability, and personalization.

- **Chapter 6: Conclusions** - This chapter summarizes the successful development of the AI-powered virtual fashion assistant, highlighting key accomplishments such as the integration of Voiceflow for conversational AI, Google Sheets for product catalog management, and Next.js for frontend development. The system delivers a personalized, efficient shopping experience by accurately capturing user preferences and providing real-time product recommendations. It also discusses design choices that contributed to the project's success, as well as its limitations and areas for future improvement, including expanded product categories, user memory, and more advanced AI models.

Chapter 2

Context and Technologies

The rapid expansion of online shopping platforms, particularly in the fashion industry, has created an environment where customers are faced with an overwhelming number of choices. While variety is traditionally viewed as a strength, an excess of options often leads to decision fatigue, frustration, and ultimately to abandoned shopping carts. Studies have shown that many online shoppers either leave without completing a purchase or avoid returning to platforms that make the decision-making process overly complex, resulting in significant revenue losses for merchants annually.

This project addresses this critical issue by introducing an AI-powered virtual fashion assistant designed to simplify the customer journey. Through real-time interaction via chat, the assistant captures user preferences — such as style, color, and occasion — and offers curated clothing recommendations that are personalized to each customer's taste. By reducing cognitive overload and streamlining the selection process, the assistant aims to enhance customer satisfaction, boost conversion rates, and improve long-term customer loyalty for online retailers.

2.1 Scope of the Project

The primary objective of this project is to design and implement an AI-powered virtual fashion assistant that enhances the online shopping experience by providing personalized

clothing recommendations. The assistant is intended to operate within an e-commerce environment, interacting with users through a conversational interface to understand their style preferences, favorite colors, occasions, and desired vibes.

The solution targets online shoppers who may feel overwhelmed by the large volume of available products or those seeking assistance in curating a complete look. By offering tailored suggestions, the assistant aims to simplify the decision-making process, making shopping more engaging, efficient, and satisfying.

The system is also designed with flexibility in mind, allowing users to either select individual products — such as a t-shirt, trousers, or footwear — or receive complete outfit suggestions that align with their preferences. The ultimate goal is to increase customer satisfaction, reduce cart abandonment rates, and boost conversion rates for online clothing retailers.

2.2 Concepts and Trends

The rapid advancement of Artificial Intelligence (AI) has revolutionized human-computer interaction, especially through conversational agents such as virtual assistants and chatbots. These systems aim to simulate intelligent conversation with users, providing assistance, automation, and personalization across various industries. Their adoption in e-commerce has been particularly transformative, helping customers navigate product catalogs, make decisions, and complete purchases with guided support.

This section provides an overview of the core concepts underpinning conversational AI, the current state of virtual assistant deployment, and the technologies that enable these experiences. We also highlight recent trends that shape the development of intelligent, context-aware, and user-centric assistants.

2.2.1 Foundations of Conversational AI

Conversational AI systems are primarily built on three components: intent recognition, dialog management, and natural language generation. At the heart of these systems is

the ability to understand what the user wants (intent) and extract relevant information (slots) from natural language input.

Intent Recognition and Slot Filling

Intent recognition refers to the process of classifying a user's input into predefined actions, such as "search for product" or "add to cart". Slot filling refers to identifying structured entities within the query, such as the product type, color, or budget. These processes are typically powered by Natural Language Understanding (NLU) pipelines trained on labeled utterances and entity schemas [1], [2].

Dialog Management Systems

Dialog management is the logic engine that governs how a virtual assistant responds across turns. Traditional dialog systems relied on rule-based logic or state machines. More recent approaches integrate machine learning and probabilistic models to allow for greater flexibility and personalization [3], [4]. Hybrid systems, which combine rule-based intent handling with neural model generation, are common in commercial applications.

Intent-Language Models (ILMs)

Intent-Language Models (ILMs) are an emerging class of models that integrate the objectives of intent classification and response generation into a unified language model architecture. These models are particularly useful in scenarios requiring structured outputs, such as JSON responses, while maintaining the semantic coherence of open-ended dialog. ILMs fine-tune transformer-based architectures (e.g., BERT, T5, or GPT) on multi-turn dialog datasets annotated with intents and slot values [5], [6].

2.2.2 Virtual Assistants and Chatbots in Practice

Virtual assistants are now widely used across industries to automate customer interactions, support decision-making, and enhance engagement. Their success is due in part

to improved natural language processing, growing integration capabilities, and shifting consumer expectations toward more conversational interfaces.

Industry Adoption

In e-commerce, virtual assistants help users filter through product options, offer recommendations based on preferences, and support checkout processes [7], [8]. In banking and finance, they assist with balance inquiries, fraud alerts, and customer onboarding [9]. In healthcare, chatbots are used for triage, appointment scheduling, and patient education [10], [11].

These use cases demonstrate how chatbots have moved beyond simple FAQs into roles that carry operational and business value, reducing support costs while improving satisfaction and retention rates.

User Experience and Trust

The effectiveness of a virtual assistant hinges on its ability to deliver relevant responses while maintaining a coherent conversational flow. Studies have shown that users are more likely to trust assistants that demonstrate empathy, personalization, and transparency in how data is used [12], [13]. As assistants become more embedded in decision-making processes, user trust plays a central role in adoption.

Additionally, chatbots that disclose their non-human identity tend to be perceived as more trustworthy than those that attempt to imitate humans too closely [14].

Multimodal and Multilingual Interactions

Modern assistants increasingly support multimodal input—allowing users to interact using voice, text, and sometimes images. Platforms such as Google Assistant and Amazon Alexa already integrate voice and screen responses. In multilingual regions, the ability to support multiple languages and dialects is a key requirement for inclusivity and accessibility [15], [16].

The ability to capture and preserve context across modalities and languages is becoming a differentiator in conversational system quality.

2.2.3 Technological Platforms and Frameworks

The development of virtual assistants and chatbots has been accelerated by the availability of robust frameworks and platforms. These tools provide pre-built components for dialog management, natural language processing, integration, and testing, enabling faster development and lower entry barriers for both technical and non-technical teams.

Voiceflow

Voiceflow is a no-code/low-code platform designed for creating conversational assistants that support multi-modal and multi-channel deployments. It allows designers to define intents, variables, flows, and API integrations using a visual canvas. More recently, it has introduced support for large language models (LLMs) and structured JSON outputs, making it suitable for building sophisticated assistants that can interact with APIs and external databases. Voiceflow supports integrations with messaging platforms, smart speakers, and web chat environments [17].

Rasa

Rasa is an open-source conversational AI framework that offers flexibility and full control over the assistant's behavior. It includes modules for intent recognition (Rasa NLU), dialog management (Rasa Core), and deployment infrastructure. Rasa's architecture supports custom actions, entity extraction, and contextual handling, making it ideal for enterprise use cases where control over data and infrastructure is crucial [18].

Dialogflow

Google's Dialogflow provides pre-trained agents and tools to build natural language interfaces for various use cases. It supports multi-language processing, integration with Google

Cloud services, and fulfillment through webhooks. While easier to use than open-source alternatives, it can be limiting in terms of customization [19].

IBM Watson Assistant and Others

IBM Watson Assistant combines intent classification, dialog modeling, and search capabilities (retrieval augmented generation). It also provides an enterprise-grade UI and integration suite. Other platforms such as Microsoft Bot Framework and Amazon Lex similarly offer scalable tools with cloud integration and enterprise support.

No-Code and Low-Code Trends

A significant trend in recent years is the adoption of no-code and low-code platforms. These tools reduce development complexity, allowing designers, marketers, and business teams to contribute directly to assistant workflows without coding. This democratization of conversational design has opened opportunities for more agile development and cross-functional collaboration [20], [21].

2.2.4 Current Trends and Future Directions

As the field of conversational AI matures, several emerging trends are shaping the development of next-generation virtual assistants. These trends reflect growing expectations around intelligence, personalization, scalability, and integration.

Integration with Large Language Models (LLMs)

The rise of foundation models such as OpenAI's GPT, Google's Gemini, Anthropic's Claude, and Meta's LLaMA has introduced new paradigms for building assistants. These models offer open-ended generation, semantic understanding, and context retention across long conversations. While powerful, they pose challenges in terms of control, safety, and grounding responses in reliable data sources [22]–[24].

To address this, developers combine LLMs with external tools (e.g., APIs, retrieval systems, or memory modules) in what is referred to as Toolformer or ReAct paradigms [25], [26].

Structured Output and Schema Alignment

For assistants embedded in transactional workflows, generating structured output (e.g., JSON) instead of free-form text has become essential. Structured responses allow front-end systems to render cards, buttons, and product layouts programmatically. This also enables assistants to interface directly with external APIs and databases [27], [28].

Schema-guided generation is a trend gaining momentum, where the assistant is constrained to follow pre-defined schemas using techniques like function calling or output format validation.

Personalization and Context Awareness

There is a shift toward assistants that remember user preferences, style history, and intent patterns to offer a more personalized experience. Context-aware agents can adapt recommendations and tone based on prior interaction history, user profiles, or session metadata. Techniques such as long-term memory, vector-based retrieval, and persona modeling are being integrated into assistants to sustain relevance and consistency [29], [30].

Ethics, Bias, and Transparency

With the widespread use of conversational systems, questions about fairness, transparency, and user data privacy have come to the forefront. Research emphasizes the need for assistants to explain their reasoning, avoid biased responses, and comply with data protection regulations such as GDPR [31], [32].

Developers are increasingly incorporating explainability techniques, such as intent confidence thresholds, and flagging unknown queries for human review to ensure ethical

deployment.

Multilingual, Multimodal, and Omnichannel Assistants

Modern assistants are being deployed across diverse platforms (e.g., websites, mobile apps, WhatsApp, voice devices) and need to support multimodal interaction. Simultaneously, multilingual capabilities are becoming non-negotiable for global access. End-to-end architectures that process text, speech, and vision are emerging to meet this demand [33], [34].

Conversational Commerce and Hybrid Interfaces

In e-commerce, assistants are evolving into full shopping agents capable of curating outfits, managing carts, and answering product queries within a single conversation. Hybrid interfaces, combining chat with clickable elements (e.g., carousels, filters, visual selectors), offer the best of both conversational and traditional UIs [35], [36].

In conclusion, the convergence of large language models, structured reasoning, personalization, and multimodal support is reshaping what is possible with virtual assistants. The next generation of chatbots will not merely answer queries—they will act, adapt, and co-create with the user.

2.3 Technologies Used

The development of the AI-powered virtual fashion assistant relies on a set of complementary technologies, each selected to efficiently address a specific need within the system architecture. The following tools and platforms were used.

The Level 0 Data Flow Diagram (DFD) shown in 2.1 provides a high-level overview of the interactions within the AI-powered fashion assistant system, depicting the system's main process, entities, data storage, and data flows. At the core of the diagram is the

AI Fashion Assistant System, which manages the overall functionality of the assistant, including user interaction, intent processing, and product recommendations.

The system interacts with several key entities. First, the User provides input through their preferences, such as desired style, color, or occasion, which are sent to the AI Fashion Assistant System. In response, the system suggests relevant outfits to the user, completing the cycle of interaction. To process user input, the system interfaces with Voiceflow, which is responsible for interpreting the user's intent and searching the Knowledge Base (KB). Once Voiceflow has processed the request, it sends a structured JSON response with personalized product recommendations back to the AI Fashion Assistant System.

The n8n workflow automation tool plays a critical role in keeping the system updated by periodically syncing data. Specifically, n8n ensures that the Knowledge Base is updated with the latest product information. Additionally, Google Sheets serves as the product catalog database, providing essential details about the available items. The AI Fashion Assistant System retrieves this product information from Google Sheets as needed to generate accurate recommendations.

The flow of data is centralized around the AI Fashion Assistant System, which manages the communication between the user, Voiceflow, n8n, and Google Sheets. The system handles the intake of user preferences, processes these through Voiceflow, and updates the knowledge base via n8n. Finally, it retrieves product data from Google Sheets to ensure that the recommendations are based on the most current inventory.

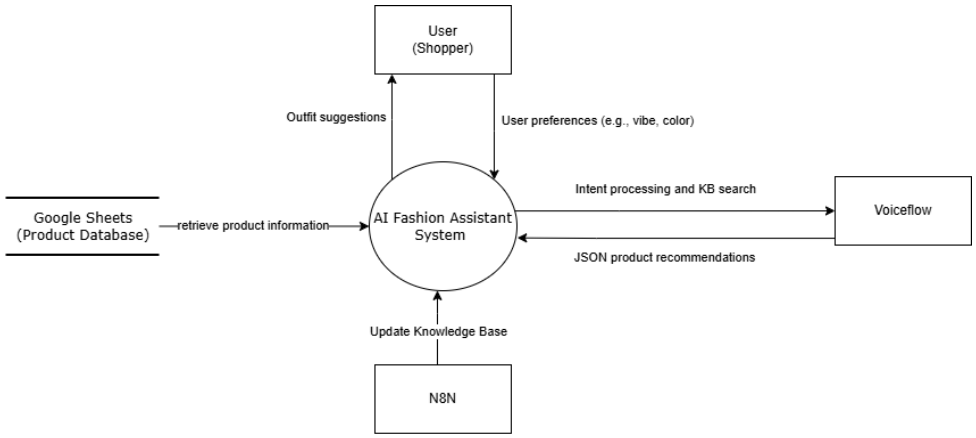

Figure 2.1: Level 0 data flow diagram

The **DFD Level 1** shown in 2.2 provides a more granular view of the AI-powered fashion assistant system, breaking down the high-level processes into individual steps that facilitate detailed interactions between the entities, storage, and processes. The core processes include *show outfit suggestion*, *capture user input & preferences*, *generate conversation & suggestion*, *fetch product details*, and *update KB periodically*. These processes work together to deliver a seamless and personalized shopping experience.

The flow begins with the **User (Shopper)**, who provides their preferences, such as desired *vibe*, *color*, and *product focus*. These preferences are captured through the *capture user input & preferences* process, which then forwards the information to the **Voiceflow Dialogue API**. This step initiates the conversation and suggestion generation process by leveraging the user preferences.

Next, the **Voiceflow Dialogue API** processes the captured preferences and generates a conversation and product suggestion. It queries the **Voiceflow KB** to retrieve the relevant product chunks that match the user's input. The **Voiceflow KB**, which stores the structured product data, responds with the necessary chunks of information, which are then processed by *generate conversation & suggestion*.

Once the suggestions are generated, the system sends a *JSON format recommendation* to the **Frontend (Next.js)**, which handles the display of the recommended products. The frontend requests the full product details by fetching them from **Google Sheets** and **n8n**, ensuring the user receives complete information such as product name, image, and price. The **n8n** automation tool plays a vital role in syncing the **Google Sheets Product Database** with the **Voiceflow KB** to ensure that the knowledge base is periodically updated with the latest product information.

Finally, the **Frontend (Next.js)** displays the *outfit suggestion* with all product details, completing the cycle by showing the user the personalized recommendations based on their preferences.

Throughout this process, **n8n** ensures periodic updates to the **Voiceflow KB**, maintaining an up-to-date catalog and product suggestions. The **Google Sheets** product catalog serves as the source of truth, providing detailed information for the assistant to

generate relevant recommendations.

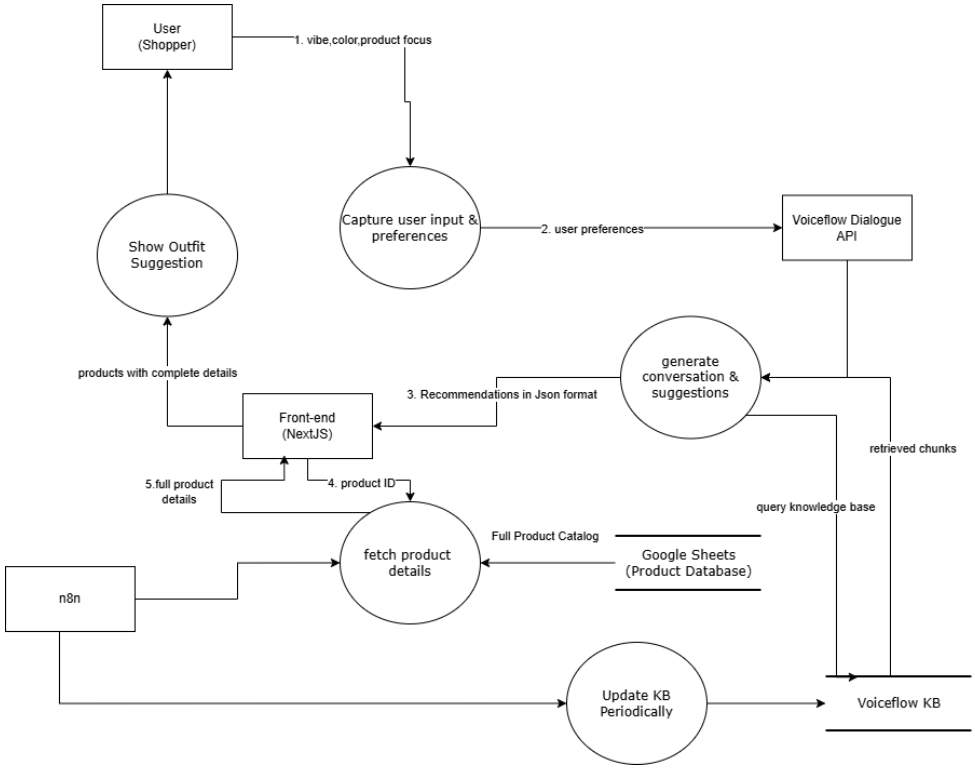

Figure 2.2: Level 1 data flow diagram

The sequence diagram in Figure 2.3 illustrates the flow of interactions between the user, the Next.js frontend, the Voiceflow Dialogue API, and n8n during a typical product recommendation session.

The process begins when the user inputs their preferences, such as desired style, color, or occasion, into the Next.js frontend (Step 1). This input is then sent to the Voiceflow Dialogue API (Step 2), where the AI assistant processes the data. Voiceflow extracts relevant entities from the user input and performs a knowledge base search to find potential product matches (Step 3). Based on this search, Voiceflow returns the relevant product information in the form of product IDs (Step 4).

The Next.js frontend then receives a structured JSON response containing these product IDs from the Voiceflow Dialogue API (Step 5). For each product received in the response, the frontend enters a loop (Step 6), where it sends a request to the n8n API

endpoint to retrieve detailed information about each product. The n8n service queries the product catalog stored in Google Sheets (Step 6.1), processes the data, and returns the full product details, including name, image, price, and other relevant attributes (Step 6.2). These details are then stored by the frontend (Step 6.3).

Finally, after gathering the complete information for all products, the Next.js frontend dynamically displays the suggested outfits to the user, including product names, images, and prices (Step 7). This interactive and seamless flow of data ensures that users receive personalized and accurate product recommendations, enhancing their shopping experience.

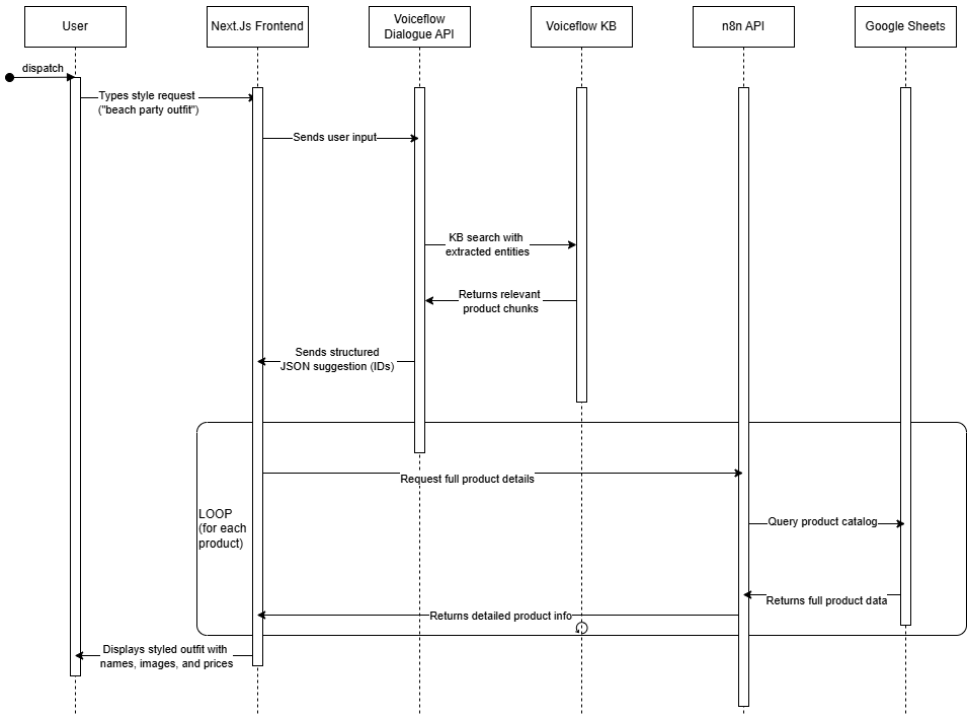

Figure 2.3: Sequence Diagram

2.3.1 Voiceflow

Voiceflow is a conversational AI design and deployment platform that enables the creation of sophisticated voice and chat assistants without the need for extensive manual coding. In this project, Voiceflow serves as the core platform for building the shopping assistant's

dialogue flow, managing knowledge base queries, capturing user preferences through entity recognition, and ensuring structured JSON output. Its flexibility and intuitive interface allowed for rapid prototyping and easy integration with external APIs.

2.3.2 Google Sheets

Google Sheets was chosen as the primary data repository for this project due to its simplicity, accessibility, and integration capabilities. While it is not a traditional relational database or RESTful service, it provided a lightweight and cost-effective solution during the early stages of development. Its real-time collaboration features, ease of editing, and compatibility with automation tools like n8n allowed for rapid prototyping and iteration. Given the relatively small scale of the product catalog and the focus on conversational interaction rather than complex transactional operations, Google Sheets offered sufficient functionality without the overhead of managing a full-fledged database infrastructure.

2.3.3 n8n

n8n is an open-source workflow automation tool that connects different systems and manages data synchronization. It is used in this project to automate the extraction of data from Google Sheets, transform it into the appropriate format, and upload it to Voiceflow's Knowledge Base via API calls. Additionally, n8n exposes the product catalog as API endpoints that the Next.js application can query dynamically.

2.3.4 Next.js

Next.js, a popular React-based web development framework, was used to build the frontend e-commerce platform. It provides features such as server-side rendering, static site generation, and API route handling, which contribute to a fast, SEO-friendly, and responsive user experience. The application interfaces with the n8n-generated APIs to display updated product information to users in real time.

2.4 Justification of Technology Choices

The selection of technologies for this project was guided by the need for rapid development, flexibility, scalability, and ease of integration between systems.

Voiceflow was chosen for building the conversational AI due to its ability to handle complex dialogue management, entity extraction, and structured outputs, without requiring extensive backend development. Its native support for API calls and knowledge base integration made it particularly well-suited for a dynamic, evolving product catalog environment.

Google Sheets served as the initial database solution because of its simplicity, accessibility, and collaborative features. For early-stage development and testing, a lightweight system was preferable to more complex database solutions, as it allowed fast updates, easy manual adjustments, and real-time viewing of the product catalog without additional infrastructure overhead.

n8n was selected to act as the middleware between Google Sheets, Voiceflow, and the Next.js application. Its no-code/low-code workflow design allowed quick creation of automated data pipelines, handling tasks such as API request formatting, data transformation, and endpoint exposure without the need for complex server setups.

Next.js was chosen for the frontend framework due to its robustness, developer experience, and built-in features like server-side rendering (SSR) and static site generation (SSG). These capabilities are essential for delivering fast page loads, SEO optimization, and responsive user interfaces, which are critical factors in modern e-commerce environments.

Overall, the technology stack balances ease of initial deployment with the ability to grow and scale as project complexity increases. Each component integrates smoothly with the others, supporting the overall goal of creating an intelligent, dynamic, and user-friendly fashion shopping experience.

Chapter 3

System Analysis

The growth of online shopping platforms has significantly expanded customer access to a vast range of fashion products. However, this abundance of options often overwhelms users, leading to decision fatigue, frustration, and ultimately a decline in conversion rates. Instead of enhancing the shopping experience, the excessive choice paradoxically impairs it, as customers may abandon their carts or disengage from the platform entirely.

Traditional filtering systems in online stores, such as category filters and search bars, only partially address this issue. These mechanisms still require users to navigate through multiple product pages, compare items manually, and invest considerable effort in assembling a coherent outfit. The lack of personalized guidance leaves many customers feeling unsupported, especially when seeking fashion combinations that match a particular style, vibe, or occasion.

From the retailer's perspective, high cart abandonment rates and low customer retention represent a substantial loss of potential revenue. Without a solution that simplifies the shopping experience and offers tailored recommendations, stores miss opportunities to engage users meaningfully and convert visits into purchases.

Therefore, there is a clear need for a system that can:

- Understand user preferences intuitively,
- Reduce the complexity of decision-making,

- Offer curated product suggestions,
- Allow users to quickly move from browsing to checkout without feeling overwhelmed.

This project aims to address these needs by developing an AI-powered fashion assistant capable of interpreting user intentions and presenting streamlined, personalized recommendations, ultimately improving customer satisfaction and increasing store profitability.

3.1 Proposed Solution

To address the challenges faced by online clothing shoppers, the project proposes the development of an AI-powered virtual fashion assistant. This assistant is designed to interact with users through a conversational chat interface, capturing their preferences and needs in real time to offer curated product suggestions.

The virtual assistant allows users to describe their desired styles, preferred colors, specific occasions, or overall vibes. Using this information, the assistant queries a structured knowledge base of fashion products and intelligently suggests combinations that match the customer's profile. A high-level overview of the proposed solution is provided in Figure 4.1.

The core functionalities of the assistant include:

- **Entity Recognition**: Capturing essential attributes such as *product focus* (e.g., t-shirt, trousers, footwear), *favorite color*, and *desired vibe or occasion* from user input.
- **Personalized Recommendations**: Suggesting individual products or complete outfits based on captured preferences.
- **Dynamic Response Generation**: Adapting the level of detail and suggestions based on the user's expressed interest (e.g., focusing only on a t-shirt or proposing a full look).

By tailoring the shopping experience to individual needs and reducing cognitive overload, the assistant seeks to make the browsing process more enjoyable and efficient. Ultimately, the solution aims to drive higher customer satisfaction, lower abandonment rates, and increase conversion rates for online fashion retailers.

3.2 System Requirements

In any software development project, requirements are essential to define the system's expected behavior and performance. They are typically categorized into two main types: **functional** and **non-functional** requirements. These categories help ensure that the system meets both user expectations and technical specifications.

Functional Requirements: Functional Requirements refer to the specific functionalities and features that the system must be able to perform. These requirements describe the interactions between the user and the system, outlining the tasks the system should accomplish to fulfill its purpose.

Non-Functional Requirements: Non-Functional Requirements, on the other hand, define the quality attributes of the system that do not relate directly to specific behaviors or functions. These requirements focus on the system's overall performance, reliability, scalability, security, and user experience.

A more detailed view of the system's data flow, illustrating the interactions and processes in greater depth, can be found in the Level 1 Data Flow Diagram presented in Figure 2.2, which further refines the functional requirements and processes.

3.2.1 Functional Requirements

- The system must accurately capture user input and extract relevant entities including `product_focus`, `favourite_color`, and `vibe`.
- The assistant must restrict its suggestions exclusively to products available in the active Voiceflow Knowledge Base, without inventing or hallucinating any items.

- The AI must respond with product recommendations in a structured JSON format that matches the schema defined in the Voiceflow prompt step.
- The system must detect when no matching products exist for the user's request and trigger a fallback message or themed suggestions in response.
- The system must ensure successful and consistent synchronization between Google Sheets (product catalog), n8n (data processing/API), Voiceflow (chatbot interface), and the frontend application (user interface).

3.2.2 Non Functional Requirements

- The assistant must maintain a natural and coherent conversational flow, avoiding repetitive or redundant questions during interaction.
- The system must ensure smooth real-time response generation and visually consistent product display within the user interface.
- The recommendations provided by the assistant should feel contextually relevant, stylistically appropriate, and personalized based on the user's stated preferences.
- The system must handle unexpected or unsupported user inputs gracefully, providing informative fallback responses without breaking the interaction flow.

3.3 System Architecture

The system architecture integrates multiple components to deliver a seamless and intelligent shopping experience. It consists of an AI-powered conversational agent, a dynamic product catalog, an automation middleware, and a user-facing frontend application. Each component plays a specific role and interacts with others to maintain real-time data flow and personalized customer interaction.

The main components of the architecture are:

- **Google Sheets**: Acts as the centralized database storing the product catalog, including essential details such as name, category, vibe, color, and occasion.
- **n8n**: Functions as the middleware platform, responsible for automating data synchronization between the database and other services. It also exposes APIs that allow external applications to access product data.
- **Voiceflow**: Serves as the conversational AI platform where the virtual fashion assistant is built. It captures user preferences, queries the Knowledge Base, and generates structured outfit suggestions.
- **Next.js Application**: Provides the frontend interface for users to browse products, interact with the virtual assistant, and complete their purchases.

The communication flow between components is structured as follows:

1. Product data is maintained and updated in Google Sheets.
2. n8n pulls the data from Google Sheets, formats the data, and uploads it to Voiceflow's Knowledge Base.
3. n8n also exposes product information through APIs, which are consumed by the Next.js application.
4. Voiceflow handles user interactions, processes input entities, and retrieves relevant product data from the Knowledge Base.
5. The Next.js frontend calls the APIs to retrieve and display specific product information dynamically.

This modular and service-oriented architecture ensures scalability, flexibility, and ease of maintenance, allowing future extensions such as integration with more complex databases or additional AI enhancements.

3.4 Modeling and Interaction Design

The user interaction with the AI-powered fashion assistant follows a structured conversational flow designed to capture preferences efficiently and provide personalized recommendations. The interaction model is based on entity recognition, knowledge base querying, and dynamic response generation.

The main stages of user interaction are:

- **Initial Input**: The assistant invites the user to freely describe what they are looking for, capturing general preferences such as style, colors, or specific needs.
- **Entity Capture**: The assistant detects key entities from the user's input, including:
  - *Product Focus*: (e.g., t-shirt, trousers, footwear, full outfit)
  - *Favorite Color*: (e.g., black, white, navy)
  - *Vibe or Occasion*: (e.g., casual, formal, sporty)
- **Clarification (if needed)**: If any required entity is missing or unclear, the assistant asks focused follow-up questions to complete the profile.
- **Knowledge Base Search**: Based on the collected entities, the assistant queries the internal product catalog via the Voiceflow Knowledge Base.
- **Suggestion Generation**: The assistant suggests individual products or complete outfits, providing explanations for each selection.

The dialogue flow is designed to minimize user effort and avoid repetitive questions. If the user provides complete information initially, redundant clarifications are skipped automatically, ensuring a smooth and efficient conversation.

3.5 Tasks and Actors

The AI shopping assistant operates based on a set of structured tasks, with different actors playing specific roles in the system's workflow as can be seen in 3.1.

3.5.1 Main Tasks

- **Initiate a Conversation with the Assistant**: The user begins an interaction with the AI-powered shopping assistant to start the personalized shopping experience.
- **Provide Style Input (e.g., vibe, color, product type)**: The user provides preferences such as style, color, and product type, which help the assistant generate personalized product recommendations.
- **View AI-Generated Product Suggestions**: Based on the user's input, the assistant presents relevant outfit suggestions tailored to the user's preferences, showcasing various products.
- **Add Item to Cart**: The user can add selected items to their shopping cart in preparation for checkout.
- **Customize Custom Look**: The user can modify an outfit by selecting individual products or altering existing suggestions to create a personalized look.
- **Extract Entities from User Input**: The Voiceflow Dialogue API processes the user input to extract important details like product type, color, and style preferences, which guide the recommendation process.
- **Execute Prompt and Knowledge Base (KB) Search Logic**: Voiceflow performs a knowledge base search using the extracted entities to find products that match the user's preferences.
- **Return Structured JSON Recommendations**: The Voiceflow Dialogue API sends back a structured JSON response containing the recommended products, formatted for display in the frontend.
- **Pull Product Data**: n8n retrieves the latest product data from Google Sheets, ensuring that the assistant has up-to-date information about the available products.

- **Upload Product Data to Voiceflow Knowledge Base**: n8n uploads the latest product details into the Voiceflow Knowledge Base to keep the catalog updated with new product information.
- **Expose API Endpoint to Retrieve Product Information**: n8n exposes API endpoints that allow the Next.js frontend to request detailed product data for display purposes.
- **Store Product Records**: Google Sheets serves as the central storage for all product data, maintaining detailed records of each item, including its attributes (e.g., name, type, color).
- **Send User Messages to the Voiceflow Dialogue API**: The frontend sends the user's input to the Voiceflow Dialogue API for processing and recommendation generation.
- **Render Product Suggestions from Voiceflow**: The frontend receives and renders the product suggestions returned by Voiceflow, displaying them to the user in an easy-to-navigate format.
- **Fetch Complete Product Data from n8n API**: The frontend queries the n8n API to fetch detailed product information (e.g., name, image, price) and display it dynamically within the user interface.

3.5.2 Primary Actors

- **User**: The online shopper interacting with the virtual assistant to find clothing items matching their preferences.
- **Voiceflow Platform**: The environment where the conversational flows are designed, entity extraction happens, and the knowledge base search is triggered.
- **n8n Automation Tool**: Acts as the bridge between the Google Sheets product database and the Voiceflow Knowledge Base, ensuring updated product data.

- **Google Sheets Database**: Serves as the central source of truth for product information (name, type, color, vibe, etc.).
- **Next.js Application**: Consumes Voiceflow API responses to integrate the assistant seamlessly into the shopping website's frontend.

The collaboration between these actors enables a smooth, dynamic shopping experience that aims to reduce user friction and increase conversion rates.

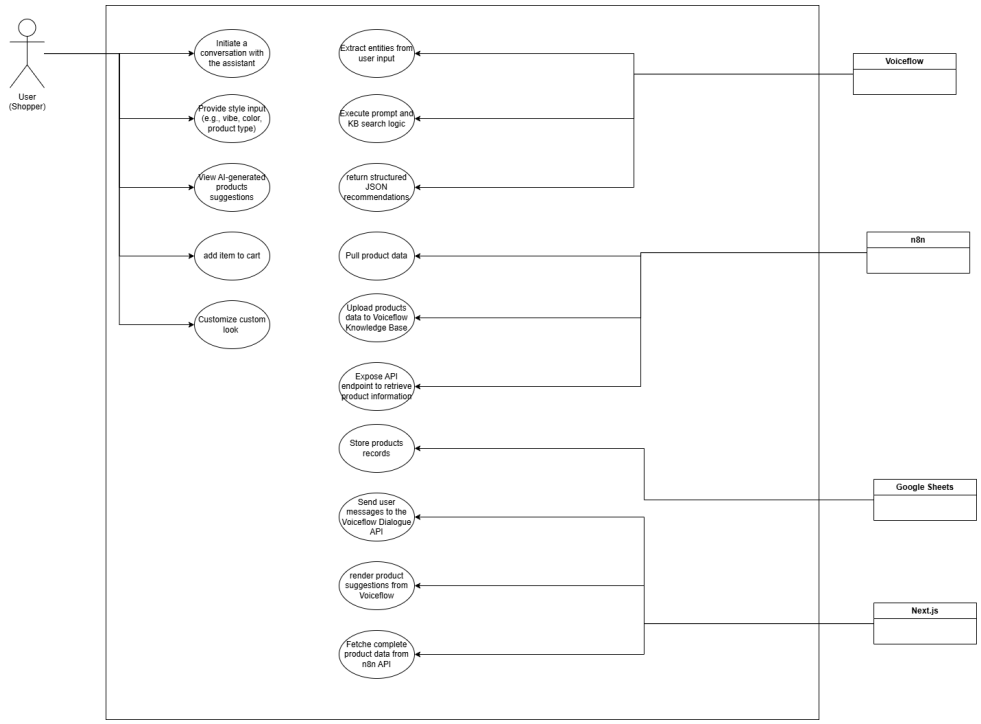

Figure 3.1: Use Case Diagram

Chapter 4

Development

4.1 System Architecture Overview

The AI-powered virtual fashion assistant system is composed of several interconnected components that work together to provide a seamless shopping experience. The architecture is modular, ensuring flexibility, scalability, and ease of maintenance.

The system is built around four main components as shown in 4.1:

- **Google Sheets Database**: Acts as the main product catalog, storing all clothing items and their associated attributes (e.g., name, type, color, vibe, occasion).
- **n8n Automation Server**: Handles the synchronization between the Google Sheets database and the Voiceflow Knowledge Base, and exposes API endpoints to external clients such as the Next.js application.
- **Voiceflow Platform**: Hosts the AI conversational assistant. It processes user input, captures entities, queries the Knowledge Base, and returns personalized product or outfit recommendations.
- **Next.js Frontend**: Provides the user interface where shoppers interact with the virtual assistant, browse products, and receive real-time recommendations.

Each component communicates using well-defined APIs and workflows, ensuring a smooth flow of information from the product catalog to the final user interaction.

A simplified view of the interaction between system components is as follows:

1. The product catalog is maintained and updated in Google Sheets.

2. n8n pulls data from Google Sheets and updates the Voiceflow Knowledge Base accordingly via API calls.

3. n8n also exposes product-related API endpoints that the Next.js application can query.

4. Users interact with the Voiceflow AI assistant through the Next.js application chat interface.

5. The assistant captures user preferences, queries the Knowledge Base, and suggests relevant products or outfits.

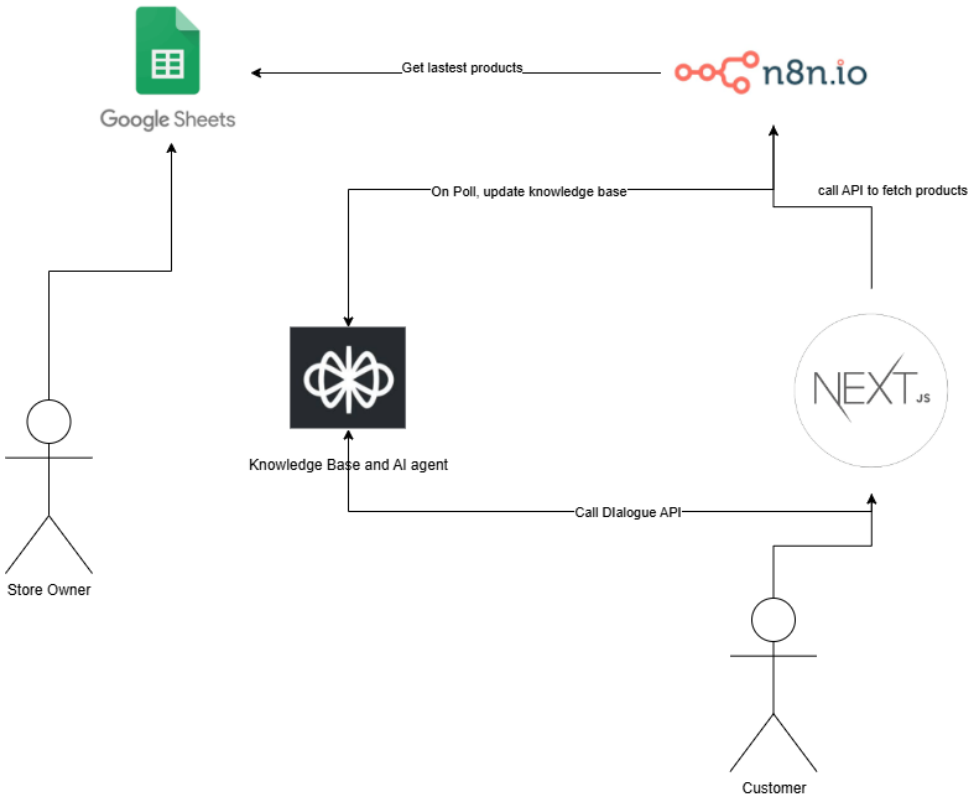

Figure 4.1: High-level system architecture of the AI-powered fashion assistant.

4.2 Database and Knowledge Base Setup

A reliable and easily maintainable product database is essential for the AI assistant to provide accurate and up-to-date recommendations. In this project, Google Sheets was selected as the primary database, while Voiceflow's Knowledge Base (KB) was used to enable efficient querying within the conversational assistant.

Google Sheets was chosen for its simplicity, accessibility, and collaborative editing capabilities. Each row in the spreadsheet represents a product, with columns capturing relevant attributes such as:

- Product ID
- Name
- Type (top, bottom, footwear)

- Category (e.g., t-shirt, trousers)
- Color
- Vibe or Occasion
- URL

The spreadsheet acts as the single source of truth for the product catalog, ensuring that updates are straightforward and transparent.

To automate the synchronization between Google Sheets and Voiceflow's Knowledge Base, an n8n workflow was created with the following steps:

1. **Trigger**: The workflow is triggered manually or on a schedule to check for updates in the Google Sheets document.
2. **Data Retrieval**: n8n reads all product rows from the specified Google Sheet.
3. **Data Transformation**: The retrieved rows are formatted into a structured JSON format compatible with the Voiceflow KB upload API.
4. **API Upload**: The structured data is sent to the Voiceflow Knowledge Base using a POST request to the Upload Table API.

This automation ensures that the assistant always operates on the most recent version of the product catalog without requiring manual intervention.

The Voiceflow Knowledge Base is populated by sending JSON-formatted product data through their Upload Table API. Each document in the KB corresponds to a fashion product, indexed by key attributes such as vibe, type, and color.

This setup enables fast, accurate retrieval of products that match the entities captured from user input, allowing the AI to suggest appropriate options dynamically.

4.3 Voiceflow Conversational Design

Voiceflow was used as the platform to design the conversational flows of the virtual fashion assistant. The assistant is built to capture user input efficiently, recognize key entities, and generate structured recommendations based on the available products.

The conversational assistant is designed to detect and capture the following main entities from user input:

- **Product Focus**: Whether the user is looking for a specific item such as a t-shirt, trousers, footwear, or a complete look.
- **Favorite Color**: Specific color preferences to personalize suggestions.
- **Vibe or Occasion**: The style, feeling, or context the user wants the outfit for (e.g., casual, sporty, elegant).

Entity capture occurs naturally during the conversation. If the user provides all necessary details upfront, no redundant questions are asked. Otherwise, targeted clarification questions are triggered to complete the information gathering process.

Special care was taken to design the system prompts that instruct the AI how to behave. The prompt explicitly states that:

- Only products from the available Knowledge Base must be used.
- Hallucination of product names or IDs is not allowed.
- Products must be selected based on vibe, color compatibility, and formality level.
- The output must strictly follow a JSON structure with clear explanations for each recommendation.

This careful prompt design helps guide the AI's generation of realistic and useful responses while maintaining structure.

The conversation flow includes specific fallback mechanisms to handle situations where:

- The Knowledge Base does not return any relevant products (e.g., the user asks for a type of product not available).
- The AI cannot confidently generate matching recommendations due to missing input.

In such cases, instead of returning an error or an empty result, the assistant returns a friendly fallback message suggesting alternative available styles, maintaining a positive user experience.

4.4 Next.js Web Application Integration

The Next.js framework was chosen to build the frontend interface through which users interact with the AI-powered fashion assistant. The integration between the web application and the conversational assistant ensures a seamless and dynamic shopping experience.

Communication between the Next.js application and the Voiceflow assistant is established using Voiceflow's API. The main steps are:

- **Session Creation**: When a user starts a chat session, a new conversation session is initialized via a POST request to Voiceflow's API.
- **Message Exchange**: User inputs are sent to the Voiceflow API, which returns structured responses containing either outfit recommendations or fallback messages.

This API-based communication allows the frontend to act as a thin, responsive client while the heavy conversational logic is handled by Voiceflow servers.

In addition to interacting with Voiceflow, the Next.js application also connects to API endpoints exposed by the n8n server to retrieve detailed product information when needed. The flow includes:

- **Product Lookup**: When the assistant suggests a product by ID, the Next.js app queries the corresponding n8n API to fetch complete product details (name, image, URL, etc.).

- **Dynamic Rendering**: Retrieved product data is dynamically rendered into the UI in a way which appeals to the customer and still allows him to create his own custom look.

To create a smooth user experience, the application implements real-time updates:

- Messages are displayed immediately after user input to maintain conversation flow.

- Loading indicators are shown while awaiting API responses to manage user expectations.

- Product cards, outfit suggestions, and fallback messages are styled consistently to ensure visual coherence.

Thanks to this architecture, users experience fast, responsive interactions, making the assistant feel natural and engaging.

The user interface of the web application was developed using Next.js and designed with a focus on simplicity, clarity, and conversion efficiency. One of the key goals of the interface was to reduce the friction that often causes users to abandon their shopping journey.

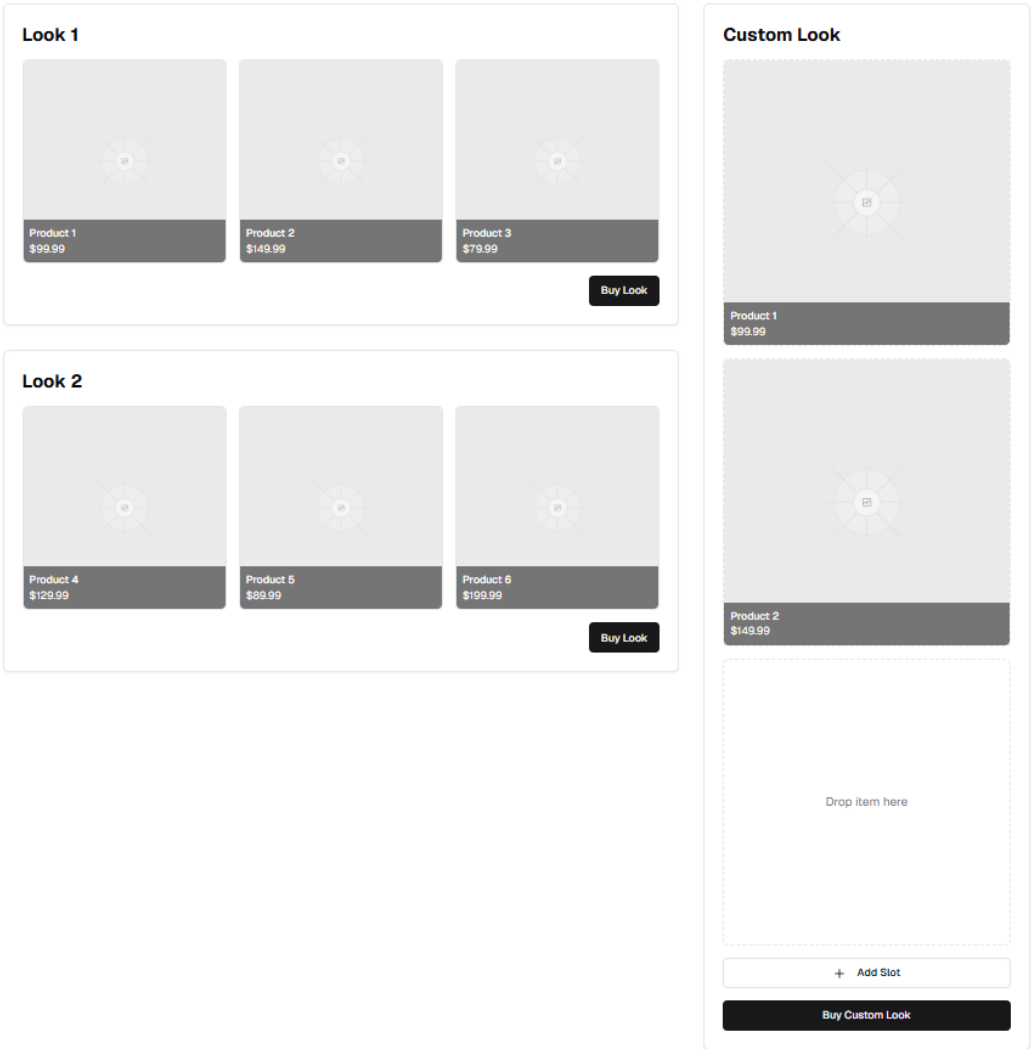

Figure 4.2: Look suggestion and custom outfit creation interface

As shown in Figure 4.2, the main UI consists of two primary areas:

- **Suggested Looks**: Displayed on the left, the assistant-generated outfit suggestions (Look 1, Look 2, etc.) are visually grouped with top, bottom, and footwear products. Each look includes product thumbnails, prices, and a "Buy Look" button to streamline checkout for users who want a complete look quickly.
- **Custom Look Builder**: On the right, users are provided with an interactive space where they can drag and drop individual products from the suggestions or the general catalog to build their own outfit. They can also add slots to extend the look

and press the “Buy Custom Look” button when ready. This gives users creative freedom while maintaining ease of use.

The UI design strikes a balance between guided discovery (through AI recommendations) and manual control (via customization). This approach aligns well with the project’s goal of minimizing decision fatigue while promoting engagement and increasing the likelihood of purchase completion.

The overall UX has been crafted to support both inspiration-driven shoppers and goal-oriented users, offering an intuitive, interactive, and enjoyable experience throughout the shopping process.

4.5 Workflow Automation with n8n

n8n was used as the automation platform to orchestrate data synchronization between the Google Sheets product catalog, the Voiceflow Knowledge Base, and the external APIs consumed by the Next.js application.

An integration was established between n8n and Google Sheets to allow the workflow to monitor changes in the product catalog, where n8n periodically retrieves the latest rows from the designated Google Sheet containing product information.

This integration makes it easy for non-technical team members to update the product database without needing direct access to code or server infrastructure.

n8n was also used to expose custom API endpoints that the Next.js frontend can call to retrieve product details. Specifically:

- **Webhook Setup**: Webhooks were configured to listen for incoming HTTP requests containing parameters such as product IDs.
- **Dynamic Data Filtering**: Upon receiving a request, the workflow dynamically filters the product list to match the requested ID(s).
- **Response Formatting**: The matching product(s) are returned as structured JSON responses suitable for frontend consumption.

This allows the Next.js application to remain lightweight while retrieving detailed product data dynamically as needed.

A separate workflow was designed specifically for updating the Voiceflow Knowledge Base. The process follows these steps:

1. Fetch product data from Google Sheets.
2. Format the data into the JSON structure required by Voiceflow's Upload Table API.
3. Authenticate and send a POST request to upload the data into the Knowledge Base.

Automating this process guarantees that Voiceflow always operates on up-to-date product information, eliminating the need for manual uploads and reducing the risk of inconsistency.

4.6 Key Technical Challenges and Solutions

During the development and integration of the system components, several technical challenges emerged. Addressing these challenges was crucial to ensure the robustness, efficiency, and usability of the final solution.

Challenge: Keeping the Google Sheets database, the Voiceflow Knowledge Base, and the n8n API endpoints in sync without manual intervention.

Solution: Automated workflows in n8n were implemented to periodically fetch the latest Google Sheets data, format it appropriately, and upload it directly to the Voiceflow Knowledge Base. Webhooks ensured that product details accessed by the Next.js frontend were always current and consistent with the main database.

Challenge: Ensuring that Voiceflow always returned outputs strictly following a predefined JSON structure, especially when suggesting outfits based on user preferences.

Solution: Detailed system prompts and structured output schemas were used inside Voiceflow. These strictly instructed the AI to format responses correctly, improving the reliability of processing and display.

Challenge: Users could express preferences in many different ways, making it difficult to consistently capture entities such as color, vibe, or product type.

Solution: Voiceflow entity models were enriched with synonyms and natural language variations. The conversational flow was also designed to allow clarification questions when the input was ambiguous, ensuring higher entity capture rates.

Challenge: In cases where the user requested unavailable products or obscure styles, the system needed to maintain a positive user experience without returning errors or dead ends.

Solution: Fallback strategies were implemented. If no matching products were found, the assistant responded with a friendly message suggesting the closest available alternatives, keeping the user engaged and guiding them back into the selection process.

Challenge: Generating outfit suggestions that feel cohesive (matching vibe, color, and formality) required fine-tuning the AI's logic beyond simply returning random products.

Solution: Explicit rules and prompt engineering were applied inside Voiceflow to instruct the AI on how to match items appropriately, resulting in more natural and stylish recommendations.

Chapter 5

Tests and Discussion

5.1 Testing Objectives

The testing phase aimed to ensure that the AI-powered fashion assistant met both its functional requirements and user experience expectations. Specific goals were defined to guide the testing efforts:

Verification of Functional Requirements

The primary objective was to verify that each component of the system performed its intended function correctly. This included:

- Ensuring accurate capture of user input and entity extraction (product focus, favorite color, vibe).
- Confirming that the assistant only suggested products from the current Knowledge Base.
- Verifying that the AI-generated recommendations followed the specified structured JSON output.
- Validating that the fallback mechanisms operated correctly when no suitable products were found.

- Checking the successful synchronization between Google Sheets, n8n, Voiceflow, and the frontend application.

Validation of User Experience Goals

In addition to functional correctness, the testing also focused on validating the quality of the user experience:

- Ensuring that conversations flowed naturally without forcing repetitive or redundant questions.
- Confirming that real-time responses and product displays were smooth and visually coherent.
- Assessing whether the recommendations felt relevant, stylish, and personalized based on the user inputs.
- Evaluating how the system handled unexpected inputs or unsupported queries gracefully.

Achieving these objectives was critical to delivering a solution that not only worked technically but also provided a pleasant and effective shopping experience for users.

5.2 Testing Methodology

The testing process combined manual testing of conversational flows with targeted technical validation of system integrations and outputs. This approach ensured comprehensive coverage of both functional and user experience requirements.

Manual Test Scenarios

Manual testing was employed to simulate real-world interactions between users and the virtual assistant. The steps involved:

- Engaging in natural language conversations to verify if the assistant accurately captured entities such as product focus, favorite color, and vibe.
- Testing various input styles, including short phrases, long descriptions, slang, and typos, to assess the assistant's flexibility.
- Validating that the suggested products aligned with the captured preferences and that the generated recommendations respected the structured JSON output.
- Intentionally requesting unavailable products (e.g., "swimwear") to verify fallback messaging and graceful handling of unsupported requests.

Manual testing was critical to identify edge cases and conversational flow improvements that would not easily emerge through automated means.

5.3 Test Cases and Results

A series of targeted tests were conducted to verify the correct functioning of key system components. The following subsections describe specific test cases, their expected outcomes, and the observed results.

Entity Recognition Accuracy Tests

Objective: Validate that the conversational assistant correctly captures entities such as product focus, favorite color, and vibe from user input.

Test Scenario: Input various phrases describing different styles, colors, and occasions.

Expected Result: Entities should be correctly extracted without asking unnecessary follow-up questions.

Outcome: entities were successfully recognized without further clarification. Slight confusion occurred when the information was totally missing, which triggered clarification questions as intended.

Knowledge Base Search Tests

Objective: Ensure that the Knowledge Base search returns only products relevant to the user's captured preferences.

Test Scenario: Provide specific vibes or color preferences and verify the recommended products.

Expected Result: Returned products should match the requested vibe and/or color.

Outcome: Matching worked correctly in most cases. When no matching products existed, fallback messages were correctly triggered, offering close alternatives or polite apology messages.

Product Recommendation Logic Tests

Objective: Verify that the assistant generates complete outfits based on available products and matches them for style and formality.

Test Scenario: Request a casual outfit for a summer vibe.

Expected Result: The assistant suggests a cohesive outfit: top, bottom, and footwear matching the casual summer theme.

Outcome: Generated outfits were coherent and appropriate. In rare cases, color matching was slightly less ideal but remained acceptable.

Next.js Application Integration Tests

Objective: Confirm that product information retrieved via n8n Webhook APIs is correctly displayed in the frontend.

Test Scenario: Request an outfit and verify that product images, names, and URLs are correctly rendered in the chat interface.

Expected Result: All product details are displayed dynamically and formatted properly in the user interface.

Outcome: Integration was fully successful. All recommended products appeared with the correct metadata and navigation links, ensuring a seamless browsing experience for

users.

5.4 Analysis and Discussion

This section analyzes the results obtained during the testing phase, highlights key learnings, and discusses opportunities for future improvement.

What Worked Well

The combination of Voiceflow for conversational logic, n8n for backend automation, and Next.js for frontend integration resulted in a robust and modular system. Key achievements include:

- High accuracy in entity recognition thanks to carefully designed entities and synonyms in Voiceflow.
- Seamless synchronization between the Google Sheets database and the Voiceflow Knowledge Base via automated workflows.
- Consistent generation of structured JSON responses, enabling predictable frontend behavior and easier error handling.
- Positive user experience, with smooth conversational flow and visually appealing product displays.

Challenges Encountered During Testing

Several challenges were encountered during the development and testing phases:

- Ambiguous user input occasionally made entity recognition less reliable, requiring fallback strategies and clarification prompts.
- Handling edge cases where the Knowledge Base lacked suitable matches required careful fallback message design to preserve a positive experience.

Lessons Learned

Important lessons emerged from the development and testing process:

- A centralized and easily editable product catalog (Google Sheets) significantly simplified database maintenance.
- Structured output enforcement at the conversational level improved both AI behavior and integration stability.
- Flexibility in conversational flows—allowing clarification only when necessary—greatly enhances user satisfaction.

Improvements for Future Iterations

Based on the observations during testing, several improvements could be implemented in future versions:

- Integrating more advanced NLP techniques or models to improve entity capture even further for ambiguous inputs.
- Expanding the product catalog to include more categories such as accessories or jackets, enriching outfit possibilities.
- Adding user preference memory across sessions to offer even more personalized suggestions over time.

Chapter 6

Conclusions

This project successfully designed and implemented an AI-powered virtual fashion assistant capable of guiding online shoppers in selecting personalized outfits. The major accomplishments include:

- Building a conversational AI using Voiceflow capable of understanding user preferences in terms of product type, colors, and style (vibe or occasion).
- Setting up a centralized Google Sheets database for product catalog management, enabling easy updates and scalability.
- Automating data synchronization between the database and Voiceflow's Knowledge Base using n8n workflows.
- Creating a Next.js frontend application that dynamically communicates with the Voiceflow assistant and n8n APIs to retrieve and display product information in real-time.
- Ensuring that the assistant reliably outputs structured JSON responses for seamless frontend rendering.

These achievements resulted in an interactive, personalized, and efficient shopping experience aimed at reducing decision fatigue and improving checkout conversion rates.

Throughout the project, several design choices proved particularly effective:

- Using modular tools (Voiceflow, n8n, Google Sheets, Next.js) allowed flexibility, rapid iteration, and scalability without introducing unnecessary complexity.
- Structuring the conversation around natural entity capture, with fallback handling, led to fluid user experiences even when inputs were incomplete or ambiguous.
- Maintaining strict adherence to structured JSON outputs improved integration stability between conversational AI and frontend systems.

However, the project also exposed areas where more sophisticated techniques (such as advanced natural language processing) could have further enhanced accuracy and personalization.

Despite its success, the project has some limitations:

- The AI relies heavily on predefined synonyms and patterns; it may struggle with highly unusual or out-of-scope user inputs.
- The system currently handles recommendations based on a single interaction, without long-term memory of users' historical preferences.
- The product catalog scope is limited to tops, bottoms, and footwear, excluding accessories or broader fashion categories.
- While n8n automation is robust, it introduces some dependency on external workflow stability and Google Sheets API availability.

Recognizing these limitations provides a clear pathway for future enhancements.

6.1 Future Development Directions

Several improvements and expansions are planned for future versions of the system:

- **Expanded Product Categories:** Including accessories (e.g., hats, bags) to offer users full styling recommendations.
- **User Profile Memory:** Implementing persistent profiles to remember favorite colors, sizes, and preferred styles across sessions.
- **Advanced AI Models:** Incorporating more sophisticated NLP models to better interpret free-form user input and suggest products even more accurately.
- **Backend Upgrade:** Migrating from Google Sheets to a more scalable database (e.g., Firebase, PostgreSQL) to handle larger product inventories and complex queries.

These improvements would make the assistant more intelligent, more scalable, and capable of delivering an even richer personalized shopping experience.

Bibliography

[1] G. Tur and R. De Mori, "Spoken language understanding: Systems for extracting semantic information from speech," *John Wiley & Sons*, 2011.

[2] G. Mesnil *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[3] S. Young *et al.*, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[4] S. Weerawarana, E. N. Houstis, J. R. Rice, A. Joshi, and C. E. Houstis, "PYTHIA: A knowledge-based system to select scientific algorithms," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 447–468, Dec. 1996. DOI: 10.1145/235815.235820.

[5] U. Şimşek and D. Fensel, "Intent generation for goal-oriented dialogue systems based on schema.org annotations," *arXiv preprint arXiv:1807.01292*, 2018.

[6] X. Li, Z. Liu, C. Xiong, *et al.*, "Structure-aware language model pretraining improves dense retrieval on structured data," in *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada: Association for Computational Linguistics, 2023, pp. 11 560–11 574. DOI: 10.18653/v1/2023.findings-acl.734.

[7] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 373–383, 2020.

[8] R. A. Carter, L. Zhang, T. L. Hunt, *et al.*, "Conversational agents to support remote personalized instruction for diverse learners," *TechTrends*, vol. 67, pp. 626–636, 2023. DOI: 10.1007/s11528-023-00877-3.

[9] A. Følstad, M. Skjuve, and P. B. Brandtzaeg, "Different chatbots for different purposes: Towards a typology of chatbots to understand interaction design," in *Internet Science. INSCI 2018*, ser. Lecture Notes in Computer Science, vol. 11551, Springer, Cham, 2019. DOI: 10.1007/978-3-030-17705-8_13.

[10] S. McInerney, T. Nash, R. Lee, M. Falis, F. Gruber, and A. Casey, "Ai chatbot for cancer patient support: Development and evaluation using llama 3.1, mistral 7b, and phi 3b," *Studies in Health Technology and Informatics*, vol. 327, pp. 890–891, May 2025. DOI: 10.3233/SHTI250494.

[11] E. Grassini, M. Buzzi, B. Leporini, *et al.*, "A systematic review of chatbots in inclusive healthcare: Insights from the last 5 years," *Universal Access in the Information Society*, vol. 24, pp. 195–203, 2025. DOI: 10.1007/s10209-024-01118-x.

[12] G. Park, J. Chung, and S. Lee, "Effect of ai chatbot emotional disclosure on user satisfaction and reuse intention for mental health counseling: A serial mediation model," *Current Psychology*, vol. 42, pp. 28 663–28 673, 2023. DOI: 10.1007/s12144-022-03932-z.

[13] M. Rheu, J. Y. Shin, W. Peng, and J. Huh-Yoo, "Systematic review: Trust-building factors and implications for conversational agent design," *International Journal of Human–Computer Interaction*, vol. 37, no. 1, pp. 81–96, 2020. DOI: 10.1080/10447318.2020.1807710.

[14] S. Liang, R. Li, B. Lan, Y. Chu, M. Zhang, and L. Li, "Untouchable them: The effect of chatbot gender on angry customers," *Journal of Research in Interactive Marketing*, vol. 18, no. 6, pp. 1099–1135, 2024. DOI: 10.1108/JRIM-02-2023-0061.

[15] J. Fu, S.-K. Ng, and P. Liu, "Polyglot prompt: Multilingual multitask prompt training," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, pp. 9919–9935. DOI: 10.18653/v1/2022.emnlp-main.674.

[16] Y. Xiang, T. Zhang, H. Di, *et al.*, "Improving zero-shot cross-lingual dialogue state tracking via contrastive learning," in *Chinese Computational Linguistics: 22nd China National Conference, CCL 2023, Harbin, China, August 3–5, 2023, Proceedings*, Berlin, Heidelberg: Springer-Verlag, 2023, pp. 127–141. DOI: 10.1007/978-981-99-6207-5_8.

[17] Voiceflow, *Voiceflow documentation*, https://docs.voiceflow.com/, 2024.

[18] T. Bocklisch *et al.*, "Rasa: Open source language understanding and dialogue management," *arXiv preprint arXiv:1712.05181*, 2017.

[19] G. Cloud, *Dialogflow documentation*, https://cloud.google.com/dialogflow/docs, 2024.

[20] P. M. Gomes and M. A. Brito, "Low-code development platforms: A descriptive study," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, Madrid, Spain, 2022, pp. 1–4. DOI: 10.23919/CISTI54924.2022.9820354.

[21] G. S. Guaki and G. P. Genove, "A literature review on low code and no code (lcnc) platforms in reshaping web and application development," in *AIP Conference Proceedings*, vol. 3287, Apr. 2025, p. 030 021. DOI: 10.1063/5.0262017.

[22] T. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[23] OpenAI, *Gpt-4 technical report*, https://openai.com/research/gpt-4, 2023.

[24] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[25] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[26] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[27] S. H. D. Nguyen, T. D. Trinh, and Q. V. Q. Tran, "Multi-agent chatbot for efficient interaction with blockchain apis," in *Information and Communication Technology. SOICT 2024*, ser. Communications in Computer and Information Science, vol. 2352, Springer, Singapore, 2025. DOI: 10.1007/978-981-96-4288-5_33.

[28] Rahul, *Jsonformer: A bulletproof way to constrain llms to generate json outputs*, https://github.com/1rgs/jsonformer, [Computer software], 2023.

[29] S. Zhang *et al.*, "Personalizing dialogue agents: I have a dog, do you have pets too?" *ACL*, 2018.

[30] S. Liu, H. Cho, M. Freedman, X. Ma, and J. May, "Recap: Retrieval-enhanced context-aware prefix encoder for personalized dialogue response generation," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada: Association for Computational Linguistics, 2023, pp. 8404–8419. DOI: 10.18653/v1/2023.acl-long.468.

[31] M. M. Rahsepar, T. Sillekens, S. Metselaar, A. van Balkom, J. Bernstein, and N. Batelaan, "Exploring the ethical challenges of conversational ai in mental health care: Scoping review," *JMIR Mental Health*, 2025. DOI: 10.2196/60432.

[32] L. Weidinger *et al.*, "Ethical and social risks of harm from language models," *arXiv preprint arXiv:2112.04359*, 2021.

[33] A. Zadeh *et al.*, "Tensor fusion network for multimodal sentiment analysis," *EMNLP*, 2017.

[34] C. Liu *et al.*, "M3it: A large-scale dataset for multilingual multimodal instruction tuning," *arXiv preprint arXiv:2306.04387*, 2023.

[35] P. Prabhu, S. Dammu, O. Alonso, and B. Poblete, "A shopping agent for addressing subjective product needs," in *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25)*, New York, NY, USA: Association for Computing Machinery, 2025, pp. 1032–1035. DOI: 10.1145/3701551.3704124.

[36] M. Eric *et al.*, "Multiwoz 2.1: A consolidated multi-domain dataset for task-oriented dialogue modeling," *LREC*, 2020.