UAM
Universidad Autónoma de Madrid

Escuela Politécnica Superior

24
25

Master thesis

A Hybrid Conversational Recommender System by integrating LLMs

Javier Wang Zhou

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Master in ICT Research & Innovation

MASTER THESIS

A Hybrid Conversational Recommender System by integrating LLMs

Conversation Design & User Profiling for Explainable Recommendations

Author: Javier Wang Zhou
Advisor: Alejandro Bellogín Kouki

septiembre 2025

Javier Wang Zhou
A Hybrid Conversational Recommender System by integrating LLMs

Javier Wang Zhou
C\ Francisco Tomás y Valiente Nº 11

PRINTED IN SPAIN

To my family & friends

Without data, you are just another person with an opinion.

Andreas Schleicher

Agradecimientos

Quisiera agradecer en primer lugar a mi tutor Alejandro Bellogín, por su apoyo y orientación durante este tiempo. Su experiencia y dedicación me han motivado a sacar adelante este trabajo conjunto.

También quiero dar las gracias a todos mis compañeros del Máster, con quienes he compartido estos dos años en los que cada tropiezo me ha permitido aprender y mejorar. Entre cafés en la cafetería de la Escuela y quedadas fuera de ésta, hemos crecido juntos tanto académica como personalmente.

Finalmente, gracias a mi madre por estar siempre ahí, apoyándome en todo momento, y a mi perro Cookie por alegrarme los días que más le he necesitado.

Mi más sincero agradecimiento a todos.

Resumen

Si bien los *Large Language Models (LLMs)* han creado nuevos paradigmas para la interacción persona-ordenador, su aplicación en los Sistemas de Recomendación presenta importantes desafíos de investigación. Los sistemas tradicionales a menudo no logran capturar los matices de las preferencias del usuario, y la naturaleza de “caja negra” de muchos modelos erosiona su confianza. Se necesitan de Sistemas Conversacionales capaces de obtener preferencias del usuario dinámicamente, construir perfiles de usuario completos y ofrecer recomendaciones transparentes y explicables.

Este Trabajo de Fin de Máster aborda estos desafíos proponiendo y evaluando un marco híbrido para la recomendación conversacional explicable. El principal objetivo de investigación es indagar cómo una combinación de interacción conversacional, perfilado dinámico del usuario y razonamiento basado en grafos puede conducir a recomendaciones más eficaces, transparentes y centradas en el usuario.

Para ello, se diseñó un Sistema de Recomendación Conversacional híbrido, cuyo flujo conversacional es orquestado por un *workflow* de LlamaIndex. Los aspectos de desarrollo web del sistema se detallan en la tesis complementaria de ingeniería. Durante la interacción, se construye un perfil de usuario dinámico capturando preferencias explícitas e implícitas. Este perfil informa una estrategia de recomendación híbrida que utiliza tanto un motor basado en grafos en tiempo real usando `FalkorDB` para recomendaciones contextuales y explicables, como un modelo experto preentrenado (`EASER`) para una personalización profunda. Una contribución de este trabajo es el uso de razonamiento basado en grafos para generar explicaciones en lenguaje natural que justifiquen las sugerencias del sistema. Además, la plataforma fue evaluada empíricamente mediante un estudio de usabilidad con 11 participantes para medir la precisión de las recomendaciones, la satisfacción del usuario, el éxito en las tareas y la usabilidad percibida a través de la *System Usability Scale (SUS)*.

Los resultados son prometedores e indican la viabilidad del enfoque híbrido propuesto. El sistema alcanzó una puntuación SUS de 92 sobre 100, y los comentarios cualitativos de los usuarios destacaron la intuitividad de la interacción conversacional y la relevancia de las recomendaciones. Este TFM aporta un marco validado para la construcción de agentes de recomendación explicables, demostrando que la integración del perfilado dinámico del usuario y la generación de explicaciones basadas en grafos es un paso hacia la creación de Sistemas de Recomendación más fiables y eficaces.

Palabras clave

Sistema de Recomendación Conversacional, IA Explicable, Perfilado de Usuario, Modelos Extensos de Lenguaje, Interacción Persona-Ordenador, Recomendación Basada en Grafos, Tests de Usabilidad

Abstract

While Large Language Models (LLMs) have created new paradigms for human-computer interaction, their application in Recommender Systems (RSs) presents significant research challenges. Traditional systems often fail to capture the nuances of user preferences, and the black-box nature of many models erodes user trust. There is a pressing need for Conversational Recommender Systems that can dynamically elicit user preferences, build comprehensive user profiles, and provide transparent, explainable recommendations.

This Master's Thesis addresses these challenges by proposing and evaluating a novel hybrid framework for explainable conversational recommendation. The primary research objective is to investigate how a combination of conversational interaction, dynamic user profiling, and graph-based reasoning can lead to more effective, transparent, and user-centric recommendation experiences.

To this end, a hybrid Conversational Recommender System was designed and implemented, whose core is a LlamaIndex workflow that orchestrates the conversational flow. The web development aspects of the system are detailed in the complementary engineering thesis. During the interaction, the system dynamically constructs a user profile by capturing explicit and implicit preferences. This profile informs a hybrid recommendation strategy that leverages both a real-time, graph-based engine using `FalkorDB` for contextual and explainable recommendations, and a pre-trained expert model (`EASER`) for deep personalization. A key contribution of this work is the use of graph reasoning techniques to generate natural language explanations that justify the system's suggestions. Moreover, the platform was empirically evaluated through a usability study involving 11 participants to assess user satisfaction, task success, and perceived usability via the System Usability Scale (SUS).

The results are promising, indicating the viability of the proposed hybrid approach. The system achieved a SUS score of 92 out of 100, and qualitative feedback from users highlighted the intuitiveness of the conversational interaction and the relevance of the recommendations. This thesis contributes a validated framework for building explainable recommender agents, demonstrating that the integration of dynamic user profiling and graph-based explanation generation is a step towards creating more trustworthy and effective Recommender Systems.

Keywords

Conversational Recommender System, Explainable AI, User Profiling, Large Language Models, Human-Computer Interaction, Graph-Based Recommendation, Usability Testing

Table of Contents

Lists

List of codes

List of equations

List of figures

List of tables

1 Introduction

The advent of powerful Large Language Models has opened new frontiers in human-computer interaction, particularly within the domain of Recommender Systems. Moving beyond traditional static interfaces, Conversational Recommender Systems (CRSs) promise a more natural and intuitive way for users to discover items that align with their preferences. This chapter lays the foundation for the research presented in this thesis. It begins by establishing the motivation for developing a novel framework for conversational recommendations, focusing on the research gaps in user profiling and explainability. Subsequently, it formulates the primary research questions that this work aims to answer and concludes by outlining the structure of the document.

1.1. Motivation

The application of LLMs to Recommender Systems is a novel and rapidly evolving field of research. While early CRSs often relied on rigid, rule-based systems [1], the fluidity of LLMs allows for a much richer dialogue. However, the use of LLMs in this domain is still a relatively underexplored area with significant challenges [2, 3]. Modern techniques such as Retrieval-Augmented Generation (RAG) and Function Calling are often employed to contextualize the model's responses and connect them to external tools.

A pivotal research gap lies in how to effectively translate a free-form conversation into a structured understanding of a user's tastes. This requires robust methods for dynamic user profiling and preference elicitation [4, Conversational Preference Elicitation]. Furthermore, as recommendation models become more complex, they often become "black boxes", making it difficult for users to understand why a particular item was suggested. This lack of transparency can hinder user trust and adoption. The field of Explainable Artificial Intelligence (XAI) has become paramount in addressing this issue, with a strong focus on generating justifications for model outputs, a task for which LLMs are particularly well-suited [4, 5, Generating Textual Explanations].

This thesis is motivated by the need to address these specific research challenges. The work focuses on designing and evaluating a system that not only converses naturally but also actively builds a user profile from the dialogue and explains its recommendations.

1.2. Research Objectives

This research aims to investigate a novel framework for a CRS that prioritizes user understanding and transparency—whereas the complementary engineering thesis implements a conversational platform to manage and interact with recommender agents from any dataset [6], using the algorithms designed in this work. The investigation is guided by the following primary research questions:

RQ1: To what extent can dynamic user profiles, constructed from natural language conversations, improve the quality and personalization of recommendations in a CRS?

RQ2: What is an effective conversational strategy for integrating different recommendation approaches to address both cold-start and warm-start user scenarios?

RQ3: How does the integration of natural language explanations, generated from the underlying recommendation logic, influence user trust and the overall interactive experience in a CRS?

1.3. Work Structure

This thesis is organized into five chapters, structured to logically present the research from its conceptualization to its final conclusions.

Chapter 1 — Introduction This chapter introduces the research topic, outlines the problem statement, and presents the research questions guiding the study.

Chapter 2 — State of the Art This chapter provides a detailed review of the academic literature and technologies relevant to this work, with a focus on LLMs, CRSs, and explainability in Artificial Intelligence (AI).

Chapter 3 — Methodology This chapter presents the core methodological framework developed to address the research questions. It details the design of the conversation workflow, the user profiling model, the hybrid recommendation strategy, and the graph-based explanation generation method.

Chapter 4 — Experiments and Results This chapter describes the experimental setup, including the datasets and evaluation protocols. It then presents and analyzes the results from the comprehensive usability study conducted to evaluate the platform.

Chapter 5 — Conclusions and Future Work The final chapter summarizes the key findings of the research, discusses their implications in relation to the initial research questions, acknowledges limitations, and proposes potential directions for future work.

2 State of the Art

This chapter provides a research-focused review of the foundational concepts and technologies that underpin this thesis. The objective is to survey the current academic and technical landscape in order to situate the project's contributions and justify the methodological choices made. The analysis is structured into four primary domains. It begins with an in-depth look at Large Language Models and the main techniques used to augment their capabilities. It then reviews the field of Recommender Systems and the specialized sub-field of Conversational Recommender Systems. Finally, it delves into the paramount topic of explainability in recommendation, a central theme of this research.

2.1. Large Language Models

The enabling technology for modern conversational AI is the Large Language Model, a class of deep learning models based on the Transformer architecture [7]. These models have demonstrated a powerful ability to understand context, reason, and generate fluent natural language; with popular implementations such as ChatGPT [8], Google Gemini [9] or Claude [10]. However, when used in isolation, they suffer from several inherent limitations, including knowledge being confined to their static training data (a *knowledge cut-off*) and a propensity to generate plausible but factually incorrect information (*hallucinations*) [11]. To build robust and reliable applications, it is necessary to augment these models with external knowledge and capabilities. The following subsections detail the state-of-the-art frameworks and techniques used to achieve this.

2.1.1. LLM Frameworks

The complexity of building applications on top of LLMs has led to the development of powerful orchestration frameworks. These frameworks provide abstractions that simplify common tasks such as managing prompts, connecting to data sources, and chaining together multiple LLM calls and tool invocations. Some of most the dominant open-source frameworks in this space are `LangChain` [12], `LlamaIndex` [13], and `Haystack` [14].

While all three serve a similar purpose, they have different core philosophies. `LangChain` is a general-purpose framework designed for maximum flexibility in creating complex, custom agentic workflows. `LlamaIndex`, conversely, is a data-centric framework that specializes in the RAG pipeline, offering a rich set of tools for data ingestion, indexing, and retrieval. `Haystack` is geared towards building production-ready, scalable search agents, with an explicit focus on deployable pipelines. Ultimately, this project leverages `LlamaIndex` for its state-of-the-art, data-centric capabilities.

2.1.2. Retrieval-Augmented Generation

RAG has emerged as the principal architectural pattern to mitigate the inherent limitations of standalone LLMs [11]. While LLMs offer powerful reasoning and language generation capabilities, their reliance on static, internal knowledge leads to several challenges for recommendation tasks:

- **Knowledge cut-off:** The model is unaware of any items, events, or information created after its training date, making it incapable of recommending new or timely content.
- **Hallucination:** LLMs are prone to generating factually incorrect or entirely fabricated information, which can severely erode user trust if the system suggests non-existent items or attributes.
- **Lack of domain awareness:** A general-purpose model has no specific knowledge of a particular service's inventory, such as which products are in stock or their current prices.
- **Difficulty leveraging behavioral data:** The rich collaborative signals from user-item interactions (e.g., ratings, purchases) are not naturally represented in the parametric knowledge of an LLM.

RAG directly addresses these issues by grounding the LLM in an external, authoritative knowledge base. The standard workflow, consists of an offline indexing stage and an online retrieval-generation stage, with the latter depicted in Figure 2.1. During indexing, documents are cleaned, segmented into chunks, converted into vector embeddings, and stored in a vector database. At runtime, a user's query is embedded and used to retrieve the most relevant chunks, which are then passed to the LLM as context to generate a factually grounded response.

The application of RAG is evolving beyond simple fact retrieval into a more sophisticated paradigm, with advanced techniques to further address the limitations of traditional LLMs [16]. This has led to the development of **Graph-RAG** [17], where the retrieval source is a structured Knowledge Graph (KG) that stores relevant information as nodes and relationships. The initial use of RAG was as a "fact-checker" to provide domain awareness and prevent hallucinations. However, state-of-the-art research in RAG applied to CRS now conceptualizes RAG as a *collaborative signal retriever*. The goal is to retrieve not just item descriptions, but the complex, relational data that powers traditional Collaborative Filtering. By applying Graph-RAG, this model allows the system to query for explicit reasoning paths. Frameworks like `G-Refer` [18] and `CRAG` [19] exemplify this approach, using graph traversal to find evidence that is then translated into natural language by the LLM to generate a recommendation and a

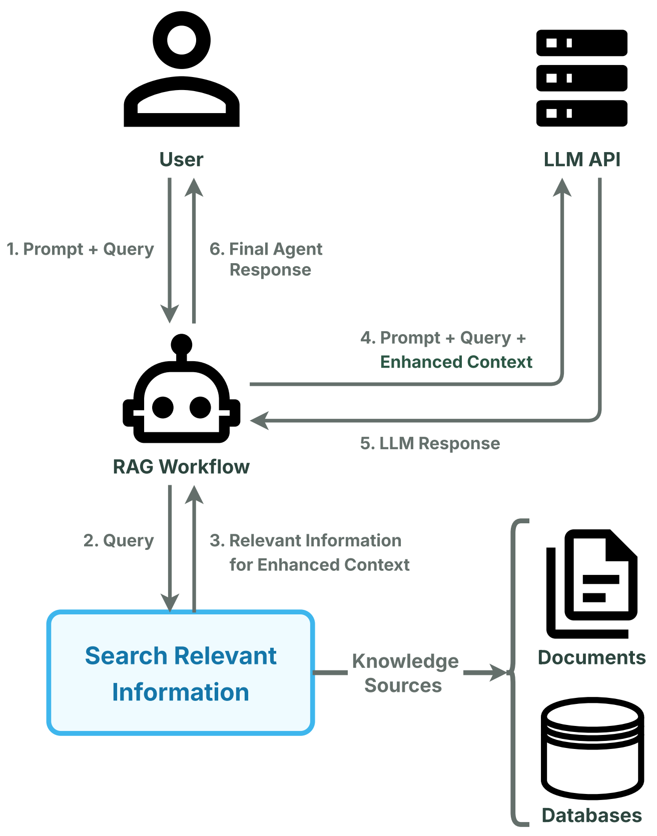


Figure 2.1: The standard Retrieval-Augmented Generation workflow [15].

faithful explanation.

2.1.3. Function Calling

Function Calling, also known as Tool Calling, is a mechanism that enables an LLM to interact with external tools and Application Programming Interfaces (APIs). This capability transforms the LLM from a passive text generator into an active agent capable of performing actions in the digital world. During an inference step, the model can decide that it needs to call an external function to fulfill a user's request. It then generates a structured JSON object containing the name of the function to call and the arguments to pass. The application code executes this function, and the result is fed back to the LLM, which uses it to generate its final response to the user. This technique is fundamental for creating agents that can interact with databases, execute code, or, as in the case of this project, update user preferences and invoke recommender models.

2.1.4. Fine-tuning

Fine-tuning is the process of further training a pre-trained LLM on a smaller, domain-specific dataset. The goal is to adapt the model's behavior, style, or knowledge to a specific task. This stands in contrast to in-context learning, which is facilitated by RAG and does not update the model's weights. While full fine-tuning can be computationally expensive, more efficient methods like Low-Rank Adaptation (LoRA) have been developed to reduce the resource requirements [20]. For this project, a RAG-based approach was chosen over fine-tuning, motivated by the fact that RAG is often more effective and manageable for incorporating new or rapidly changing factual knowledge (such as an item catalog) and avoids the high computational cost and potential for "catastrophic forgetting" that can occur during fine-tuning.

2.1.5. LLM Quantization

Quantization refers to the process of representing a deep learning model's parameters (and sometimes activations) with reduced numerical precision. Instead of storing weights in high-precision floating-point formats such as `FP32` (32-bit), they are mapped to lower-precision formats like `FP16`, `INT8`, or even `INT4`. The main motivation is to reduce the model's size, memory footprint and computational cost, thereby making inference faster and enabling deployment on resource-limited devices. However, because quantization introduces rounding errors, careful methods are needed to mitigate accuracy loss.

There are several well-established approaches to quantization [21]:

- In **Post-Training Quantization (PTQ)**, the model is trained in full precision and converted to a lower-precision format afterwards, making it quick to apply but potentially less accurate for sensitive models.
- **Quantization-Aware Training (QAT)** addresses this loss of accuracy by simulating low-precision effects during training so that the model learns to be robust to them, though this requires additional training time.
- **Dynamic Quantization** applies quantization to weights ahead of inference but computes activation ranges on the fly, offering a flexible solution without calibration data.
- Advanced strategies use *mixed precision*, assigning higher precision to critical layers and lower precision elsewhere, or *group-wise* (block-wise) quantization, where groups of weights share a scaling factor to improve performance at very low bit widths.

Recently, LLM-specific methods have emerged. Algorithms such as *Accurate Post-Training Quantization for Generative Pre-trained Transformers (GPTQ)* use second-order information to minimize quantization error [22], *Activation-aware Weight Quantization for LLM Compression and Acceleration (AWQ)* targets weight distributions to preserve accuracy in low-bit settings [23], and *SmoothQuant* reduces activation outliers before quantization [24]. These methods are designed to balance compression and accuracy for large transformer-based architectures, making it possible to run models with billions of parameters on consumer-grade Graphics Processing Units (GPUs) or even edge devices.

2.1.6. Evaluation

The selection of an appropriate LLM is a paramount decision for useful and effective conversational agents. The chosen model must not only demonstrate strong general language understanding and generation capabilities, but also excel at the specific task of Function Calling, which is central to the proposed architecture. For this objective, LLM evaluation is crucial, as it provides insights into the model's performance across various benchmarks and metrics.

General Evaluation Metrics

The general performance of open-source LLMs is displayed on the Open LLM Leaderboard [25], a platform hosted by HuggingFace that ranks models on a variety of benchmarks (archived since March 13th 2025). The leaderboard employed various common metrics used to evaluate LLMs on standard benchmarks, including:

- **Instruction-Following Evaluation (IFEval):** Evaluates the model's ability to follow explicit formatting instructions, including instruction following, formatting, and generation. Scoring is based on whether the required format was strictly followed (accuracy).
- **Big Bench Hard (BBH):** A suite of difficult tasks across multiple domains such as reasoning and world knowledge. Scoring is based on whether the correct answer was chosen (accuracy).
- **Mathematics Aptitude Test of Heuristics (MATH):** Focuses on advanced high school math problems, including algebra, geometry, and calculus. Scoring is based on exact match to the correct answer.
- **Graduate-Level Google-Proof Q&A (GPQA):** Multiple-choice questions requiring PhD-level knowledge in science (biology, chemistry, physics). Scoring is based on accuracy.
- **Multistep Soft Reasoning (MuSR):** Tests reasoning over long contexts with emphasis on comprehension and logic. Accuracy is used as the scoring metric.
- **Massive Multitask Language Understanding – Professional (MMLU-Pro):** Covers expert-reviewed questions across domains like medicine, law, engineering, and more. Scoring is based on accuracy.
- **Carbon Dioxide Emissions (CO_2 Cost):** Reflects the environmental impact of model inference in kilograms of CO_2, considering data center location and energy mix.

Function Calling Performance Metrics

More critical to this thesis is the model's ability to perform Function Calling. The Berkeley Function Calling Leaderboard (BFCL) [26] is a specialized benchmark designed to evaluate this specific capability in LLMs. To provide a granular understanding of a model's function-calling prowess, the BFCL utilizes several metrics, each targeting a different facet of the task.

- **Overall Accuracy:** This metric represents the unweighted average accuracy across all sub-categories of the BFCL benchmark. It provides a high-level summary of the model's general function-calling ability, balancing its performance across simple, multiple, parallel, and relevance-detection tasks.
- **Abstract Syntax Tree (AST) Evaluation:** This method focuses on the structural and syntactical correctness of the generated function call. The process involves parsing the model's output into an AST, which is a tree representation of the source code's abstract syntactic structure. The evaluation then meticulously checks for several criteria:
    - **Function Name Matching:** Verifies that the name of the function called by the model matches the expected function name in the ground truth answer.
    - **Parameter Matching:** Ensures that all required parameters are present in the model's output and that no extraneous, or "hallucinated", parameters are included.
    - **Type & Value Adherence:** Strictly checks that the data types and values of the provided arguments match the function's definition. For instance, while Python might auto-convert an integer to a float, this is not permissible for other languages like Java or JavaScript in the evaluation unless explicitly allowed. For lists and tuples, the order of elements must match exactly.
- **Single-Turn AST Accuracy (Live and Non-live):** This measures the model's ability to correctly generate a function call in a single interaction (one user query, one model response).
    - **Non-live Accuracy** refers to the static AST evaluation where the syntactic correctness of the function name, parameters, and types is checked against a predefined set of answers without executing the code.
    - **Live Accuracy** goes a step further by executing the generated function call. This is used for tests involving real-world APIs (e.g., REST APIs) to verify that the call is not only syntactically correct but also functional and produces the expected outcome.
- **Multi-turn Accuracy:** Introduced in BFCL V3, this metric evaluates a model's performance in more complex, conversational scenarios that require multiple back-and-forth exchanges to complete a task. Instead of just checking the final function call, this evaluation uses a state-based approach, verifying the actual state of the system (e.g., a file system or booking system) after the model executes its functions. This assesses a model's ability to handle complex workflows, manage context over several turns, and reason about sequential actions. The evaluation includes challenging sub-tasks such as identifying missing parameters and asking clarifying questions, or recognizing when a necessary tool is not available.
- **Relevance and Irrelevance (Hallucination Measurement):** A crucial aspect of a reliable agent is knowing when *not* to call a function. These metrics evaluate a model's ability to avoid hallucination, which in this context means generating a function call when none of the provided tools are appropriate for the user's query.
    - **Relevance Detection:** This measures the model's ability to correctly identify and invoke a relevant

function when one is provided among the available tools.

- **Irrelevance Detection:** This specifically tests the model's resilience to hallucination. In these scenarios, none of the provided functions are suitable for the user's request. A correct response is to invoke no function call. This metric is useful for determining if a model will incorrectly generate a function call despite lacking the proper tools or information.

Qwen2.5 3B Evaluation

For this project, the Qwen2.5 model of 3 billion parameters, quantized during post-training, was utilized [27]. A key factor in its selection was its 32K token context window, which is the maximum number of combined input and output tokens the LLM can process. While output tokens often have a lower limit that varies by model, Qwen2.5's window provides ample capacity for the conversational context and retrieves enough information for the scope of this project, a conclusion reached after preliminary testing. This model size also proved to be optimal for the available 4GB GPU, allowing it to run entirely on VRAM.

In terms of general performance, the largest version, Qwen2.5 with 72 billion parameters, ranks impressively at 6th out of 4576 models. Impressively, the 3 billion parameter version used in this project also holds a respectable position at approximately 1300, placing it in the top 28 %. This strong showing, especially considering its smaller size, indicates a high level of general capability. General performance metrics for Qwen2.5 3B are summarized in Table 2.1.

Metric	**Score**
Average	27.16 %
IFEval	64.75 %
BBH	25.80 %
MATH	36.78 %
GPQA	3.02 %
MuSR	7.57 %
MMLU-Pro	25.05 %
CO_2 Cost	2.78 kg

Table 2.1: General evaluation metrics for Qwen2.5 3B.

In terms of Function Calling performance, on the BFCL as of June 18th, 2025 [1], the 72B Qwen2.5 model was ranked in the top 20 out of 121 models, with most of the higher-ranking models being closed-source. The 3B variant of Qwen2.5, the model used in this research, was ranked 81 out of 121, making it one of the top performers among similarly-sized models. The performance of Qwen2.5 3B on the BFCL Function Calling metrics is detailed in Table 2.2.

[1] *BFCL V3.* `https://web.archive.org/web/20250618114222/https://gorilla.cs.berkeley.edu/leaderboard.html`

Metric	Score
Overall Accuracy	50.37
Mean Latency	4.3s
Non-live (AST)	78.83
Live (AST)	69.39
Multi-turn Accuracy	6
Relevance	88.89
Irrelevance	64.26

Table 2.2: Function Calling evaluation metrics for Qwen2.5 3B.

Other models of a similar size were evaluated, but they exhibited inferior performance in Function Calling accuracy. Consequently, Qwen2.5 3B was selected as the most suitable model for this project, balancing strong Function Calling capabilities with manageable computational requirements.

2.2. Recommender Systems

Recommender Systems represent a mature and fundamental field of research within information filtering. The primary goal of an RS is to mitigate information overload by predicting a user's preference for an item and proactively suggesting relevant items from a large catalog [3]. While conventional RSs have typically relied on a static, single-interaction paradigm, they provide the foundational predictive models upon which modern CRSs are built. This section provides an overview of the primary classifications of recommendation models and the standard methodologies used for their evaluation.

2.2.1. Model Classification

Recommendation algorithms can be broadly categorized into several families, each with distinct approaches to modeling user preferences.

- **Collaborative Filtering (CF):** This is the most prevalent paradigm in RS. It operates on the principle of homophily, or "wisdom of the crowd", by identifying users with similar tastes or items with similar interaction patterns. CF methods are typically divided into two sub-categories: *memory-based* approaches, such as the widely-used Item-based k-Nearest Neighbors (`ItemKNN`) algorithm [28], and *model-based* approaches, which learn latent factor representations of users and items. Latent factor models, such as those based on Matrix Factorization or Factorization Machines (FM), are powerful but can suffer from the cold-start problem when new users or items are introduced. This has been addressed by applying an incremental algorithm to avoid having to compute the entire matrix from scratch [29]. Recent approaches such as A-LLMRec demonstrate how LLMs can be coupled with collaborative filtering knowledge to achieve strong performance across both cold and warm scenarios [30]

- **Content-Based Filtering:** In contrast to CF, content-based methods recommend items based on their intrinsic properties (e.g., genre, brand, textual description). The system learns a profile of the user's interests based on the features of items they have previously liked and recommends new items with similar features.
- **Hybrid Models:** Most modern, production-grade systems are hybrid, combining multiple recommendation strategies to leverage their respective strengths and mitigate their weaknesses. A common hybrid approach is to combine collaborative and content-based signals to improve recommendation quality, particularly for cold-start scenarios.
- **Deep Learning-Based Models:** More recently, deep learning has had a significant impact on the field. Graph Neural Networks (GNNs), for example, have become a powerful tool for learning complex, high-order relationships directly from the user-item interaction graph. Another impactful class of models is based on autoencoders, with models like `EASER` demonstrating strong performance on sparse data by learning a dense representation of the item space [31].

2.2.2. Evaluation

Evaluating the performance of a recommendation model is a paramount step in the research and development process. The methodology is typically divided into two main paradigms: offline evaluation, which uses historical data to measure predictive accuracy, and online evaluation, which assesses the model's impact on user behavior in a live system [32].

Offline Evaluation

Offline evaluation is the most common approach in academic research due to its low cost and reproducibility. The protocol involves splitting a historical dataset of user-item interactions into training, validation, and test sets. The model is trained on the former, assessed on the second, and its final performance is measured by its ability to predict the held-out interactions in the test set. The metrics used depend on the specific recommendation task. For rating prediction tasks, accuracy is measured using metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

For the more common top-N ranking task, the goal is to measure the accuracy of the recommended list. Common metrics include Precision [2.1a], Recall [2.1b] and F1-Score [2.1c]:

$$P@k = \frac{|Relevant \cap Returned_k|}{k} \tag{2.1a}$$

$$R@k = \frac{|Relevant \cap Returned_k|}{|Relevant|} \tag{2.1b}$$

$$F_1@k = \frac{2 \cdot P@k \cdot R@k}{P@k + R@k} \tag{2.1c}$$

While Precision and Recall measure the accuracy of the recommendation set, they do not account for the ranking of the items within that set. Rank-aware metrics, such as Normalized Discounted Cumulative Gain (NDCG) [2.2c], address this by assigning higher importance to relevant items that appear at the top of the list. It considers both the relevance and the rank of recommended items, and stems from the normalization of the Discounted Cumulative Gain (DCG) [2.2a], with the ideally ordered DCG or IDCG [2.2b]. Higher scores are assigned to relevant items that are ranked higher in the recommendation list, applying a discount based on the position in the list.

$$DCG = \sum_{k=1}^{|Rel|} \frac{Relevance(d_k)}{\log_2(k+1)} \tag{2.2a}$$

$$IDCG = DCG_{Ideal\ order} \tag{2.2b}$$

$$NDCG = \frac{DCG}{IDCG} \in [0,1] \tag{2.2c}$$

Here, $Relevance(d_k)$ is the relevance of the item at position k, and IDCG is the ideal DCG score for a perfect ranking.

Beyond accuracy, other important offline metrics include **coverage** (the proportion of items the system can recommend), **diversity** (how different the recommended items are from each other), and **novelty** (the ability to recommend new or unexpected items). Although these offline metrics are the standard for quantitative model comparison, this thesis focuses on a qualitative, user-centric evaluation through usability testing, as the primary research questions are related to the interactive experience and user perception.

Online Evaluation

Online evaluation, typically conducted through A/B testing or user studies, is frequently used for measuring the real-world impact of an RS. In this method, one or more user groups are exposed to a system, and their behavior is monitored and analyzed. This approach allows for the measurement of important business metrics and user experience indicators, such as:

- **Click-Through Rate (CTR):** The proportion of recommended items that users click on.
- **Conversion Rate:** The proportion of recommendations that lead to a desired action (e.g., a purchase or a subscription).
- **User Engagement and Satisfaction:** Measured through metrics like session duration, interaction frequency, or direct user feedback and surveys.

While online evaluation provides the most practically valid results, its high cost, complexity, and slow iteration time mean that it is typically performed after a model has already been validated through offline experiments. This thesis focuses on a user-centric study that aligns with the principles of online evaluation by directly measuring user satisfaction and task success.

2.2.3. The Cold-Start Problem

A fundamental and persistent challenge for Recommender Systems, particularly those based on Collaborative Filtering, is the **cold-start problem** [33]. This issue arises when the system lacks sufficient historical interaction data to make reliable inferences. It manifests in two primary forms:

- **User Cold-Start:** This occurs when a new user joins the system. Without any past ratings or interactions, the CF model has no data upon which to base a personalized recommendation, rendering it effectively blind to the user's preferences.
- **Item Cold-Start:** This occurs when a new item is added to the catalog. Until the item has been rated by a sufficient number of users, the system cannot recommend it to others based on collaborative patterns, hindering the discovery of new content.

Various strategies have been developed to mitigate the cold-start problem in traditional RSs. A common approach is to employ a **hybrid model** that falls back on content-based features. For a new item, its metadata (e.g., genre, author) can be used to recommend it to users who have liked similar items. For a new user, demographic information or a simple onboarding process that asks them to select a few interests can be used to build an initial profile. More advanced techniques include incremental learning models like incremental FMs [29] or Dynamic EASER [34], which can adapt to new data without full retraining, and graph-based reasoning to infer preferences for cold-start users [35].

While these methods provide partial solutions, the conversational paradigm offers a more direct and natural resolution to the user cold-start problem. Instead of relying on proxy information, a CRS can simply ask a new user about their preferences, mirroring how a human expert would begin a recommendation dialogue. This ability to actively elicit preferences makes conversational systems well-suited to overcoming one of the longest-standing challenges in the field of Recommender Systems.

2.3. Conversational Recommender Systems

Conversational Recommender Systems represent a paradigm shift in the field of recommendation, moving from the static, one-shot presentation of item lists to a dynamic, multi-turn dialogue between the user and the system. This evolution is motivated by the inherent shortcomings of traditional RSs, which struggle with unreliable preference estimation from sparse historical data, an inability to adapt to the user's immediate context, and the flawed assumption that users always have well-defined goals [36]. A CRS addresses these issues by engaging the user in a conversation to actively elicit their current needs and preferences, thereby providing a natural solution to the cold-start problem and enabling more accurate, context-aware recommendations [37].

To provide a structured understanding of this domain, this section first deconstructs a typical CRS into its core architectural components, and it then explores the latest methodological developments and research trends in the field.

2.3.1. Architectural Components of a CRS

A modern CRS can be deconstructed into several core architectural components, each responsible for a specific aspect of the conversational recommendation process. A conceptual view of how these components interact is shown in Figure 2.2. Common building blocks referred to in research are: Interaction Modalities, Underlying Knowledge, and Computational Tasks [36].

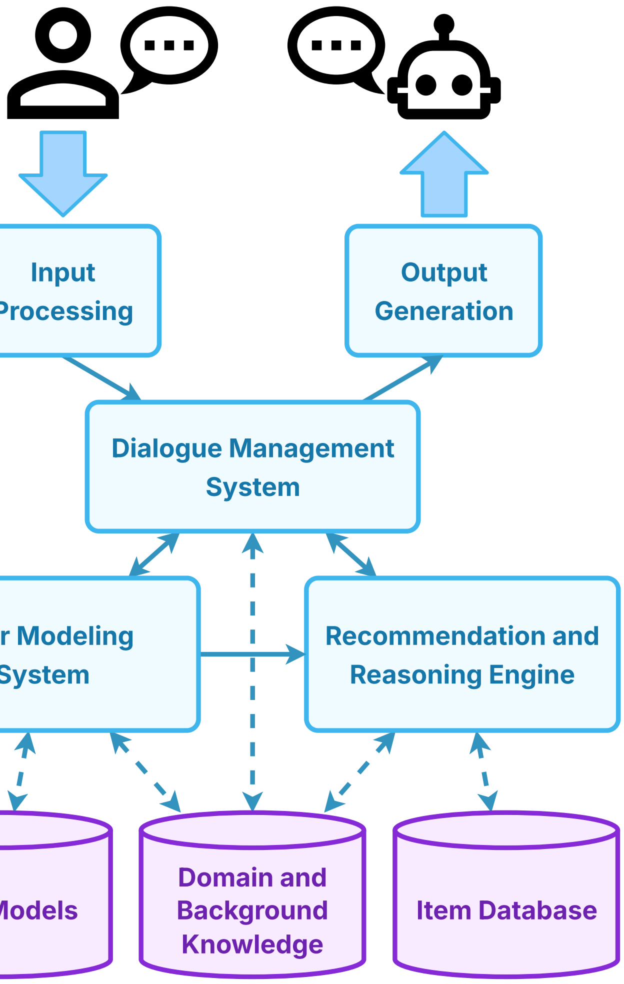

Input Processing
Output Generation
Dialogue Management System
User Modeling System
Recommendation and Reasoning Engine
User Models
Domain and Background Knowledge
Item Database

Figure 2.2: A conceptual diagram of the core architectural components of a Conversational Recommender System [38].

Interaction Modalities

This dimension defines *how* the user and the system interact. It encompasses the **input/output methods**, which can range from structured inputs like buttons and forms to the more flexible but complex paradigm of natural language text or speech. Many contemporary systems employ a hybrid approach, combining natural language input with visual outputs like interactive item cards [39]. It also includes the **interaction initiative**, which determines who leads the dialogue. This can be *system-driven*, where the system asks a series of questions (a model often called “System Ask, User Respond” or SAUR); *user-driven*, where the user directs the flow; or *mixed-initiative*, which allows for a more natural back-and-forth and is the most common approach in modern systems [36, Section 3].

Underlying Knowledge and Data

This category describes the information sources that the CRS relies upon to function effectively [36, Section 4].

- **User Modeling:** This is the process of acquiring and representing user preferences. The model can be built from explicit item ratings, preferences for specific item *facets* (e.g., genre, brand), or unstructured features. This information can be stored ephemerally for a single session or as part of a persistent, long-term user profile.
- **Dialogue State Tracking:** The system must maintain a representation of the conversation’s current state to inform its next action. This can be managed by an explicit state machine or learned implicitly by a neural model.
- **Background Knowledge:** This is a paramount component for providing context. It often takes the form of a structured Knowledge Graph containing item metadata and the relationships between them, which is invaluable for reasoning and generating explanations.

Computational Tasks

This dimension outlines the core functions that the CRS must execute. The main tasks include deciding which question to **Request** next, generating item suggestions (**Recommend**), providing justifications for those suggestions (**Explain**), and handling non-recommendation-related dialogue (**Respond**). To support these, a CRS relies on several underlying processes, most notably Natural Language Understanding (NLU) to detect user intents and Sentiment Analysis to gauge user feedback from their responses [36, Section 5].

2.3.2. Latest Developments

The methodologies for building CRSs have evolved significantly, moving from modular, pipeline-based systems to more integrated, end-to-end frameworks powered by modern LLMs.

Preference Elicitation Strategies

The core task of any CRS is to understand user preferences through conversation. Early research focused on several notable strategies for this **preference elicitation** process [4, Section 3]:

- **Critiquing:** In this paradigm, the system presents a set of recommendations, and the user refines them by providing natural language critiques or constraints (e.g., "show me something cheaper"). This allows for an iterative narrowing of the search space based on direct feedback [4, Section 3.2].
- **Facets-Based Elicitation:** Here, the system identifies primary item attributes, or *facets* (e.g., genre, brand, color), and asks targeted questions about them to build a structured representation of the user's needs [4, Section 3.3].
- **Question-Driven Approaches:** More advanced systems strategically decide which questions to ask to maximize information gain. The **System Ask, User Respond (SAUR)** model is a well-known formalization of this, where the system learns a policy for asking clarification questions to efficiently guide the conversation toward a successful recommendation [4, Section 3.4.1].

Towards LLM-Native Architectures

A significant recent trend is the move away from complex, multi-component architectures towards unified frameworks that formulate the entire CRS task as a language problem. While older toolkits like `CRSLab` [40] treated recommendation, dialogue, and policy as separate, interconnected modules, modern approaches leverage a single, powerful LLM to handle multiple functions simultaneously [41, Section 2.2]. Frameworks like `RecInDial` [42] demonstrate this by using a pre-trained language model to unify the understanding of conversational context and the generation of both dialogue responses and item recommendations, removing the semantic gap that often exists between separate modules.

Reinforcement Learning for Dialogue Policy Optimization

To optimize the multi-turn conversational strategy, Reinforcement Learning (RL) has emerged as a state-of-the-art approach [43]. In this paradigm, the CRS is framed as an agent that learns an optimal *dialogue policy* through trial and error. At each turn of the conversation, the agent must decide on an action (e.g., ask about an attribute, recommend an item) to maximize a cumulative long-term reward, which is typically a function of user satisfaction and conversation efficiency. This allows the system to learn sophisticated strategies that balance exploration (gathering more information about user preferences) and exploitation (making recommendations based on current knowledge) [44].

Holistic vs. Simulated Systems

A critical distinction in modern CRS research is the nature of the data used for training and evaluation. Early work relied on *simulated* user interactions, which, while useful for optimizing algorithms, fails to capture the complexity and unpredictability of real human conversation. In response, the field is increasingly moving towards building and evaluating **holistic** CRSs [41]. A holistic system is one that is trained and evaluated end-to-end on datasets of real human-to-system conversations. This shift requires more comprehensive evaluation, assessing not only recommendation accuracy but also the linguistic quality and coherence of the dialogue itself.

2.4. Explainability in Recommendation

As Recommender Systems become more powerful and integrated into daily decision-making, the need for transparency and accountability has grown from a desirable feature to an important requirement. Explainable Artificial Intelligence is the field dedicated to addressing this need, aiming to make the decisions of AI systems understandable to humans. In the context of RSs, explanations can significantly improve user satisfaction, enhance trust, increase the persuasiveness of recommendations, and provide a level of transparency that helps users make more effective decisions [5].

This need is amplified in Conversational Recommender Systems. An explanation is not merely a static justification appended to a recommendation; it is an active component of the dialogue, acting as a way of gathering more accurate user feedback. By presenting the reasons for a recommendation (e.g., “I'm suggesting this movie because it shares an actor you like”), the system invites the user to agree or disagree with those specific reasons, enabling a collaborative refinement of both the recommendations and the system's understanding of the user's preferences. This section establishes a formal taxonomy for classifying XAI methods, surveys the most popular explanation generation techniques, and concludes by discussing the complex challenge of evaluating explanation quality.

2.4.1. A Taxonomy of XAI Methods

The field of XAI encompasses a wide variety of techniques. To better understand their applicability and limitations, a formal taxonomy is often used to classify them along several dimensions. A widely accepted classification is based on the **scope** of the explanation and the **timing** at which the explanation is generated relative to the model's training [45].

Scope of the Explanation

This dimension describes whether an explanation pertains to a single decision or the model as a whole.

- **Global Explanations:** A global explanation aims to make the entire logic of a model understandable. It describes the model's general behavior across all possible inputs, for example, by identifying the most influential features on average. While useful for model developers and auditors, this scope is often too general to be helpful to an end-user of an RS.
- **Local Explanations:** A local explanation, in contrast, is focused on a single, specific prediction. In the context of RSs, this means explaining why one particular item was recommended to one particular user at a specific point in time. This is the most relevant scope for this thesis, as the goal is to provide

justifications for individual recommendations to build user trust.

Timing of the Explanation

This dimension distinguishes between models that are transparent by design and those that require an external method to be explained after the fact.

- **Ante-hoc (Intrinsic) Explainability:** This refers to the use of models that are considered inherently interpretable or “white-box” by design. Examples include linear regression, simple decision trees, and rule-based systems where the decision-making process is directly inspectable. These models often involve a trade-off, sacrificing some predictive accuracy for a higher degree of transparency [46].
- **Post-hoc Explainability:** This is the most common approach for complex, high-performance models (e.g., deep neural networks) that are considered “black-boxes”. A post-hoc method is a separate technique that is applied after a model has been trained. It analyzes the model’s inputs and outputs to generate an explanation for its behavior without modifying the model itself. Nearly all of the advanced methods discussed in this chapter, such as graph-based reasoning and counterfactual explanations, are post-hoc techniques.

In this thesis, a post-hoc approach is adopted to generate local explanations for individual recommendations. The explanations are derived from a hybrid model that combines graph-based reasoning with LLM inference, allowing for a transparent justification of the system’s recommendations.

2.4.2. Popular Methods for Explaining Recommendations

A diverse array of techniques has been developed for generating explanations in Recommender Systems, each with different underlying principles and goals.

Feature and Review-Based Explanations

One of the most direct methods for explainability involves leveraging the textual data associated with items. **Feature-level explanations** present key item attributes that align with a user’s inferred preferences (e.g., “...because you like the *Action* genre”), and can be obtained through techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) [47, 48]. **Review-level explanations** go a step further by extracting representative segments from user-generated reviews that highlight prominent aspects of an item. Early work in this area used topic modeling to identify latent topics in reviews that could be presented to users. More advanced models like the Neural Attentional Regression model with Review-level Explanations (NARRE) learn the “usefulness” of reviews to select the most informative ones as explanations [4, Section 4.1].

Graph-Based Explanations

With the rise of graph modelling in recommendation, a powerful paradigm has emerged that treats explainability as a graph reasoning problem. The core idea is that a compelling explanation can be represented as a path or subgraph that connects a user to a recommended item through a series of meaningful relationships [49]. For instance, a path like `User-[RATED]->Item1-[HAS_ACTOR]->Actor-[ACTED_IN]->Item2 (Recommended)` provides a transparent and interpretable reasoning trail ("You liked `Item1` which features `Actor`, who also appears in `Item2`"). This approach is highly effective because these paths represent concrete relationships in the data, enabling explainable recommendations while leveraging both user-item interactions and contextual knowledge from a domain-specific knowledge graph [50]. This method is central to this thesis, relying on `FalkorDB`'s Cypher query capabilities to find collaborative and content-based reasoning paths that are then translated into natural language.

Counterfactual Explanations

Inspired by causal inference, counterfactual reasoning explains a decision by identifying the minimal change to an input that would alter the outcome. The explanation answers the question, "Why was item A recommended instead of item B?" by identifying the critical feature that led to the decision [51]. Frameworks like `CECR` (Counterfactual Explainable Conversational Recommender) integrate this technique directly into a CRS, using the generated counterfactuals not only as explanations but also as augmented training data to continuously improve the model's performance [52].

LLM-Native Explanations

The state-of-the-art in explainability for reommender systems spans towards LLM-generated explanations. LLMs can act as surrogate models to interpret and mimic black-box recommenders, allowing for faithful natural language explanations [53], and incremental conversational frameworks integrate recommendation, explanation generation, and user feedback to enhance both accuracy and user-aligned explainability [54].

This approach is often combined with other methods to ensure the explanations are grounded in factual evidence. The **Graph-RAG** paradigm is particularly relevant here; frameworks like `G-Refer` first retrieve explicit Collaborative Filtering paths from a graph and then use an LLM to translate this structured evidence into a coherent, natural language explanation [18]. This ensures the explanation is not a post-hoc rationalization but is causally faithful to the underlying recommendation logic. This synergy between graph reasoning and LLM generation represents a noteworthy trend towards more trustworthy and effective explainable systems [55, 56].

2.4.3. Evaluating Explanation Quality

Evaluating the quality of an explanation is a multi-faceted and complex challenge. An explanation must be not only factually correct but also understandable, persuasive, and useful to the end-user. The evaluation of XAI methods can be broadly categorized into offline (or proxy) evaluation and online (or human-grounded) evaluation [46].

Offline (Proxy) Evaluation

Offline evaluation uses algorithmic metrics to assess the properties of an explanation without direct human involvement, serving as proxies for explanation quality. Prominent offline metrics include:

- **Fidelity:** This measures how accurately an explanation reflects the underlying model's reasoning process. For a post-hoc explainer, fidelity assesses how well it mimics the behavior of the original black-box model.
- **Consistency / Stability:** This evaluates whether the explainer produces similar explanations for similar models or inputs. A stable explainer should not generate wildly different explanations for minor changes in the input data.
- **Faithfulness:** A highly sought-after property, faithfulness measures the causal link between the explanation and the model's prediction. For counterfactual explanations, for example, metrics like **Probability of Necessity** ("is the feature necessary for the outcome?") and **Probability of Sufficiency** ("is the feature sufficient to cause the outcome?") can quantitatively measure the causal strength of an explanation.

The development of specialized datasets, such as `E-ReDial` [57], which contains thousands of high-quality, human-annotated explanations, is also a crucial step towards creating standardized benchmarks for training and evaluating explainable systems offline.

Online (Human-Grounded) Evaluation

While offline metrics are useful for technical validation, the ultimate goal of an explanation is to be useful to a human. Human-grounded evaluation, conducted through user studies, is therefore considered the gold standard for assessing explanation quality. These studies measure a range of user-centric dimensions, including:

- **Transparency:** How clearly the user understood *why* the system provided a specific output.
- **Satisfaction:** The user's overall satisfaction with the combination of the system's output and the explanations received.
- **Trust & Persuasiveness:** Whether the explanation increased the user's trust and confidence in the system's suggestions.

- **Cognitive Load:** How mentally demanding the explanations were to read and understand.

The experimental methodology of this thesis is firmly rooted in this human-grounded paradigm, using qualitative ratings and the standardized SUS questionnaire to directly measure these user-centric aspects of the generated explanations.

3

Methodology

This chapter details the methodological framework designed and implemented to address the research questions outlined in the introduction. It provides a technical blueprint of the system, describing the overall architecture and the specific methods employed for conversational workflow management, user preference elicitation, hybrid recommendation, and natural language explanation generation. The methodology is presented in two main parts: first, an overview of the high-level framework and its primary components, and second, a deeper dive into the implementation of the Hybrid Conversational Recommender System itself.

3.1. Methodological Framework

The research is centered on a hybrid, modular framework that integrates a conversational layer, managed by an LLM, with specialized backend modules for user profiling, recommendation, and explanation. The system is designed as an event-driven workflow orchestrated by the LlamaIndex framework. This architecture allows the LLM to act as an intelligent agent that can reason about the dialogue context and employ external, specialized tools to fulfill user requests. The following subsections detail the core components of this framework, each designed to address a specific research question.

3.1.1. Conversation Workflow and Preference Elicitation

To address **RQ1** regarding dynamic preference elicitation, the system is built around an event-driven conversational workflow. This workflow manages the multi-turn, mixed-initiative dialogue between the user and the agent. The entire conversation is modeled as a graph of events, where nodes represent different states and steps in the process, as illustrated in Figure 3.1.

At each turn, the LLM processes the user's natural language input within the context of the conversation history. Based on this context, the workflow's underlying logic determines the next action. This could be to ask a clarifying question to elicit more specific preferences (e.g., asking about a genre if the user's intent is vague), or to trigger a recommendation. This decision is not pre-scripted but is gui-

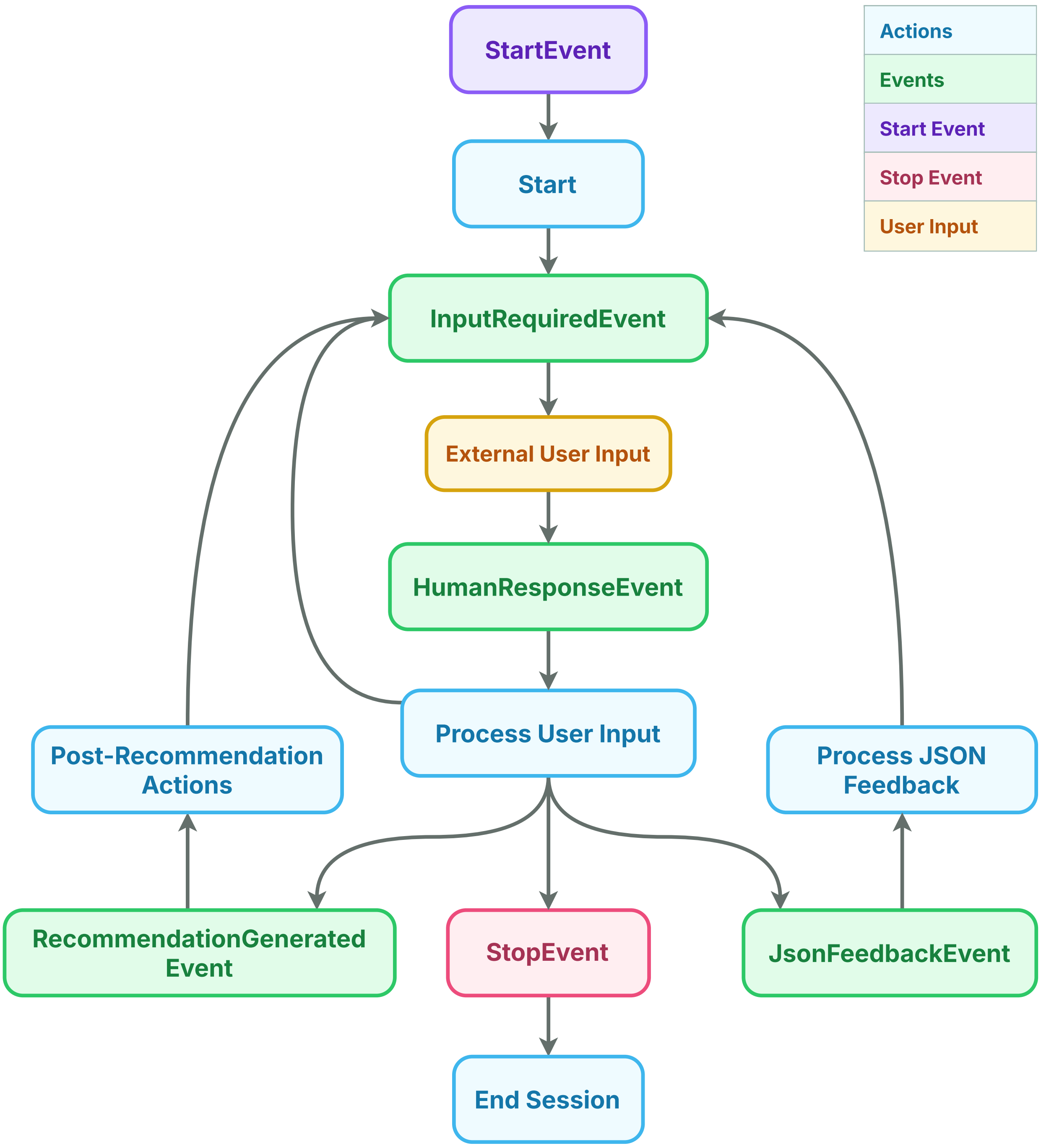


Figure 3.1: An event graph diagram representing the conversational workflow.

ded by the LLM's reasoning capabilities. To interact with the system's other components, the workflow leverages Function Calling. This allows the LLM to autonomously decide when to call the specialized recommendation or explanation modules, passing the relevant information from the conversation as arguments.

3.1.2. Retrieval-Augmented Generation for Contextual Recommendations

To address the first part of **RQ2** concerning contextual recommendations, particularly in cold-start scenarios, the framework employs a strategy analogous to Graph-RAG. The system's KG, built and managed by `FalkorDB`, serves as a rich, structured source for real-time retrieval.

When the conversational agent determines that a contextual recommendation is needed, it executes Cypher queries against the dataset's graph to find relevant items based on the immediate conversational context (e.g., item attributes mentioned by the user). It can perform content-based filtering by finding items with matching properties or Collaborative Filtering by identifying items liked by similar users, even with minimal interaction data.

3.1.3. Expert Recommender Model for Personalized Recommendations

To address the second part of **RQ2** regarding deeply personalized recommendations for established users, the framework integrates a pre-trained expert recommender model. This component is responsible for capturing the complex, latent patterns within the user-item interaction data.

The expert model is invoked by the conversational agent via a function call when the dialogue suggests that a more personalized set of recommendations is required. The model used is `EASER` (Embarrassingly Shallow Autoencoders for Sparse Data) [31], a powerful autoencoder-based algorithm known for its strong performance on sparse data. This model is trained offline on the entire interaction history for a given agent's dataset, allowing it to provide high-accuracy recommendations that complement the real-time, contextual (content-based) and Collaborative Filtering suggestions from the graph-based engine. The reasoning behind selecting `EASER` is explained in Subsection 3.2.4

3.1.4. Explainability using Natural Language Explanations

The methodology for generating explanations, addressing **RQ3**, forms a central component of the framework. It is designed as a neuro-symbolic, two-stage approach that integrates the structured reasoning capabilities of a KG with the generative fluency of an LLM, ensuring that explanations are both causally faithful and accessible to end users.

Stage 1: Symbolic Evidence Retrieval

In the first stage, the system queries the KG to identify human-interpretable reasoning paths linking a user to a recommended item. Evidence is retrieved hierarchically according to the following principles:

- **Content-based:** Identifies shared attributes between the recommended item and items previously preferred by the user. This supports explanations such as: "...because you enjoyed other items in the same category".
- **Collaborative:** Detects `User-Item-User-Item` paths to leverage preferences of similar users, enabling explanations like: "...because other users with similar tastes also liked this".
- **Popularity:** As a fallback, global metrics (e.g., average rating, number of interactions) are retrieved to provide general evidence: "...because it is generally well-received".

Stage 2: Natural Language Synthesis

The retrieved symbolic evidence is compiled into a structured prompt for the LLM, which synthesizes it into a coherent, fluent explanation. This combination ensures that the resulting explanations are grounded in actual evidence while remaining persuasive and natural.

By design, this method functions as a post-hoc, model-agnostic local *explanation engine*, producing plausible, data-grounded justifications even for black-box recommenders [49, 55].

3.2. Hybrid Conversational Recommender System

This section provides a detailed examination of the implemented Hybrid Conversational Recommender System as an LLM-based agent. It builds upon the high-level methodological framework by describing the concrete methods used to realize the system's core functionalities. The overall architecture is designed as a hybrid workflow that combines a real-time graph-based engine with a pre-trained expert model, coordinated by an LLM.

3.2.1. Workflow Implementation

The conversational workflow is instantiated through a dynamic, state-driven interaction model that coordinates between user inputs and recommendation logic, all defined in a `HybridCRSWorkflow` module. It maintains context, tracks preferences, and integrates multi-source recommendations. This interaction happens across discrete steps, each encapsulated as an event in the graph-based conversational controller, as described previously in Figure 3.1. The implementation of the conversational process can be broken down in several steps, as depicted in the high-level workflow diagram in Figure 3.2.

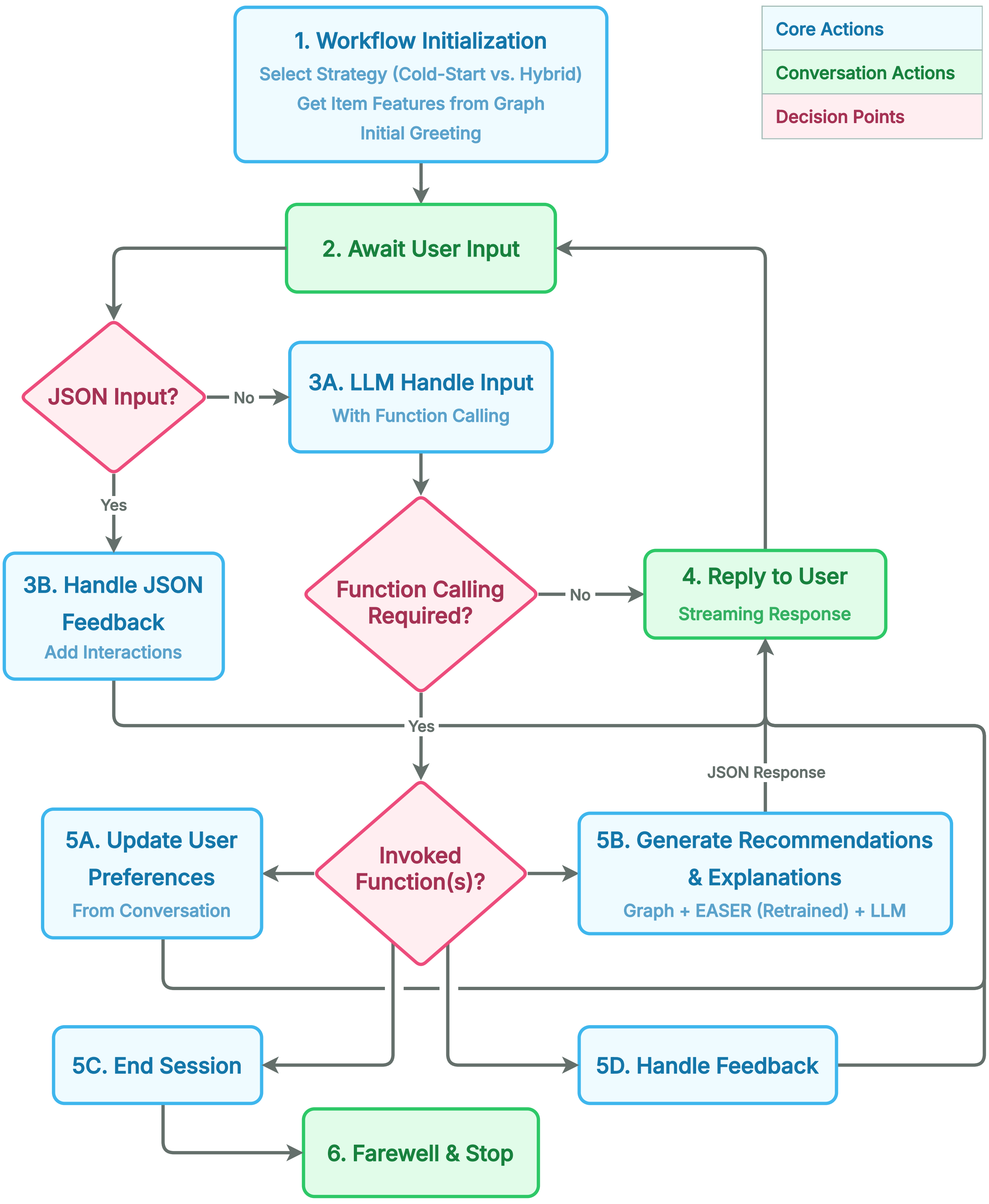


Figure 3.2: A high-level diagram of the Hybrid Conversational Recommendation Workflow.

1. **Session Initialization**:
   - The agent begins by extracting the feature schema from a structured item graph (e.g., *name*, *category*, *rating*).
   - The system dynamically determines user status (`cold-start` vs. `expert-enabled`) from interaction count.
   - A memory buffer is initialized with an introductory prompt, which includes the dataset description.
   - The agent introduces itself and initiates the first user interaction.
2. **User Turn and Input Processing**:
   - Upon receiving a user utterance, the memory buffer logs the interaction history.
   - The agent evaluates whether the input contains structured feedback (e.g., explicit ratings) or free-text natural language.
   - Based on this classification, different response paths are followed—either to register feedback or to query further preferences.
3. **Tool Invocation and Autonomous Action**:
   - The agent determines—without hardcoded rules—whether a tool should be invoked.
   - Tool options include preference updates, recommendation requests, feedback processing, or terminating the session.
   - Each tool is dynamically chosen based on context and invoked with parameters derived from the ongoing dialogue.
4. **Preference Elicitation and Correction**:
   - When a preference is expressed, it is validated against the known schema of item features.
   - If a mismatch is detected (e.g., *"comedyish"* instead of *"comedy"*), the agent attempts to infer the correct canonical value.
   - Preferences are stored incrementally in a structured user profile that supports categorical, numerical, and token-based types.
5. **Recommendation Generation**:
   - When sufficient information has been gathered, the agent requests personalized recommendations.
   - For new users, contextual reasoning is applied using item-attribute filtering from the graph-based backend.
   - For existing users, a hybrid strategy is adopted—merging graph-based reasoning (contextual recommendations & Collaborative Filtering) and the expert `EASER` model.
   - The top results are merged and presented with natural language explanations.

6. **Explanation Generation**:
   - Structured explanation facts from the graph-based model (e.g., shared categories, popularity) are aggregated.
   - These are then paraphrased into a coherent, user-facing justification by the agent.
   - The final output is delivered as a fluent paragraph that contextualizes each recommended item.
7. **Feedback Collection and Adaptation**:
   - Users may respond to recommendations with explicit (e.g., JSON ratings) or implicit feedback (e.g., *“I liked the first one”*).
   - The agent interprets this feedback, updates the user profile, inserts user-item interactions into the graph, and into the interaction dataset in disk.
8. **Session Termination**:
   - If the user signals completion (e.g., *“that’s all for now”*), the agent invokes a termination tool.
   - A closing message is generated and the session is finalized.

The agent’s operation within this workflow is entirely asynchronous and stateful. Each event transition is informed by the user’s evolving preferences and previous interactions, enabling a fluid, mixed-initiative dialogue that adapts to diverse user needs. As the session progresses, the agent alternates seamlessly between eliciting preferences, recommending items, and incorporating feedback—all while preserving a coherent conversation thread.

3.2.2. User Profiling

To address **RQ1**, a dynamic user profile is constructed in memory for each conversational session. This profile serves as a short-term model of the user’s current interests and constraints, and it is built and refined turn-by-turn throughout the dialogue.

The profile is composed of two primary components:

- **Contextual Preferences:** This is a structured dictionary that stores explicit constraints mentioned by the user. When a user expresses a preference related to an item’s attributes (e.g., “I’m looking for a comedy movie”), the LLM extracts these entities. The system then updates the profile with this information, such as by setting the desired value for a “category” feature.
- **Item Preferences:** This component stores explicit feedback on items that have been presented to the user. When a user provides feedback (e.g., “I liked the first one”), the LLM interprets this input and infers a numerical rating (e.g., 5.0 for positive feedback), logging this rating together with the specific item identifier in the profile.

This dynamic profile is the primary input for the recommendation-generation tools, ensuring that the suggestions are always aligned with the user's most recently expressed needs.

3.2.3. Integration of Retrieval-Augmented Generation

The system employs a hybrid recommendation strategy that adapts to the user's interaction history, integrating graph-based methods (parallel to Graph-RAG) with an expert model to provide relevant and diverse suggestions. This methodology addresses the complex nature of **RQ2** by combining contextual, collaborative, and deep learning-based signals. The overall procedure, illustrated in Figure 3.3, begins by assessing the user's interaction history with the `FalkorDBRecommender` module, which is responsible for graph-based recommendations and explanations.

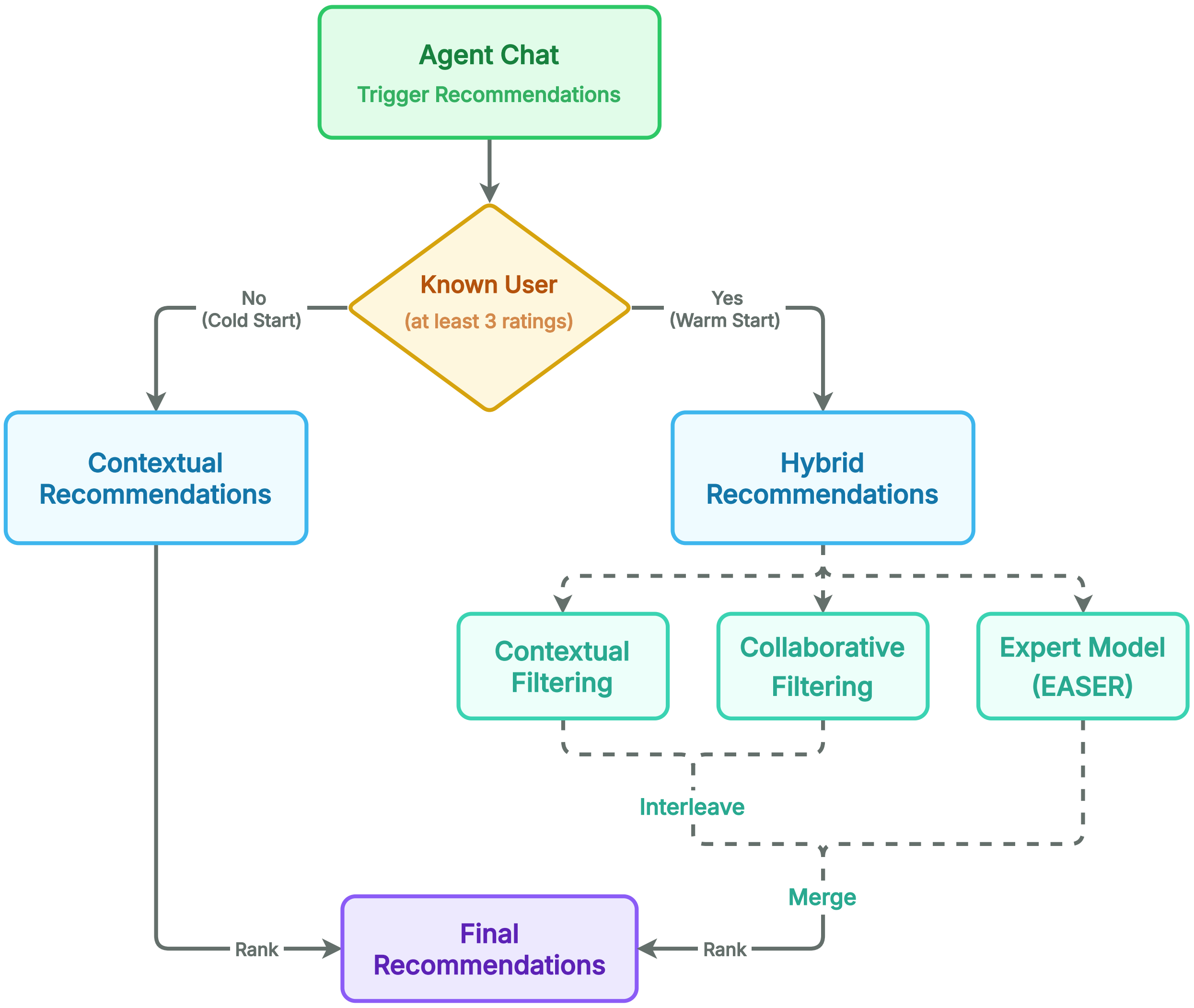


Figure 3.3: A diagram of the generation of hybrid recommendations.

A key decision point in the workflow is whether the user has rated at least three items. If not, the system operates in a "cold-start" mode, relying exclusively on a contextual recommendation method.

Contextual Recommendations for Cold-Start Scenarios

In cold-start situations or when a user's request is highly specific, the system utilizes a graph-based contextual recommender. This component provides the system with its contextual reasoning capabilities, effectively implementing the Graph-RAG paradigm. The process for generating these recommendations is depicted in Figure 3.4.

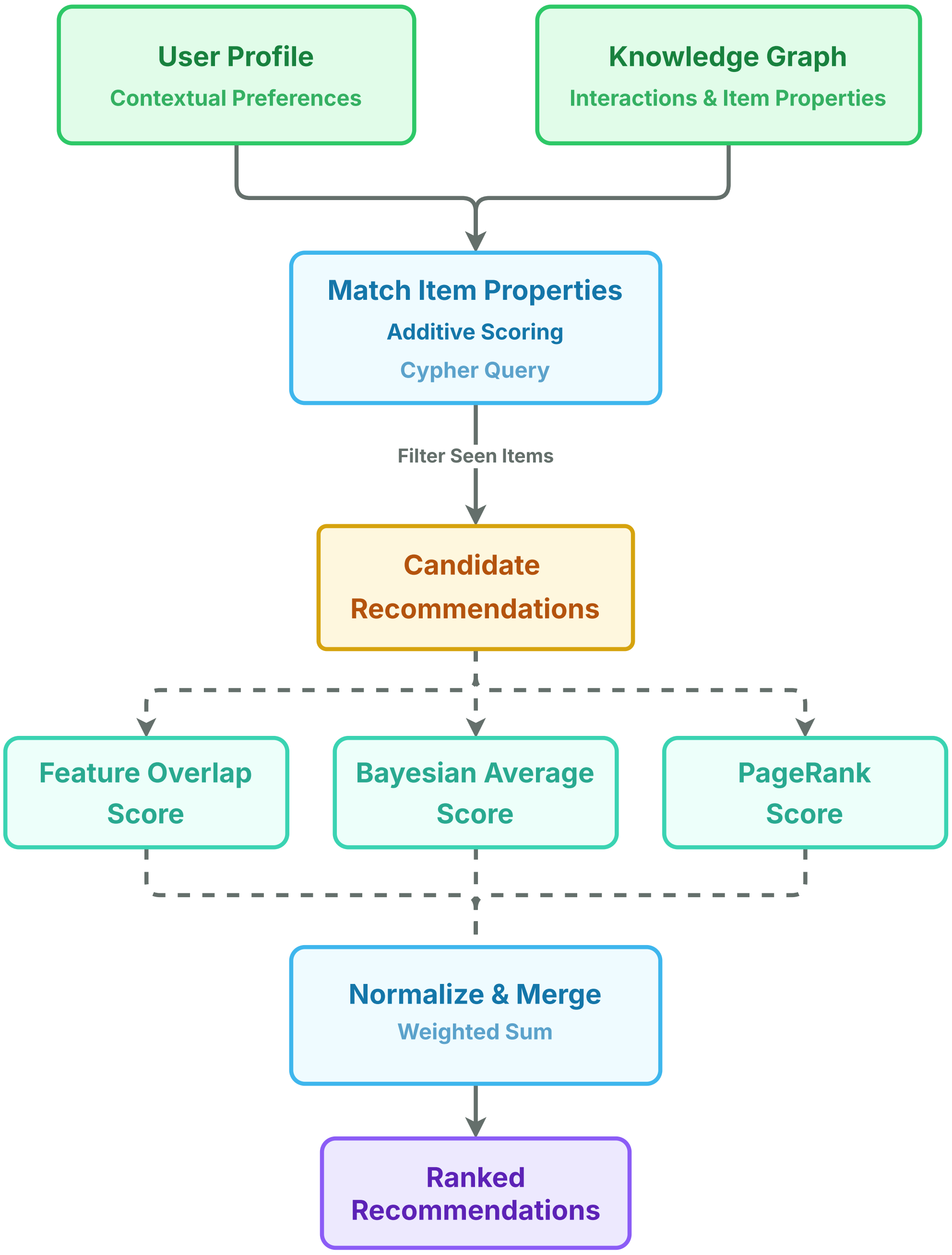


Figure 3.4: A diagram of the contextual recommendation process.

The process begins by translating the user's active contextual preferences into a Cypher query to retrieve items with matching properties from the KG. A representative example of this query is provided in Appendix A, Code A.1. After filtering out items previously seen by the user, each candidate item is evaluated using a weighted combination of three distinct signals:

- **Feature Overlap Score:** A score based on how well an item's attributes match the user's stated contextual preferences.
- **Bayesian Average Score:** To ensure that the popularity score is reliable and not skewed by items with very few ratings, a Bayesian Average is used. This formula, shown in Equation 3.1, smooths the average rating of an item by incorporating the global average rating across all items.

$$\text{BayesianAverage} = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C \tag{3.1}$$

  where v is the number of ratings for the item, R is the item's own average rating, m is a minimum number of ratings required for confidence, and C is the global average rating across all items.
- **PageRank Score:** To measure the structural importance of an item within the user-item interaction graph, the PageRank algorithm [58] is precomputed for all items, as described in Equation 3.2.

$$PR_i = \frac{1-d}{n} + d \sum_{j \in \{1,\ldots,n\}} \frac{PR_j}{c_j} \tag{3.2}$$

  where PR_i is the PageRank score of item i, n is the total number of items, d is a damping factor (typically 0.85), and the sum is over all items j that have a rating for item i, with c_j being the total number of outgoing ratings from item j. The final ranked list of items represents a context-aware set of recommendations that can immediately respond to the user's needs.

Hybrid Recommendations for Existing Users

For users with an established history of at least three rated items, the system enhances its recommendations by incorporating Collaborative Filtering and an expert model. In this mode, the final list of recommendations is a blend of results from three sources: the contextual method described previously, a Collaborative Filtering method, and the `EASER` model.

The Collaborative Filtering component, illustrated in Figure 3.5, identifies users with similar tastes to the target user through Cypher queries. It achieves this by finding "neighbors" in the KG who have positively rated the same items as the target user. The items liked by these neighbors, which the target user has not yet seen, are then considered as potential recommendations and ranked by their Bayesian average score. A representative example of these queries is included in Appendix A, Code A.2.

Finally, recommendations from the contextual and Collaborative Filtering methods are combined with a list of high-quality recommendations from the `EASER` model. The final set of recommendations is generated by interleaving the ranked lists from all three sources, ensuring a diverse yet personalized output.

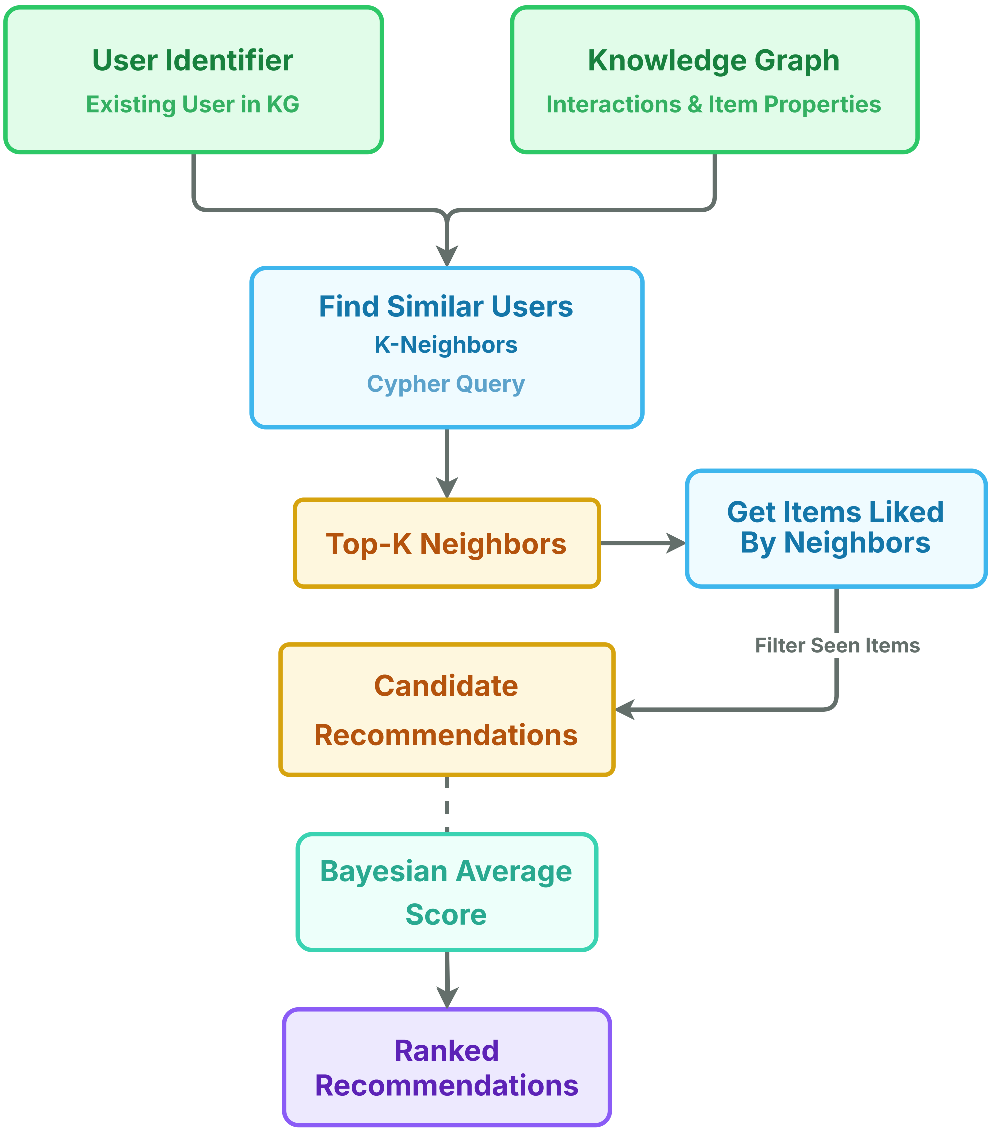


Figure 3.5: A diagram of the Collaborative Filtering recommendation process.

3.2.4. Integration of an Expert Recommender Model

For users with a more established interaction history, the system introduces a pre-trained expert recommender model. This component is responsible for providing deeply personalized recommendations by capturing the latent collaborative patterns in the entire user-item interaction dataset, thus addressing the second side of **RQ2**.

Recommendation Framework

The training, evaluation, tuning and serving of the expert model are managed using the RecBole [59] library, a comprehensive framework for building and evaluating Recommender Systems built on the PyTorch [60] machine learning library. RecBole was chosen due to its comprehensive and unified nature as an open-source library. It provides standardized implementations for a wide array of recommendation algorithms, along with tools for data processing, hyperparameter tuning, and reproducible evaluation. This makes it a suitable framework for conducting the research and experimentation required for this thesis, ensuring that the expert model component is both powerful and built upon a solid basis.

Another option, Cornac [61], was considered, but it was ultimately not selected due to its smaller set of supported algorithms and less extensive documentation compared to RecBole. The choice of RecBole ensures that there is a generalized configuration and dataset format compatibility across the different components of the framework, allowing for better integration and maintenance.

Model Selection

To select an appropriate expert model, preliminary tests were conducted with several candidate algorithms from the RecBole library. The evaluation was performed on the well-known `MovieLens-100k` dataset, with a focus on both recommendation accuracy and computational efficiency.

Based on these tests, the **EASER** model was selected, demonstrating the strongest performance in the preliminary evaluation by achieving high accuracy on key rank-aware metrics such as Recall@k and NDCG@k. These results are in line with recent findings from the community [62]. An additional advantage was its efficiency; the model trains quickly and the hyperparameter tuning process is streamlined, as it primarily depends on a single regularization weight parameter, `reg_weight`. The high accuracy and efficiency made `EASER` the most suitable choice for the expert model component of the hybrid system.

3.2.5. Generation of Natural Language Explanations

To address **RQ3**, a two-stage pipeline was implemented for generating natural language explanations for recommendations, as illustrated in Figure 3.6. This ensures that explanations are fluent, coherent, and grounded in the underlying data.

Stage 1: Symbolic Evidence Retrieval

The system queries the KG to identify human-interpretable reasoning paths linking a user to a recommended item. Evidence is retrieved according to a hierarchical process:

- **Content-based paths:** Items previously rated highly by the user are compared with the recommended item to identify shared features (e.g., `genre`, `category`). Both single-value attributes and sequential/multi-value features are considered. Evidence includes the specific items and attributes that support the recommendation.
- **Collaborative paths:** Multi-hop paths through other users are identified. Users who rated both the recommended item and items previously liked by the target user are counted, providing a social or collaborative explanation. Where applicable, overlapping features between these items and the recommended item are noted. Explanations quantify the number of similar users and highlight shared attributes.
- **Popularity signals:** If personal or collaborative evidence is insufficient, global metrics for the recommended item are retrieved, including the average rating and total number of interactions. This provides a fallback explanation based on general acceptance and quality.

Representative queries and implementation details are provided in Appendix A, Codes A.3, A.4, and A.5.

Stage 2: Natural Language Synthesis

Retrieved evidence is compiled into a structured prompt for the LLM. The model synthesizes all reasoning paths into a single, coherent explanation. This process ensures that the output is faithful to the retrieved evidence, integrating multiple evidence types into a cohesive, human-readable narrative, rather than producing generic or hallucinated statements.

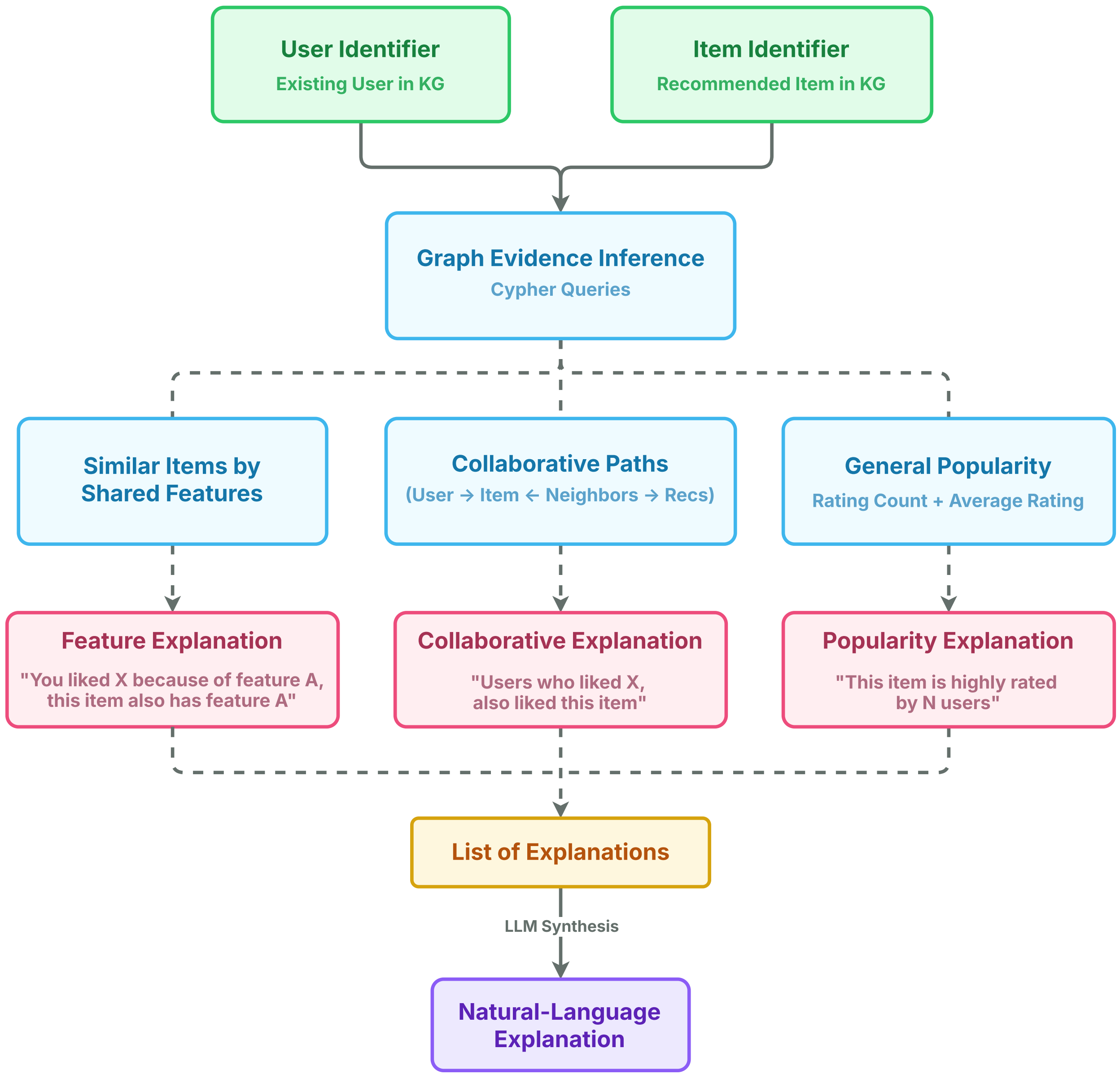

User Identifier
Existing User in KG
Item Identifier
Recommended Item in KG
Graph Evidence Inference
Cypher Queries
Similar Items by Shared Features
Collaborative Paths
(User → Item ← Neighbors → Recs)
General Popularity
Rating Count + Average Rating
Feature Explanation
"You liked X because of feature A, this item also has feature A"
Collaborative Explanation
"Users who liked X, also liked this item"
Popularity Explanation
"This item is highly rated by N users"
List of Explanations
LLM Synthesis
Natural-Language Explanation

Figure 3.6: The process for generating natural language explanations.

4 Experiments and Results

This chapter presents the empirical evaluation of the Hybrid Conversational Recommender System presented herein. The primary goal of the experiments is to validate the methodologies proposed in the previous chapter and to answer the core research questions concerning user profiling, recommendation quality, and explainability. The chapter begins by detailing the experimental environment and the datasets used for the evaluation. It then describes the specific experimental procedures, with a strong focus on the usability testing conducted with human participants. Finally, it presents and analyzes the results obtained from these experiments, providing both quantitative and qualitative insights into the platform's performance and user perception.

4.1. Environment

All experiments and evaluations described in this chapter were conducted on a single machine with the following hardware specifications, representing a typical modern developer-grade laptop:

- **CPU:** Intel® Core™ i7-11800H @ 2.30 GHz (16 Cores)
- **RAM:** 16 GB DDR4 @ 3200MHz
- **GPU:** NVIDIA® GeForce® RTX 3050 Laptop GPU with 4GB of VRAM

The software environment was managed entirely through Docker Compose. The setup included a containerized Ollama service for hosting the LLM, a FalkorDB database, and a FastAPI backend. For the performance tests detailed in the complementary thesis [6], the backend was configured to run with 4 workers to fully utilize the hardware; however, for the experiments interacting with real users, it was run with a single worker to accurately reflect the production deployment constraint.

4.2. Datasets

To evaluate the implemented system, two publicly available datasets from different domains were selected. These datasets were chosen to test the platform's data processing pipeline and to provide a basis for the user-centric evaluation of the conversational agents. The first is the `MovieLens-100k` dataset, a standard benchmark in Recommender Systems research. The second is a larger-scale dataset of user preferences for anime. The key statistics for both are summarized in Table 4.1.

Statistic	**MovieLens-100k**	**Anime**
Domain	Movies	Anime
# Users	943	73,515
# Items	1,682	12,294
# Interactions	100,000	7,813,737
Data Sparsity	93.7 %	99.1 %

Table 4.1: Key statistics of the datasets used in the experiments.

4.2.1. MovieLens

The **MovieLens-100k** dataset is a classic benchmark for evaluating recommendation algorithms, collected by the GroupLens Research group. It was selected for this study due to its manageable size for rapid testing and its rich feature set, which is well-suited for evaluating preference elicitation and explanation generation.

The dataset consists of 100,000 ratings (from 1 to 5) from 943 users on 1,682 movies. In addition to the interaction data, it provides metadata for both users (age, gender, occupation) and items (movie title, release year, and genres). The dataset is notably sparse, with over 93 % of possible user-item interactions being absent, which makes it a realistic test case for recommendation challenges like the cold-start problem.

4.2.2. Anime

To evaluate the system on a larger scale, an **Anime** dataset containing user preference data from the anime tracking website `https://myanimelist.net` was used. This dataset is significantly larger and sparser than MovieLens-100k, providing a different set of challenges for the recommendation and conversational agents.

It contains over 7.8 million ratings from tens of thousands of users on 12,294 unique anime titles. Unlike MovieLens, it does not contain user-specific metadata. However, it provides a rich set of item features, including the anime's name, genres, type (e.g., TV, movie), number of episodes, overall average rating, and the number of users who have it on their list (a measure of popularity). This feature set is ideal for testing the system's ability to handle content-based and contextual recommendations within a large item catalog.

4.3. Experiments

To evaluate the implemented system and address the research questions, a user-centric experimental methodology was designed. As the research focuses on the quality of the human-computer interaction, a comprehensive usability study was conducted with human participants. This approach allows for the collection of both quantitative and qualitative data regarding the platform's performance from the user's perspective. These tests were carried out on the `HybridCRS` web platform, a modern, responsive application for creating and chatting with recommender agents, developed for the complementary engineering thesis [6]. The following subsections detail the usability testing protocol and the specific criteria used to evaluate the quality of the recommendations and their explanations.

4.3.1. Usability Testing

Usability testing was carried out to evaluate the `HybridCRS` platform's user experience, focusing on its usability and overall performance. The testing involved a group of participants who interacted with the system, providing feedback on various aspects such as recommendation quality, usability, and design. The System Usability Scale [63] was employed to quantify the usability of the platform, a ten-item attitude Likert scale giving a global view of subjective assessments of usability. The complete questionnaire used for these tests is included in Appendix B.

Participant Demographics

A total of 11 participants took part in the usability testing study. The participants were between 23 and 24 years old, with an average age of approximately 23.5 years. The group consisted of 10 male participants and one female participant. Most participants reported having prior experience with LLMs and chatbots: 10 of the 11 had experience with LLM chat interfaces, and all 11 had experience using chatbots. This sample reflects a relatively homogeneous group in terms of age and high technological familiarity.

Guided Testing

The core of the study involved participants completing three guided tasks, each designed to test a key functionality of the platform. After each task, participants were asked to rate its perceived difficulty on a 5-point Likert scale, where 1 was "Very Easy" and 5 was "Very Difficult". The tasks were as follows:

- **Use Case 1: Agent Creation and Editing.** Participants were instructed to download a sample dataset, create a new recommender agent via the platform's multi-step creation wizard, and then edit the agent's metadata (e.g., name, description) to verify that the changes were reflected in the Agent Hub.
- **Use Case 2: First Conversation.** Participants were asked to initiate a conversation with the agent

they had just created. The task involved engaging in a dialogue, providing preferences to the agent until recommendations were received, and then providing feedback on those recommendations.

- **Use Case 3: Retraining and Subsequent Conversation.** The final task required participants to use the "Retrain" function on their agent, which incorporates the feedback from the previous session into the expert model. After the agent finished retraining, they were asked to start a new conversation to see if the recommendations were updated.

System Usability Scale (SUS)

The System Usability Scale is a standardized and widely validated tool for measuring perceived usability. It consists of a 10-item questionnaire with five response options, ranging from "Strongly Disagree" to "Strongly Agree". After completing all three guided tasks, participants were asked to fill out the SUS questionnaire. The final score, ranging from 0 to 100, is calculated using the standard formula, where odd-numbered questions represent positive statements and even-numbered questions represent negative statements.

$$SUS = 2{,}5 \times \left(\sum_{i \in \{1,3,5,7,9\}} (score_i - 1) + \sum_{j \in \{2,4,6,8,10\}} (5 - score_j) \right) \tag{4.1}$$

4.3.2. Recommendation Evaluation

The quality of the recommendations generated by the system was evaluated qualitatively as part of the user study. After completing the conversational tasks (Use Cases 2 and 3), participants were asked to rate the recommendations they received on a 10-point scale across four distinct criteria:

- **Accuracy:** The general relevance and appeal of the recommended items to the user.
- **Contextual Relevance:** How well the recommendations matched the specific preferences and constraints stated during the conversation.
- **Novelty:** The degree to which the recommendations were new, unusual, or not simply popular, mainstream items.
- **Diversity:** The variety of the recommended items in terms of their attributes (e.g., genre).

4.3.3. Explainability Evaluation

Similarly, the impact and quality of the natural language explanations were evaluated qualitatively. For each conversational task that resulted in recommendations (Use Cases 2 and 3), participants rated the accompanying explanations on a 10-point scale according to the online metrics introduced in

Section 2.4.3: **Transparency**, **Satisfaction**, **Trust/Persuasiveness**, and **Cognitive Load**.

4.4. Results

This section presents the analysis of the data collected from the usability study described in the previous chapter. The results are organized into three main parts. First, an overview of the platform's general usability is presented, including the System Usability Scale score and overall user ratings. This is followed by a qualitative analysis of user satisfaction and the constructive feedback gathered. Finally, a quantitative comparison of the perceived quality of recommendations and explanations between the initial (cold-start) and post-retraining conversational scenarios is detailed.

4.4.1. System Usability

The overall usability of the `HybridCRS` platform was measured using the SUS questionnaire and a general rating scale. The platform achieved a final SUS score of **92 out of 100**, with a standard deviation of 4.76. This score is well above the average of 68 and falls into the highest percentile, indicating an excellent level of perceived usability. The detailed breakdown of the average scores for each of the ten SUS questions is visualized in Figure 4.1.

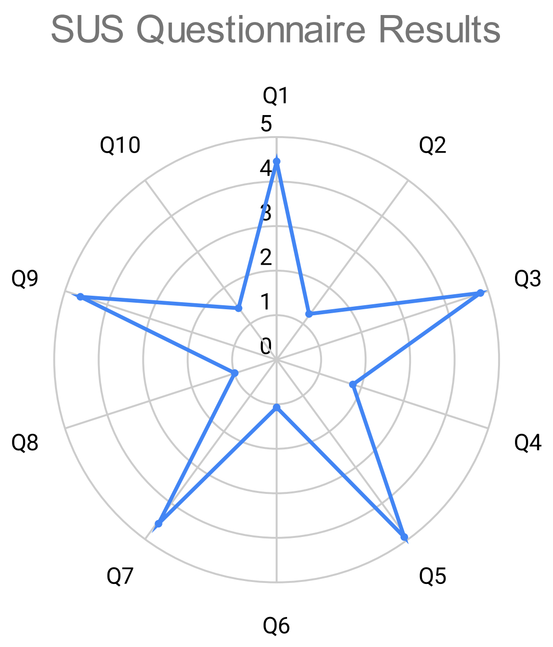


Figure 4.1: Average scores for each of the ten SUS questions (1-5 scale).

This strong result was reinforced by the participants' general rating of the platform, which received an average score of **9.36 out of 10** (standard deviation of 0.81). This suggests a consistent and highly positive user experience across the participant group. While the overall usability was rated highly, the individual SUS question scores indicate that some participants felt they might need some initial assistance or time to learn the more complex functionalities, such as the agent creation wizard.

User Satisfaction

The qualitative feedback gathered from participants indicates a high degree of user satisfaction with the platform's core functionalities. The user interface was consistently described as clear, intuitive, and aesthetically appealing. Users reported that the system felt stable and was easy to use, even for those without a technical background.

A notable point of satisfaction was the quality of the recommendations. Participants found the suggestions to be relevant and often satisfying. The system's ability to recommend lesser-known items was also noted, suggesting a capacity for novelty that goes beyond mainstream popularity. The dual-agent approach, providing both a structured recommender agent and a flexible open chat, was also highlighted as a positive feature.

User Feedback

In addition to the positive feedback, the usability study yielded several points of constructive feedback and valuable suggestions for improvement. A recurring theme was that in initial conversational turns (the cold-start phase), the recommendations could sometimes be generic and focused on popular items. Users also noted that the agent's ability to interpret context was limited when prompts included specific attributes not present in the item dataset.

Based on this feedback, several areas for future enhancement were identified:

- **Improved Context Awareness:** Enhancing the LLM's ability to handle queries that fall outside the dataset's explicit schema.
- **Recommendation Transparency:** Adding labels or tags to recommendations to clarify their origin (e.g., Collaborative Filtering vs. Content-based).
- **Better Feedback Interface:** Modifying the rating interface for unseen items, for instance, by first asking for confirmation of viewership before presenting a rating slider.
- **User Onboarding:** Implementing a brief tutorial or context-sensitive tooltips to guide new users through more complex features like the agent creation process.

4.4.2. Recommendation Accuracy

To evaluate the impact of the system's learning capabilities, the perceived quality of recommendations was compared between the initial cold-start conversation (Use Case 2) and the conversation after the agent was retrained with the user's feedback (Use Case 3). The average scores for each metric are visualized in Figure 4.2 and presented in Table 4.2.

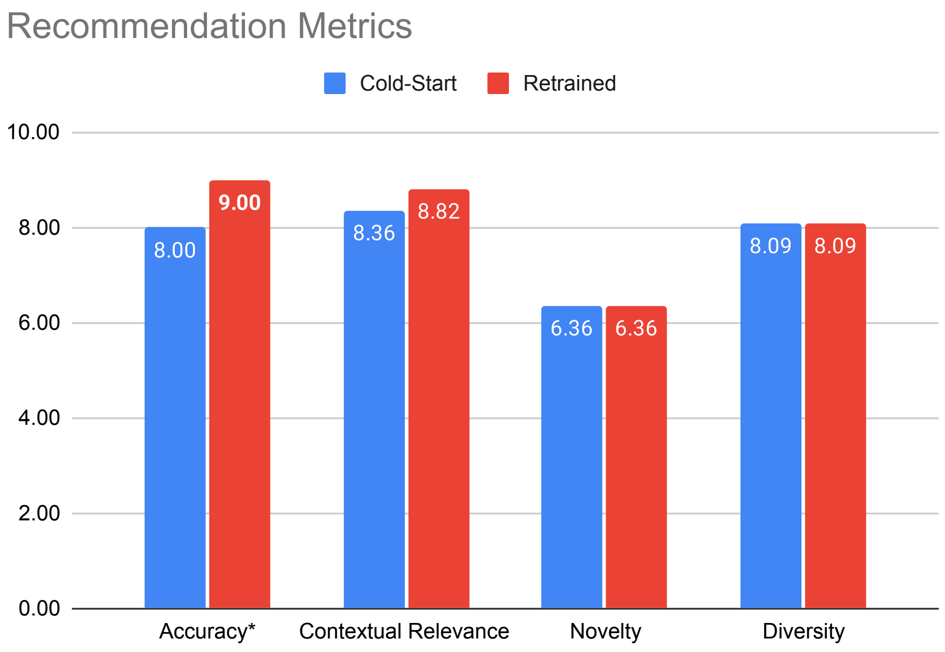


Figure 4.2: Comparison of recommendation evaluation results.

Metric	Cold-Start (UC2)	Retrained (UC3)
Accuracy*	8.00	9.00
Contextual Relevance	8.36	8.82
Novelty	6.36	6.36
Diversity	8.09	8.09

*Indicates a statistically significant improvement ($p < 0{,}05$)

Table 4.2: Average scores for recommendation quality (1-10 scale).

A one-tailed, paired-samples t-test was conducted to assess the statistical significance of these changes. The results show a **statistically significant improvement in perceived Accuracy** after retraining (p = 0.042), indicating that users found the recommendations to be more relevant after the agent had learned from their initial feedback. While there was also a slight increase in Contextual Relevance, this change, along with Novelty and Diversity, was not found to be statistically significant.

4.4.3. Explainability Impact

The perceived quality of the explanations was also compared between the cold-start (Use Case 2) and post-retraining (Use Case 3) scenarios. The average scores for each explanation metric are visualized in Figure 4.3 and shown in Table 4.3.

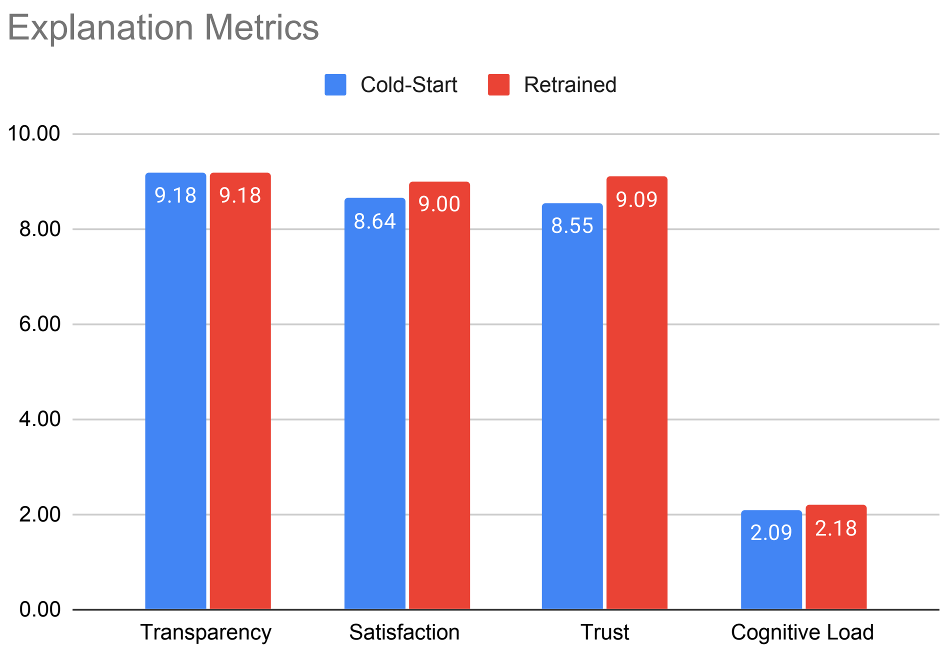


Figure 4.3: Comparison of explanation evaluation results.

Metric	Cold-Start (UC2)	Retrained (UC3)
Transparency	9.18	9.18
Satisfaction	8.64	9.00
Trust / Persuasiveness	8.55	9.09
Cognitive Load	2.09	2.18

Table 4.3: Average scores for explanation quality (1-10 scale).

The results indicate that the explanations were rated very highly in both scenarios, particularly in terms of Transparency and low Cognitive Load. After retraining, there were observable improvements in user Satisfaction and Trust. However, a one-tailed, paired-samples t-test revealed that none of these changes were statistically significant. This suggests that while the retraining process improved the accuracy of the underlying recommendations, the graph-based method for generating explanations was perceived as consistently effective in both the cold-start and warm-start phases, with a slight increase in Satisfaction and Trust in during the latter.

5 Conclusions and Future Work

5.1. Conclusions

This research work sets out to investigate and develop a framework for a hybrid CRS that prioritizes dynamic user profiling and transparent, explainable recommendations. The primary contribution of this thesis is the successful implementation and user-centric evaluation of a novel system that demonstrates the viability of combining conversational AI with graph-based reasoning and expert recommender models. The findings of the experimental evaluation provide answers to the core research questions posed at the outset of this document.

Regarding **RQ1**, the study demonstrated that a dynamic user profile, constructed in real-time from natural language conversation, is an effective method for improving personalization. The high user satisfaction ratings and low task difficulty scores from the usability study indicate that the conversational preference elicitation process was both intuitive and successful in capturing user needs.

In response to **RQ2**, the results affirm that orchestrating a hybrid recommendation strategy is an effective approach. The statistically significant improvement in perceived recommendation accuracy after retraining the expert model ($p < 0{,}05$) validates the benefit of integrating a deep, personalized model for established users, while the high contextual relevance scores in the cold-start phase demonstrate the effectiveness of the graph-based engine.

Finally, concerning **RQ3**, the research shows that leveraging graph-based reasoning to generate natural language explanations has a significant positive impact on the user experience. The explanations received exceptionally high scores for transparency and trust, and a relatively low score for cognitive load, suggesting that users found them clear, persuasive, and easy to understand. This solution effectively proves that graph-based reasoning can be applied even to black box Recommender Systems, enhancing user trust and understanding without necessarily compromising the system's performance.

While the study's findings are promising, it is important to acknowledge its limitations. The usability study was conducted with a small and relatively homogeneous group of technologically experienced participants. A broader study would be needed to generalize these findings to a wider population.

Furthermore, the evaluation focused on user-perceived quality rather than offline quantitative metrics.

The project repository with the code for the `HybridCRS` web platform, detailed in the complementary engineering thesis [6], can be found at `https://github.com/Acervans/Hybrid-CRS`.

5.2. Future Work

While the current platform provides a robust and functional foundation, the research opens up numerous avenues for future investigation and development. The following points outline potential directions for extending this work.

- **Methodological Enhancements:**
    1. **Advanced User Modeling:** The current user profile is session-based. Future work could explore methods for building persistent, long-term user profiles that evolve across multiple conversations, potentially capturing preference drift and incorporating a wider array of user and interaction features.
    2. **Dialogue Policy Optimization:** A significant advancement would be to incorporate Reinforcement Learning (RL) to train a specific dialogue policy. This would allow the agent to learn the optimal strategy for when to ask questions versus when to recommend, with the goal of maximizing long-term user satisfaction.
    3. **Testing with Advanced Expert Models:** The current system uses `EASER` as its expert model. Future research could evaluate the integration of other, more complex models, particularly context-aware recommenders that could more dynamically leverage the state of the conversational user profile.
    4. **Expanded Agent Toolset:** The agent's capabilities could be broadened by providing it with a larger set of tools. This could include functions for more complex multi-modal interactions, deeper data analysis, or integrations with external knowledge sources beyond simple web search.
    5. **Multilingual Capabilities:** Currently, the conversational workflow operates mainly in English. An interesting research direction would be to explore the development of multilingual agents. This would involve evaluating the performance of multilingual LLMs and adapting the prompt engineering and user profiling techniques to handle cross-lingual conversations and recommendations.
- **Evaluation & Experiments:**
    1. **Quantitative Offline Evaluation:** A critical next step is to conduct a large-scale offline evaluation of the hybrid recommendation strategy using standard accuracy metrics (e.g., NDCG@k, Recall@k) to quantitatively benchmark its performance.
    2. **Evaluation with Larger Language Models:** The experiments in this thesis were conducted with a moderately-sized local LLM. Re-running the user studies with more powerful models,

either larger open-source models on more capable hardware or proprietary models via an API (e.g., GPT-5), would provide valuable insights into how model scale impacts the quality of conversation, recommendations, and explanations.

3. **Diverse User Studies:** To validate the findings of the initial usability study, future experiments should be conducted with a larger and more demographically diverse group of participants.

- **User Feedback:** The constructive feedback from the usability study provides a clear roadmap for UI improvements, such as adding an onboarding tutorial for agent creation and providing explicit labels that indicate the source of a given recommendation.

Bibliography

[1] D. Pramod and P. Bafna, “Conversational recommender systems techniques, tools, acceptance, and adoption: A state of the art review,” *Expert Systems with Applications*, vol. 203, p. 117539, 2022.

[2] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, and Q. Li, “Recommender Systems in the Era of Large Language Models (LLMs),” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, p. 6889–6907, Nov. 2024.

[3] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Springer, 2022.

[4] O. Sar Shalom, H. Roitman, and P. Kouki, *Natural Language Processing for Recommender Systems*, pp. 447–471. In [3], 2022.

[5] A. Vultureanu-Albişi and C. Bădică, “Recommender systems: An explainable AI perspective,” in *2021 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, IEEE, 2021.

[6] J. W. Zhou, “A Hybrid Conversational Recommender System by integrating LLMs: Development of a Scalable Platform for Conversational Recommendations,” 2025. Master’s Thesis, Universidad Autónoma de Madrid.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[8] OpenAI, “ChatGPT.” `https://www.chatgpt.com/`, 2023.

[9] Google DeepMind, “Gemini.” `https://www.deepmind.com/research/`, 2023.

[10] Anthropic, “Claude.” `https://www.anthropic.com/`, 2023.

[11] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, “Retrieval-Augmented Generation for Large Language Models: A Survey,” 2024.

[12] H. Chase, “LangChain.” `https://github.com/langchain-ai/langchain`, October 2022.

[13] J. Liu, “LlamaIndex.” `https://github.com/jerryjliu/llama_index`, 11 2022.

[14] M. Pietsch, T. Möller, B. Kostic, J. Risch, M. Pippi, M. Jobanputra, S. Zanzottera, S. Cerza, V. Blagojevic, T. Stadelmann, T. Soni, and S. Lee, “Haystack: the end-to-end NLP framework for pragmatic builders.” `https://github.com/deepset-ai/haystack`, November 2019.

[15] Amazon Web Services, “What is Retrieval-Augmented Generation?.” `https://aws.amazon.com/what-is/retrieval-augmented-generation/`, 2025. Accessed 2025-08-09.

[16] S. Karzhev, “Advanced RAG Techniques,” 9 2024.

[17] Z. Qiu, L. Luo, Z. Zhao, S. Pan, and A. W.-C. Liew, "Graph Retrieval-Augmented LLM for Conversational Recommendation Systems," 2025.

[18] Y. Li, K. He, Z. Wang, Y. Li, J. Li, J.-Y. Nie, and J.-R. Wen, "G-Refer: Graph Retrieval-Augmented Large Language Model for Explainable Recommendation," 2025.

[19] Y. Zhu, C. Wan, H. Steck, D. Liang, Y. Feng, N. Kallus, and J. Li, "Collaborative Retrieval for Large Language Model-based Conversational Recommender Systems," in *Proceedings of the ACM on Web Conference 2025*, WWW '25, (New York, NY, USA), p. 3323–3334, Association for Computing Machinery, 2025.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[21] W. L. Cheng, "Introducing post-training model quantization feature and mechanics explained," *Datature Blog*, Jan. 2024.

[22] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," 2023.

[23] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration," in *MLSys*, 2024.

[24] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[25] C. Fourrier, N. Habib, A. Lozovskaya, K. Szafer, and T. Wolf, "Open LLM Leaderboard v2." `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`, 2024.

[26] S. G. Patil, H. Mao, C. Cheng-Jie Ji, F. Yan, V. Suresh, I. Stoica, and J. E. Gonzalez, "The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models," in *Forty-second International Conference on Machine Learning*, 2025.

[27] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen Technical Report," *arXiv preprint arXiv:2309.16609*, 2023.

[28] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, p. 143–177, Jan. 2004.

[29] T. Kitazawa, "Incremental Factorization Machines for Persistently Cold-starting Online Item Recommendation," 2016.

[30] S. Kim, H. Kang, S. Choi, D. Kim, M. Yang, and C. Park, "Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, (New York,

NY, USA), p. 1395–1406, Association for Computing Machinery, 2024.

[31] H. Steck, “Embarrassingly Shallow Autoencoders for Sparse Data,” in *The World Wide Web Conference*, WWW ’19, p. 3251–3257, ACM, May 2019.

[32] A. Gunawardana, G. Shani, and S. Yogev, *Evaluating Recommender Systems*, pp. 547–601. In [3], 2022.

[33] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland* (K. Järvelin, M. Beaulieu, R. A. Baeza-Yates, and S. Myaeng, eds.), pp. 253–260, ACM, 2002.

[34] O. Jeunen, J. Van Balen, and B. Goethals, “Embarrassingly shallow auto-encoders for dynamic collaborative filtering,” *User Modeling and User-Adapted Interaction*, vol. 32, p. 509–541, Sept. 2022.

[35] J. Frej, M. Knezevic, and T. Kaser, “Graph Reasoning for Explainable Cold Start Recommendation,” 2024.

[36] D. Jannach, A. Manzoor, W. Cai, and L. Chen, “A survey on conversational recommender systems,” *ACM Comput. Surv.*, vol. 54, May 2021.

[37] K. Christakopoulou, F. Radlinski, and K. Hofmann, “Towards conversational recommender systems,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, (New York, NY, USA), p. 815–824, Association for Computing Machinery, 2016.

[38] D. Jannach, “Evaluating conversational recommender systems,” *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2365–2400, 2023.

[39] U. Maes, L. Michiels, and A. Smets, “GenUI(ne) CRS: UI Elements and Retrieval-Augmented Generation in Conversational Recommender Systems with LLMs,” in *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys ’24, (New York, NY, USA), p. 1177–1179, Association for Computing Machinery, 2024.

[40] K. Zhou, X. Wang, Y. Zhou, C. Shang, Y. Cheng, W. X. Zhao, Y. Li, and J.-R. Wen, “CRSLab: An open-source toolkit for building conversational recommender system,” 2021.

[41] C. Li, H. Hu, Y. Zhang, M.-Y. Kan, and H. Li, “A Conversation is Worth A Thousand Recommendations: A Survey of Holistic Conversational Recommender Systems,” in *Proceedings of the Fifth Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop*, 2023.

[42] L. Wang, H. Hu, L. Sha, C. Xu, D. Jiang, and K.-F. Wong, “RecInDial: A Unified Framework for Conversational Recommendation with Pretrained Language Models,” in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pp. 489–500, 2022.

[43] Y. Deng, Y. Li, F. Sun, B. Ding, and W. Lam, “Unified conversational recommendation policy learning via graph-based reinforcement learning,” in *Proceedings of the 44th International ACM SIGIR*

Conference on Research and Development in Information Retrieval, SIGIR '21, (New York, NY, USA), p. 1431–1441, Association for Computing Machinery, 2021.

[44] W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Conversational recommendation: Formulation, methods, and evaluation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, (New York, NY, USA), p. 2425–2428, Association for Computing Machinery, 2020.

[45] W. Yang, Y. Wei, H. Wei, Y. Chen, G. Huang, X. Li, R. Li, N. Yao, X. Wang, X. Gu, M. B. Amin, and B. Kang, "Survey on explainable AI: From approaches, limitations and applications aspects," *Human-Centric Intelligent Systems*, vol. 3, no. 3, pp. 161–188, 2023.

[46] F. Sovrano, S. Sapienza, M. Palmirani, and F. Vitali, "A survey on methods and metrics for the assessment of explainability under the proposed AI act," *CoRR*, vol. abs/2110.11168, 2021.

[47] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 4768–4777, Curran Associates Inc., 2017.

[48] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 1135–1144, Association for Computing Machinery, 2016.

[49] Y. Chen and J. Miyazaki, "A Model-Agnostic Recommendation Explanation System Based on Knowledge Graph," in *Database and Expert Systems Applications* (S. Hartmann, J. Küng, G. Kotsis, A. M. Tjoa, and I. Khalil, eds.), pp. 149–163, Springer International Publishing, 2020.

[50] M. H. Syed, T. Q. B. Huy, and S.-T. Chung, "Context-aware explainable recommendation based on domain knowledge graph," *Big Data and Cognitive Computing*, vol. 6, no. 1, 2022.

[51] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, and Y. Zhang, "Counterfactual explainable recommendation," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, (New York, NY, USA), p. 1784–1793, Association for Computing Machinery, 2021.

[52] D. Yu, Q. Li, X. Wang, Q. Li, and G. Xu, "Counterfactual explainable conversational recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 6, pp. 2388–2400, 2024.

[53] Y. Lei, J. Lian, J. Yao, X. Huang, D. Lian, and X. Xie, "RecExplainer: Aligning Large Language Models for Explaining Recommendation Models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, p. 1530–1541, ACM, Aug. 2024.

[54] Z. Chen, X. Wang, X. Xie, M. Parsana, A. Soni, X. Ao, and E. Chen, "Towards Explainable Conversational Recommendation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20* (C. Bessiere, ed.), pp. 2994–3000, International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[55] G. Balloccu, L. Boratto, G. Fenu, F. M. Malloci, and M. Marras, "Explainable Recommender Systems with Knowledge Graphs and Language Models," in *Advances in Information Retrieval*

(N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, and I. Ounis, eds.), pp. 352–357, Springer Nature Switzerland, 2024.

[56] F. d. S. Silva, “Segment, Recommend, and Explain: Advancing Conversational Recommender Systems with Large Language Model Agents,” in *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, UMAP ’25, p. 404–408, Association for Computing Machinery, 2025.

[57] S. Guo, S. Zhang, W. Sun, P. Ren, Z. Chen, and Z. Ren, “Towards Explainable Conversational Recommender Systems,” in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’23, p. 2786–2795, Association for Computing Machinery, 2023.

[58] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” tech. rep., Stanford Digital Library Technologies Project, 1998.

[59] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, and J.-R. Wen, “RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms,” 2021.

[60] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[61] A. Salah, Q.-T. Truong, and H. W. Lauw, “Cornac: a comparative framework for multimodal recommender systems,” *J. Mach. Learn. Res.*, vol. 21, Jan. 2020.

[62] V. W. Anelli, A. Bellogín, T. D. Noia, D. Jannach, and C. Pomo, “Top-n recommendation algorithms: A quest for the state-of-the-art,” in *UMAP ’22: 30th ACM Conference on User Modeling, Adaptation and Personalization, Barcelona, Spain, July 4 - 7, 2022* (A. Bellogín, L. Boratto, O. C. Santos, L. Ardissono, and B. P. Knijnenburg, eds.), pp. 121–131, ACM, 2022.

[63] J. B. Brooke, “SUS: A ’Quick and Dirty’ Usability Scale,” 1996.

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
BFCL	Berkeley Function Calling Leaderboard.
CF	Collaborative Filtering.
CPU	Central Processing Unit.
CRS	Conversational Recommender System.
DCG	Discounted Cumulative Gain.
FM	Factorization Machines.
GPT	Generative Pre-trained Transformer.
GPU	Graphics Processing Unit.
IDCG	Ideal Discounted Cumulative Gain.
KG	Knowledge Graph.
LLM	Large Language Model.
MAE	Mean Absolute Error.
NDCG	Normalized Discounted Cumulative Gain.
RAG	Retrieval-Augmented Generation.
RAM	Random Access Memory.
RMSE	Root Mean Squared Error.
RS	Recommender System.
SUS	System Usability Scale.
XAI	Explainable Artificial Intelligence.

Appendices

A

FalkorDB Cypher Queries

Code A.1: Example of Cypher query to generate contextual recommendations from user preferences.

```
// Goal: Recommend items based on context, popularity, and authority.
// Parameters:
// -User ID: 'user_A'
// -Item Properties (Context): category = 'Books', genre = 'Sci-Fi'
// -Seen Items: ['item_101', 'item_205'] (Items to exclude)
// -MIN_R: 10 (Minimum ratings for an item to be considered reliable)
// -Global Average Rating: 3.5

// Step 1: Find all items that match the desired contextual properties.
// We also filter out items the user has already interacted with.
MATCH (itm:Item)
WHERE (itm.category = 'Books' OR itm.genre = 'Sci-Fi')
  AND NOT itm.item_id IN ['item_101', 'item_205']

// Step 2: For each matched item, calculate its average rating and rating count.
// OPTIONAL MATCH is used because an item might not have any ratings yet.
OPTIONAL MATCH ()-[r:RATED]->(itm)
WITH itm, avg(r.rating) AS avgRating, count(r) AS cnt

// Step 3: Calculate the three scores for ranking.
// The final ranking will be done in the application layer after normalizing these scores.
RETURN
    itm.item_id AS itemId,

    // Score 1: Context Score. A simple count of how many desired properties match.
    (CASE WHEN itm.category = 'Books' THEN 1 ELSE 0 END + CASE WHEN itm.genre = 'Sci-Fi'
         THEN 1 ELSE 0 END) AS contextScore,

    // Score 2: Weighted Score. This is a smoothed average rating.
    // It pulls the item's average rating towards the global average,
    // which gives more reliable scores for items with few ratings.
    ((cnt *avgRating) + (10 *3.5)) / (cnt + 10) AS weightedScore,

    // Score 3: PageRank. A measure of the item's authority or importance in the graph.
    // This is pre-calculated and stored as a node property.
    itm.pagerank AS pageRank
```

Code A.2: Example of Cypher query to generate Collaborative Filtering recommendations based on closest neighbors.

```
// Goal: Recommend items based on the preferences of similar users (neighbors).
// Parameters:
// -User ID: 'user_A'
// -Min Rating for "Like": 3.0
// -K (Neighbors): 10
// -Top-N (Recommendations): 10
// -Seen Items: ['item_101', 'item_205']
// -Global Average Rating: 3.5

// ---Part 1: Find Top-K Similar Users (Neighbors) ---

// Step 1: Find a user (u1) and identify other users (u2) who have rated the same items (i).
MATCH (u1:User {user_id: 'user_A'})-[r1:RATED]->(i:Item)<-[r2:RATED]-(u2:User)
// We only consider items that both users "liked" (rating >= 3.0).
WHERE r1.rating >= 3.0 AND r2.rating >= 3.0 AND u1 <> u2

// Step 2: Count the number of co-rated items for each user (u2) to measure similarity.
WITH u2, count(i) AS sharedItems
// Order by the count to find the most similar users.
ORDER BY sharedItems DESC
// Limit to the top K neighbors.
LIMIT 10

// Step 3: Collect the IDs of these neighbors to use in the next stage.
WITH collect(u2.user_id) AS neighborIds

// ---Part 2: Recommend Items from Neighbors ---

// Step 4: Find all items that the neighbors liked.
MATCH (neighbor:User)-[r:RATED]->(i:Item)
WHERE neighbor.user_id IN neighborIds
  AND r.rating >= 3.0
  // Exclude items the target user has already seen.
  AND NOT i.item_id IN ['item_101', 'item_205']

// Step 5: Calculate a weighted score for each potential recommendation.
WITH i, avg(r.rating) AS avgRating, count(r) AS cnt
WITH i, ((cnt *avgRating) + (10 *3.5)) / (cnt + 10) AS weightedScore

// Step 6: Return the top N recommended items, ordered by their score.
RETURN i.item_id AS recommendation, weightedScore
ORDER BY weightedScore DESC
LIMIT 10
```

Code A.3: Example of Cypher query to explain any recommendation by feature similarity.

```
// Goal: Find items liked by the user that share a specific property
// (e.g., 'category', 'brand') with the recommended item.
//
// This query runs in a loop for each property in `shared_props`.
// Parameters:
// -$user_id: The ID of the target user (e.g., 'user_A').
// -$min_rating: The minimum rating to consider an item "liked" (e.g., 4.0).
// -$top_feat_exp: The max number of examples to return (e.g., 5).
// -The WHERE clause (i.{property} = {value}) is built dynamically.

// Case 1: Regular property (exact match)
MATCH (u:User {user_id: $user_id})-[r:RATED]->(i:Item)
// The condition below is constructed by the Python code.
// For example, if the recommended item's category is 'Books',
// the condition becomes: i.category = 'Books'
WHERE r.rating >= $min_rating AND i.brand = 'Nintendo'
// Return the shared property value and the full item node.
// The application uses this to construct the explanation string.
RETURN i.brand AS value, i
ORDER BY r.rating DESC
LIMIT $top_feat_exp;

// Case 2: Sequential property (contains match)
MATCH (u:User {user_id: $user_id})-[r:RATED]->(i:Item)
WHERE r.rating >= $min_rating AND (i.category CONTAINS 'Action' OR i.category CONTAINS
    'Adventure')
RETURN i.category AS value, i
ORDER BY r.rating DESC
LIMIT $top_feat_exp;
```

Code A.4: Example of Cypher query to explain any recommendation by collaborative evidence.

```
// Goal: Find social proof by identifying "similar users" who also liked the recommended item.
// Path: (Target User)-[:LIKED]->(Shared Item)<-[:LIKED]-(Similar User)-[:LIKED]->(Recommended
    Item)
//
// Parameters:
// -$user_id: The ID of the target user (e.g., 'user_A').
// -$item_id: The ID of the recommended item (e.g., 'item_550').
// -$min_rating: The rating threshold for a "liked" item (e.g., 3.0).
// -$similar_item_ids: A list of item IDs found in the Feature Similarity step.
// -$top_collab_exp: The max number of collaborative explanations (e.g., 5).

// Step 1: Match the full 3-hop collaborative path.
MATCH (u:User {user_id:
    $user_id})-[r1:RATED]->(i:Item)<-[r2:RATED]-(u2:User)-[r3:RATED]->(rec:Item {item_id:
    $item_id})
WHERE r1.rating >= $min_rating
  AND r2.rating >= $min_rating
  AND r3.rating >= $min_rating

// Step 2: For each shared item 'i', count the supporting users and check if 'i'
// was also found in the feature similarity step (passed in as a parameter).
WITH
  count(DISTINCT u2) AS numUsers,
  i AS shared_item,
  (i.item_id IN $similar_item_ids) AS sharesProp

// Step 3: Return the evidence, prioritizing paths that also have shared features.
RETURN
  numUsers,
  shared_item,
  sharesProp
ORDER BY
  sharesProp DESC, // Prioritize items that also explain via feature similarity
  numUsers DESC
LIMIT $top_collab_exp
```

Code A.5: Example of Cypher query to explain any recommendation by general popularity.

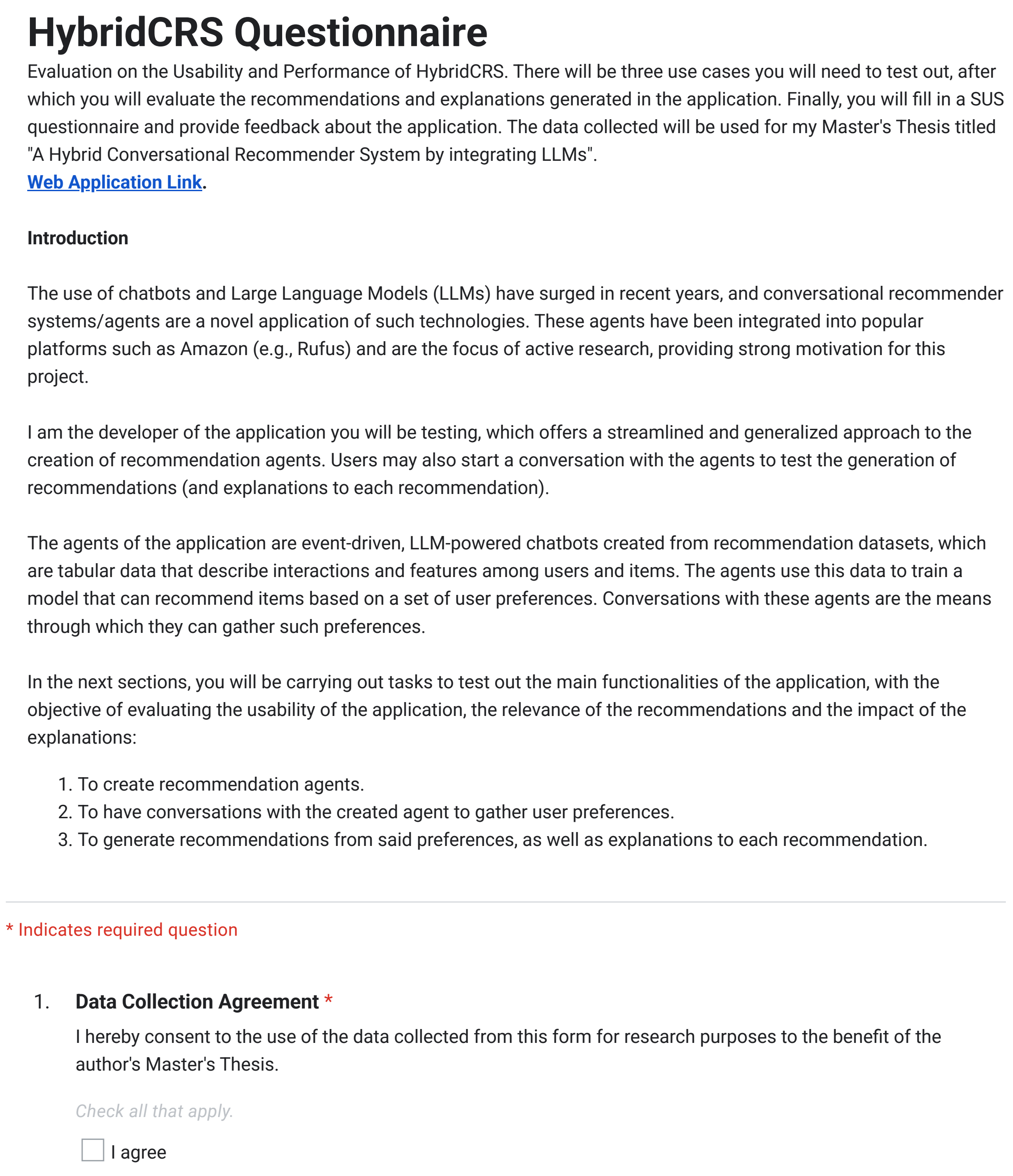
```
// Goal: Calculate the average rating and total number of ratings for the recommended item.
//
// Parameters:
// -$item_id: The ID of the recommended item (e.g., 'item_550').

// Find all RATED relationships pointing to the recommended item.
MATCH (:User)-[r:RATED]->(rec:Item {item_id: $item_id})

// Use aggregation to get the average and count.
RETURN
  avg(r.rating) AS avgRating,
  count(r) AS totalRatings
```

B

System Usability Questionnaire



HybridCRS Questionnaire

Evaluation on the Usability and Performance of HybridCRS. There will be three use cases you will need to test out, after which you will evaluate the recommendations and explanations generated in the application. Finally, you will fill in a SUS questionnaire and provide feedback about the application. The data collected will be used for my Master's Thesis titled "A Hybrid Conversational Recommender System by integrating LLMs".
Web Application Link.

Introduction

The use of chatbots and Large Language Models (LLMs) have surged in recent years, and conversational recommender systems/agents are a novel application of such technologies. These agents have been integrated into popular platforms such as Amazon (e.g., Rufus) and are the focus of active research, providing strong motivation for this project.

I am the developer of the application you will be testing, which offers a streamlined and generalized approach to the creation of recommendation agents. Users may also start a conversation with the agents to test the generation of recommendations (and explanations to each recommendation).

The agents of the application are event-driven, LLM-powered chatbots created from recommendation datasets, which are tabular data that describe interactions and features among users and items. The agents use this data to train a model that can recommend items based on a set of user preferences. Conversations with these agents are the means through which they can gather such preferences.

In the next sections, you will be carrying out tasks to test out the main functionalities of the application, with the objective of evaluating the usability of the application, the relevance of the recommendations and the impact of the explanations:

1. To create recommendation agents.
2. To have conversations with the created agent to gather user preferences.
3. To generate recommendations from said preferences, as well as explanations to each recommendation.

* Indicates required question

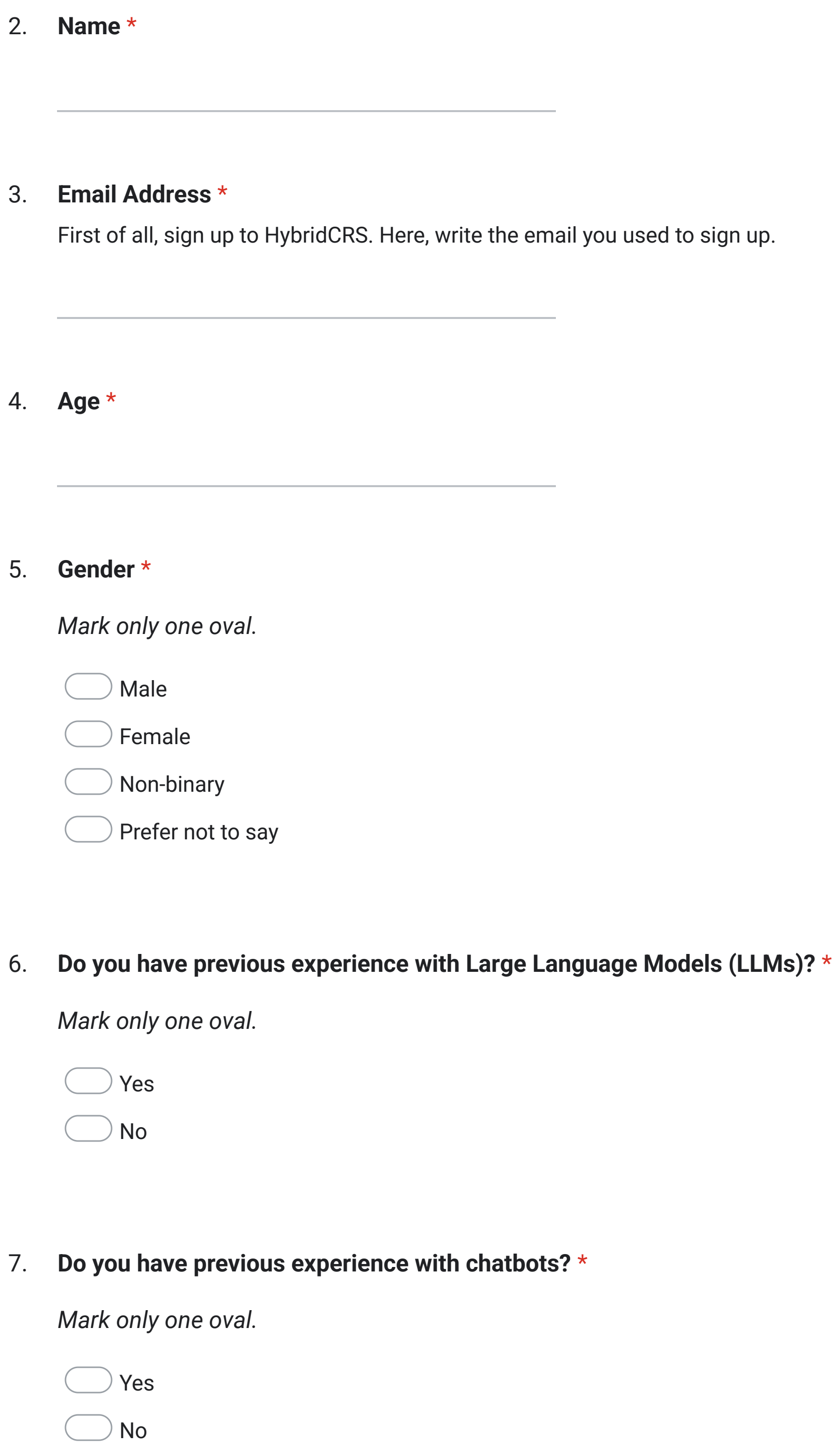
1. **Data Collection Agreement** *

   I hereby consent to the use of the data collected from this form for research purposes to the benefit of the author's Master's Thesis.

   Check all that apply.

   ☐ I agree

Figure B.1: Questionnaire: Page 1



2. **Name** *

3. **Email Address** *

   First of all, sign up to HybridCRS. Here, write the email you used to sign up.

4. **Age** *

5. **Gender** *

   Mark only one oval.

   ◯ Male
   ◯ Female
   ◯ Non-binary
   ◯ Prefer not to say

6. **Do you have previous experience with Large Language Models (LLMs)?** *

   Mark only one oval.

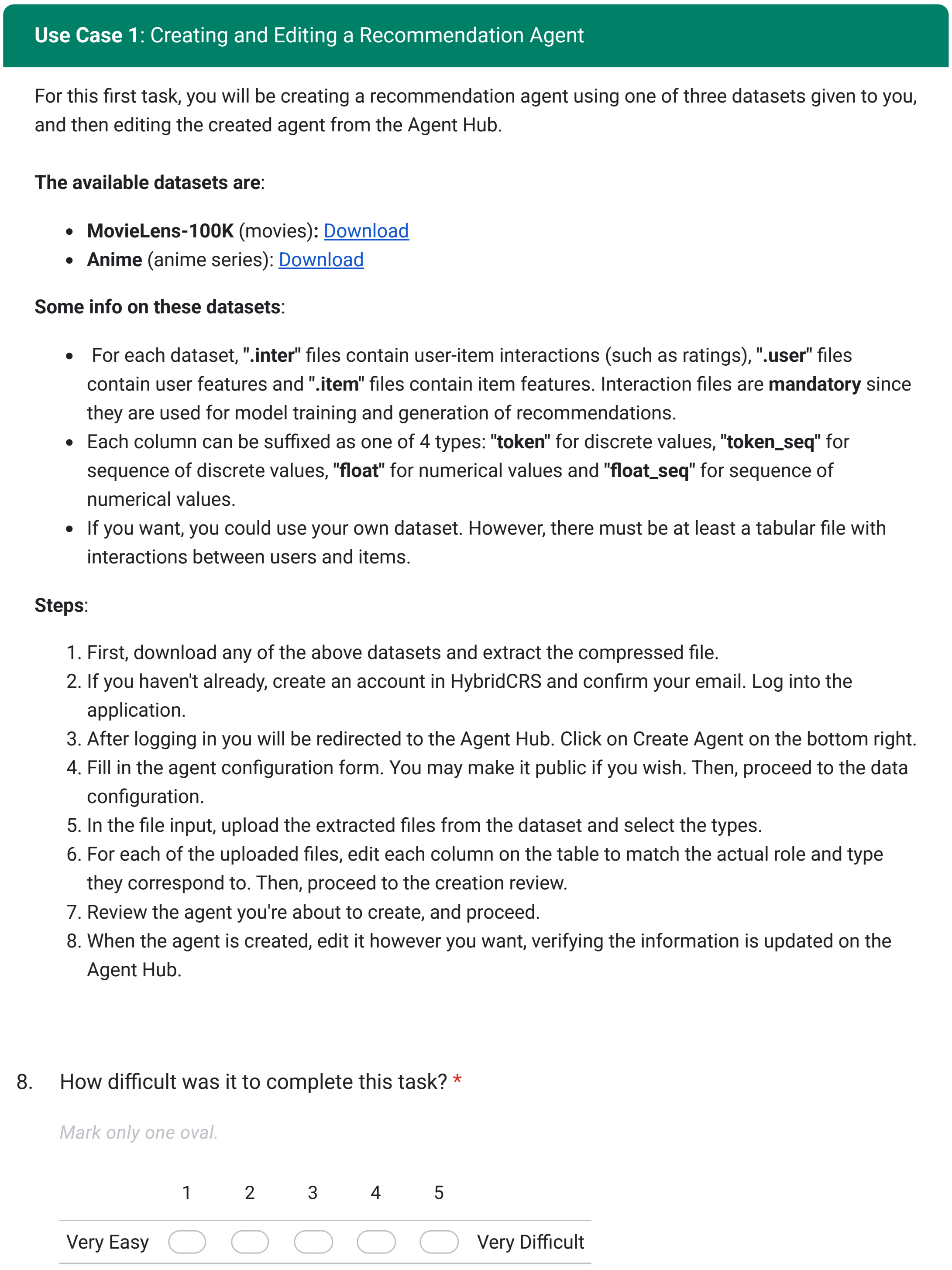
   ◯ Yes
   ◯ No

7. **Do you have previous experience with chatbots?** *

   Mark only one oval.

   ◯ Yes
   ◯ No

Figure B.2: Questionnaire: Page 2



Use Case 1: Creating and Editing a Recommendation Agent

For this first task, you will be creating a recommendation agent using one of three datasets given to you, and then editing the created agent from the Agent Hub.

The available datasets are:

- **MovieLens-100K** (movies)**:** Download
- **Anime** (anime series): Download

Some info on these datasets:

- For each dataset, "**.inter**" files contain user-item interactions (such as ratings), "**.user**" files contain user features and "**.item**" files contain item features. Interaction files are **mandatory** since they are used for model training and generation of recommendations.
- Each column can be suffixed as one of 4 types: "**token**" for discrete values, "**token_seq**" for sequence of discrete values, "**float**" for numerical values and "**float_seq**" for sequence of numerical values.
- If you want, you could use your own dataset. However, there must be at least a tabular file with interactions between users and items.

Steps:

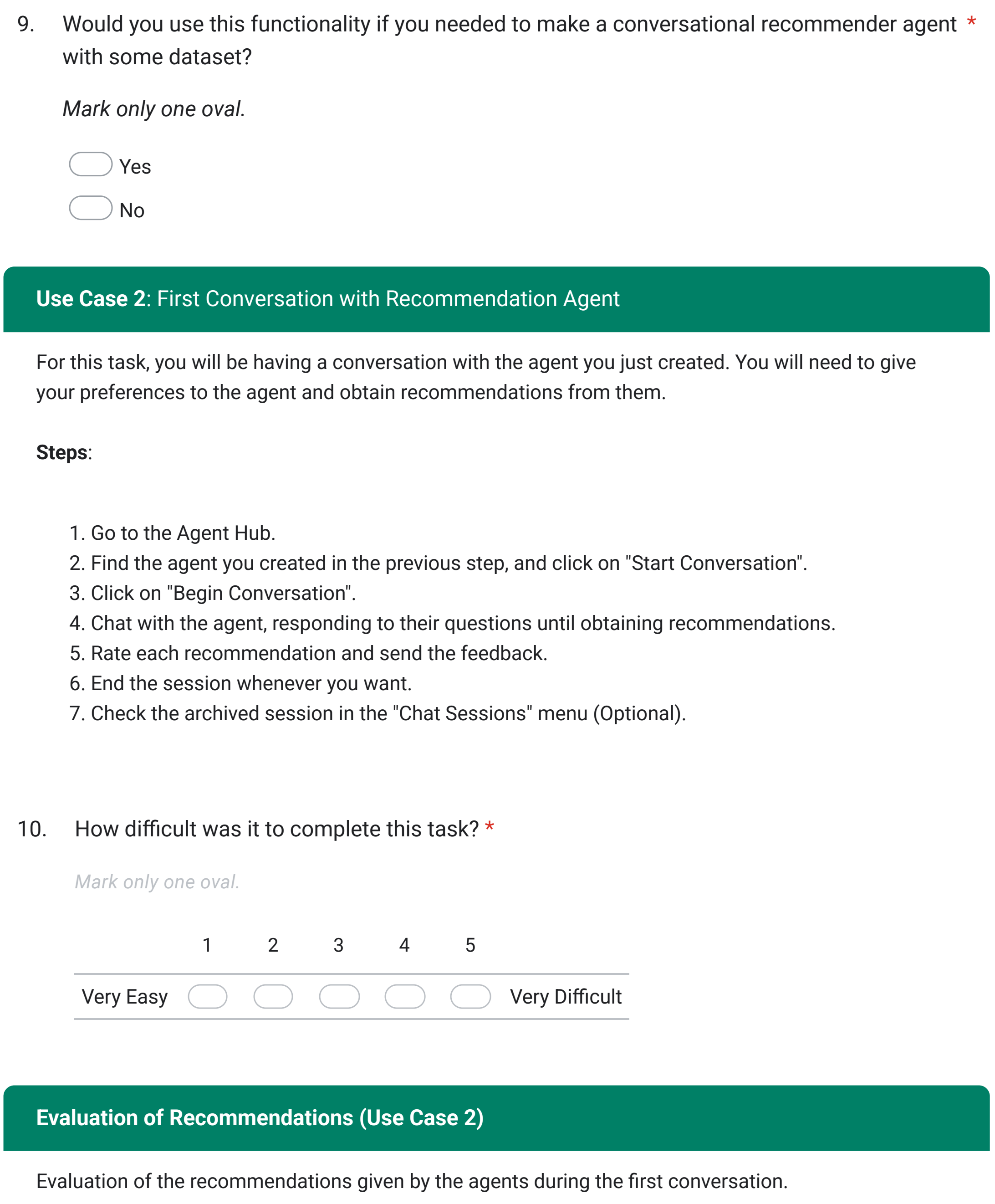
1. First, download any of the above datasets and extract the compressed file.
2. If you haven't already, create an account in HybridCRS and confirm your email. Log into the application.
3. After logging in you will be redirected to the Agent Hub. Click on Create Agent on the bottom right.
4. Fill in the agent configuration form. You may make it public if you wish. Then, proceed to the data configuration.
5. In the file input, upload the extracted files from the dataset and select the types.
6. For each of the uploaded files, edit each column on the table to match the actual role and type they correspond to. Then, proceed to the creation review.
7. Review the agent you're about to create, and proceed.
8. When the agent is created, edit it however you want, verifying the information is updated on the Agent Hub.

8. How difficult was it to complete this task? *

Mark only one oval.

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Difficult

Figure B.3: Questionnaire: Page 3



9. Would you use this functionality if you needed to make a conversational recommender agent * with some dataset?

Mark only one oval.

◯ Yes

◯ No

Use Case 2: First Conversation with Recommendation Agent

For this task, you will be having a conversation with the agent you just created. You will need to give your preferences to the agent and obtain recommendations from them.

Steps:

1. Go to the Agent Hub.
2. Find the agent you created in the previous step, and click on "Start Conversation".
3. Click on "Begin Conversation".
4. Chat with the agent, responding to their questions until obtaining recommendations.
5. Rate each recommendation and send the feedback.
6. End the session whenever you want.
7. Check the archived session in the "Chat Sessions" menu (Optional).

10. How difficult was it to complete this task? *

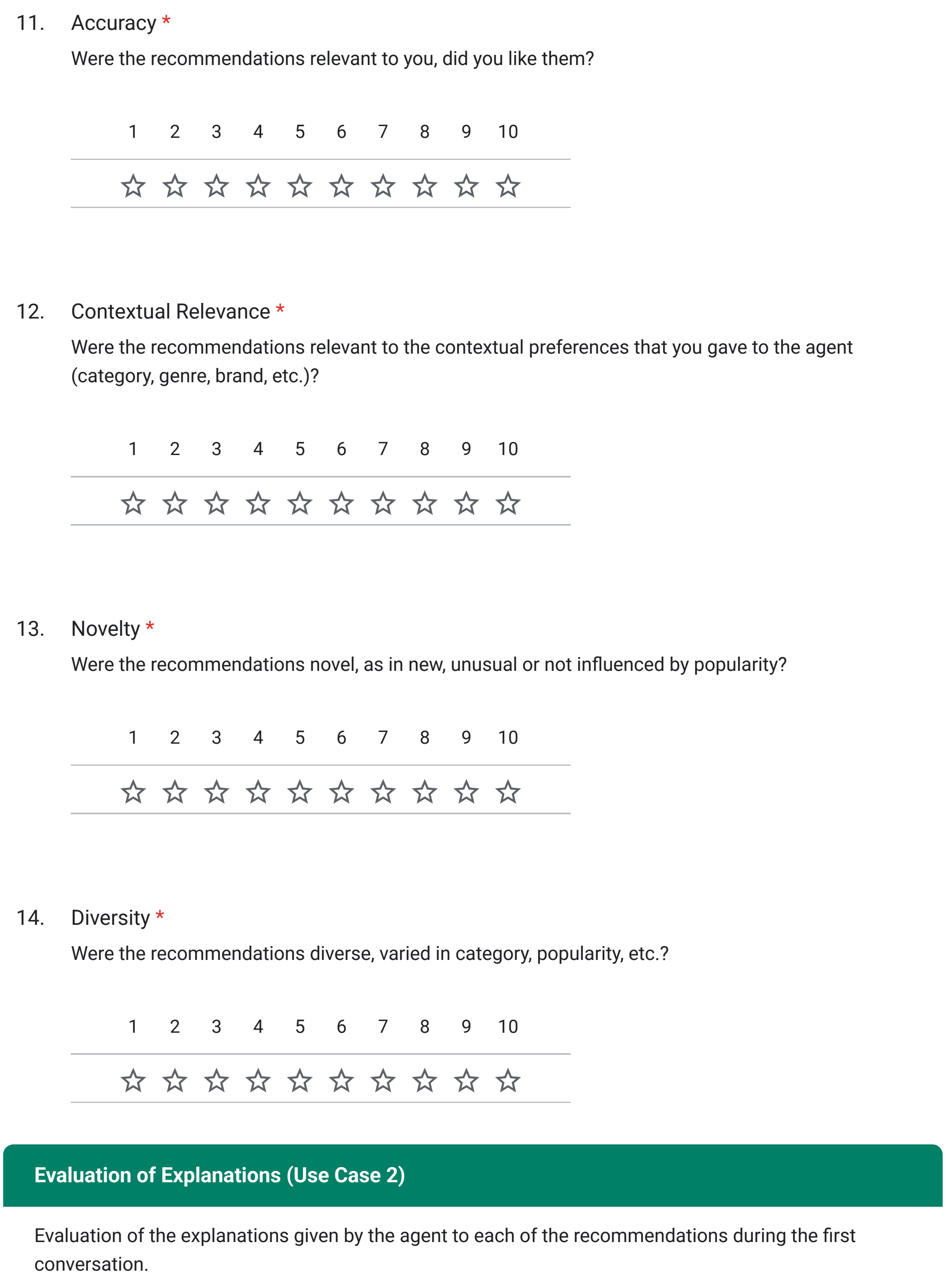
Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Difficult

Evaluation of Recommendations (Use Case 2)

Evaluation of the recommendations given by the agents during the first conversation.

Figure B.4: Questionnaire: Page 4



11. Accuracy *

Were the recommendations relevant to you, did you like them?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

12. Contextual Relevance *

Were the recommendations relevant to the contextual preferences that you gave to the agent (category, genre, brand, etc.)?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

13. Novelty *

Were the recommendations novel, as in new, unusual or not influenced by popularity?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

14. Diversity *

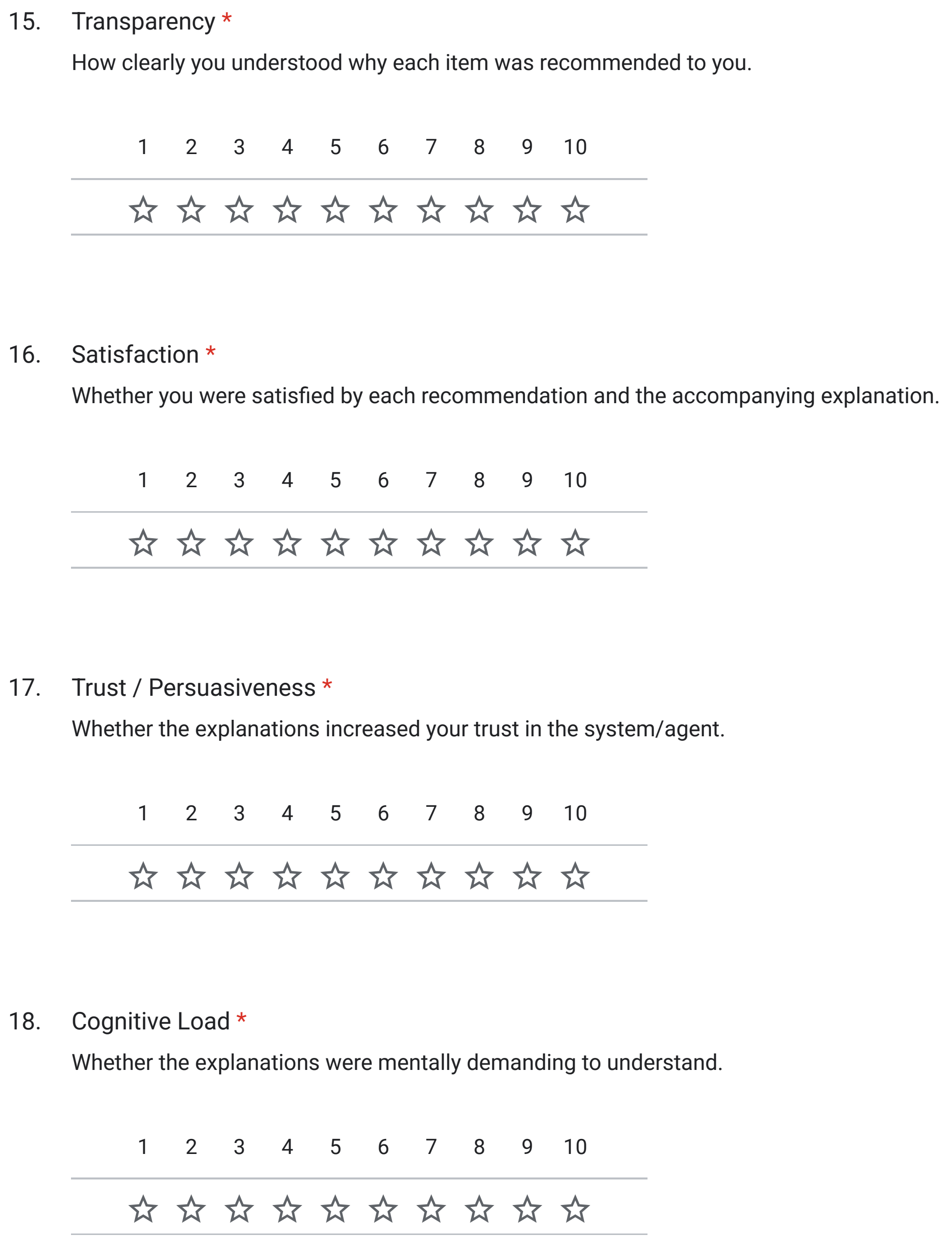
Were the recommendations diverse, varied in category, popularity, etc.?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

Evaluation of Explanations (Use Case 2)

Evaluation of the explanations given by the agent to each of the recommendations during the first conversation.

Figure B.5: Questionnaire: Page 5



15. Transparency *

How clearly you understood why each item was recommended to you.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

16. Satisfaction *

Whether you were satisfied by each recommendation and the accompanying explanation.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

17. Trust / Persuasiveness *

Whether the explanations increased your trust in the system/agent.

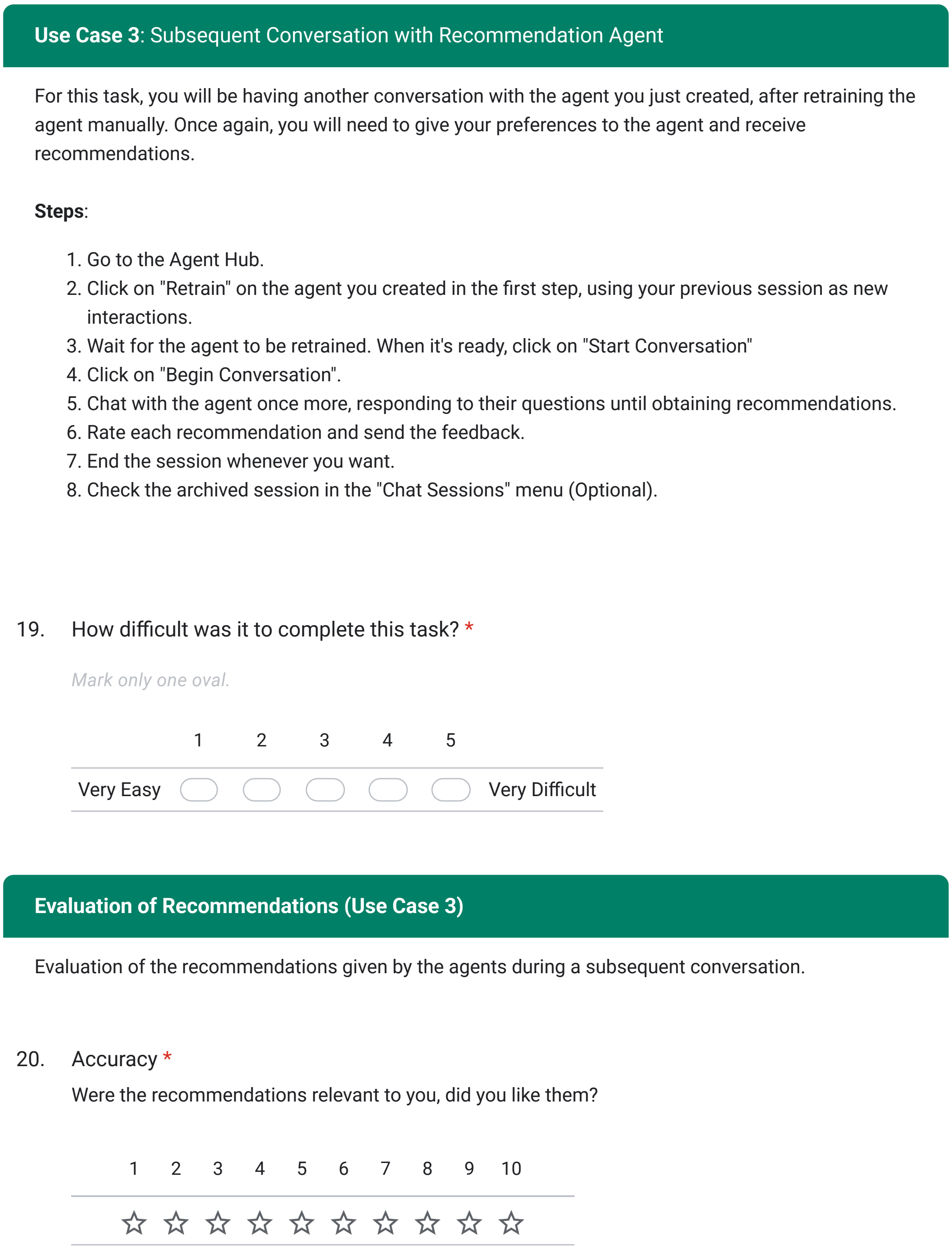
1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

18. Cognitive Load *

Whether the explanations were mentally demanding to understand.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

Figure B.6: Questionnaire: Page 6



Use Case 3: Subsequent Conversation with Recommendation Agent

For this task, you will be having another conversation with the agent you just created, after retraining the agent manually. Once again, you will need to give your preferences to the agent and receive recommendations.

Steps:

1. Go to the Agent Hub.
2. Click on "Retrain" on the agent you created in the first step, using your previous session as new interactions.
3. Wait for the agent to be retrained. When it's ready, click on "Start Conversation"
4. Click on "Begin Conversation".
5. Chat with the agent once more, responding to their questions until obtaining recommendations.
6. Rate each recommendation and send the feedback.
7. End the session whenever you want.
8. Check the archived session in the "Chat Sessions" menu (Optional).

19. How difficult was it to complete this task? *

Mark only one oval.

	1	2	3	4	5	
Very Easy	○	○	○	○	○	Very Difficult

Evaluation of Recommendations (Use Case 3)

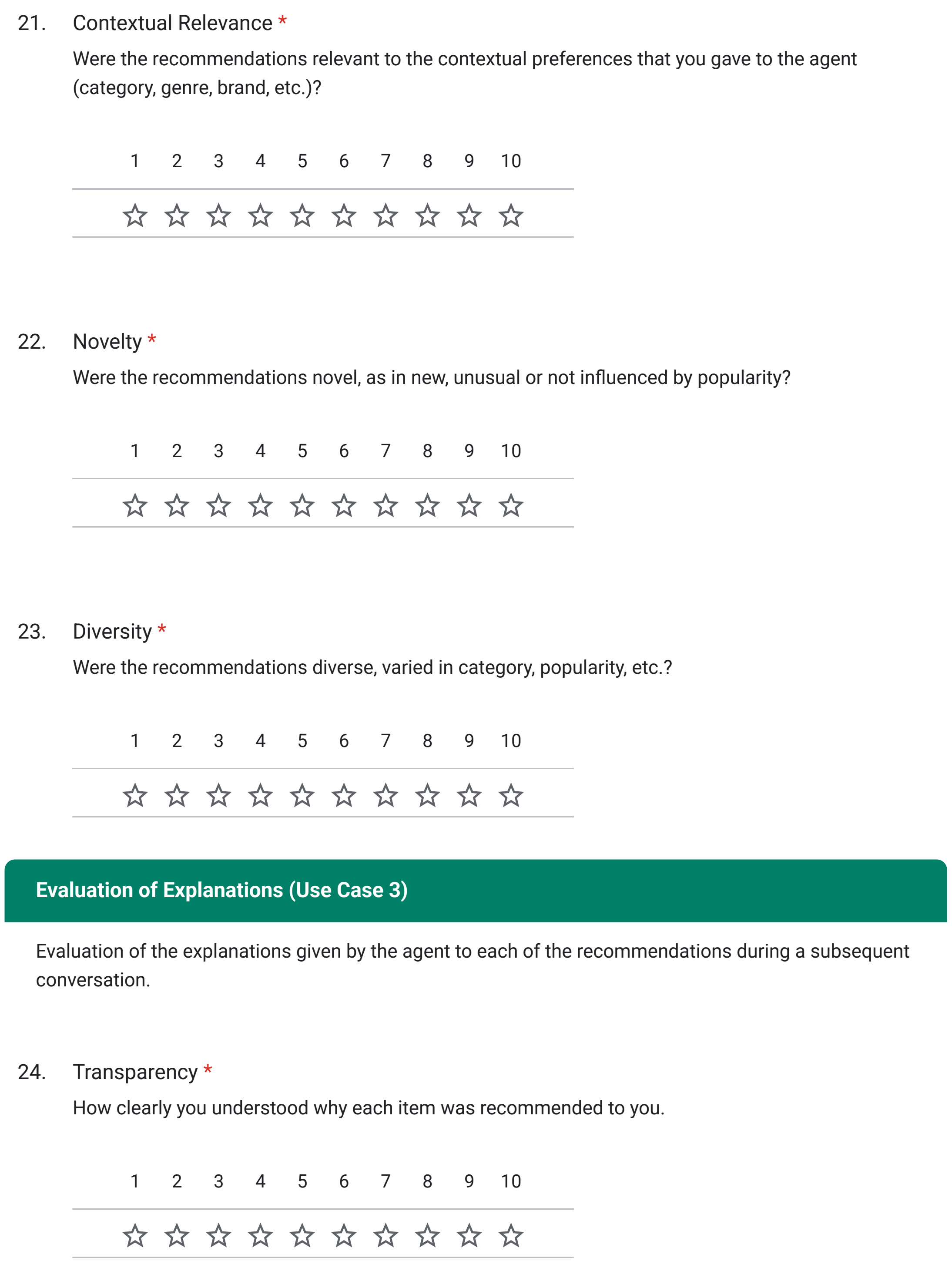
Evaluation of the recommendations given by the agents during a subsequent conversation.

20. Accuracy *

Were the recommendations relevant to you, did you like them?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

Figure B.7: Questionnaire: Page 7



21. Contextual Relevance *

Were the recommendations relevant to the contextual preferences that you gave to the agent (category, genre, brand, etc.)?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

22. Novelty *

Were the recommendations novel, as in new, unusual or not influenced by popularity?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

23. Diversity *

Were the recommendations diverse, varied in category, popularity, etc.?

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

Evaluation of Explanations (Use Case 3)

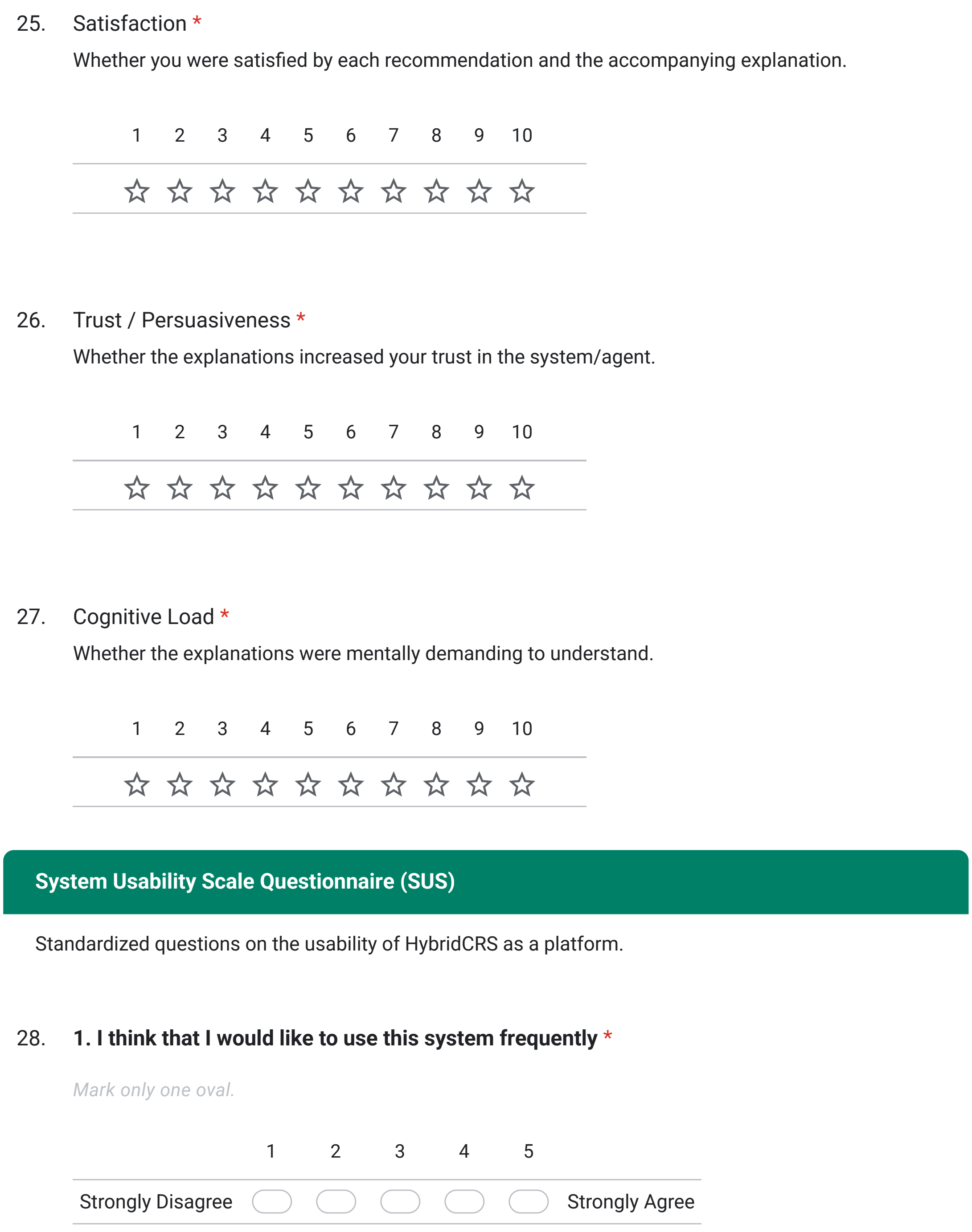
Evaluation of the explanations given by the agent to each of the recommendations during a subsequent conversation.

24. Transparency *

How clearly you understood why each item was recommended to you.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

Figure B.8: Questionnaire: Page 8



25. Satisfaction *

Whether you were satisfied by each recommendation and the accompanying explanation.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

26. Trust / Persuasiveness *

Whether the explanations increased your trust in the system/agent.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

27. Cognitive Load *

Whether the explanations were mentally demanding to understand.

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

System Usability Scale Questionnaire (SUS)

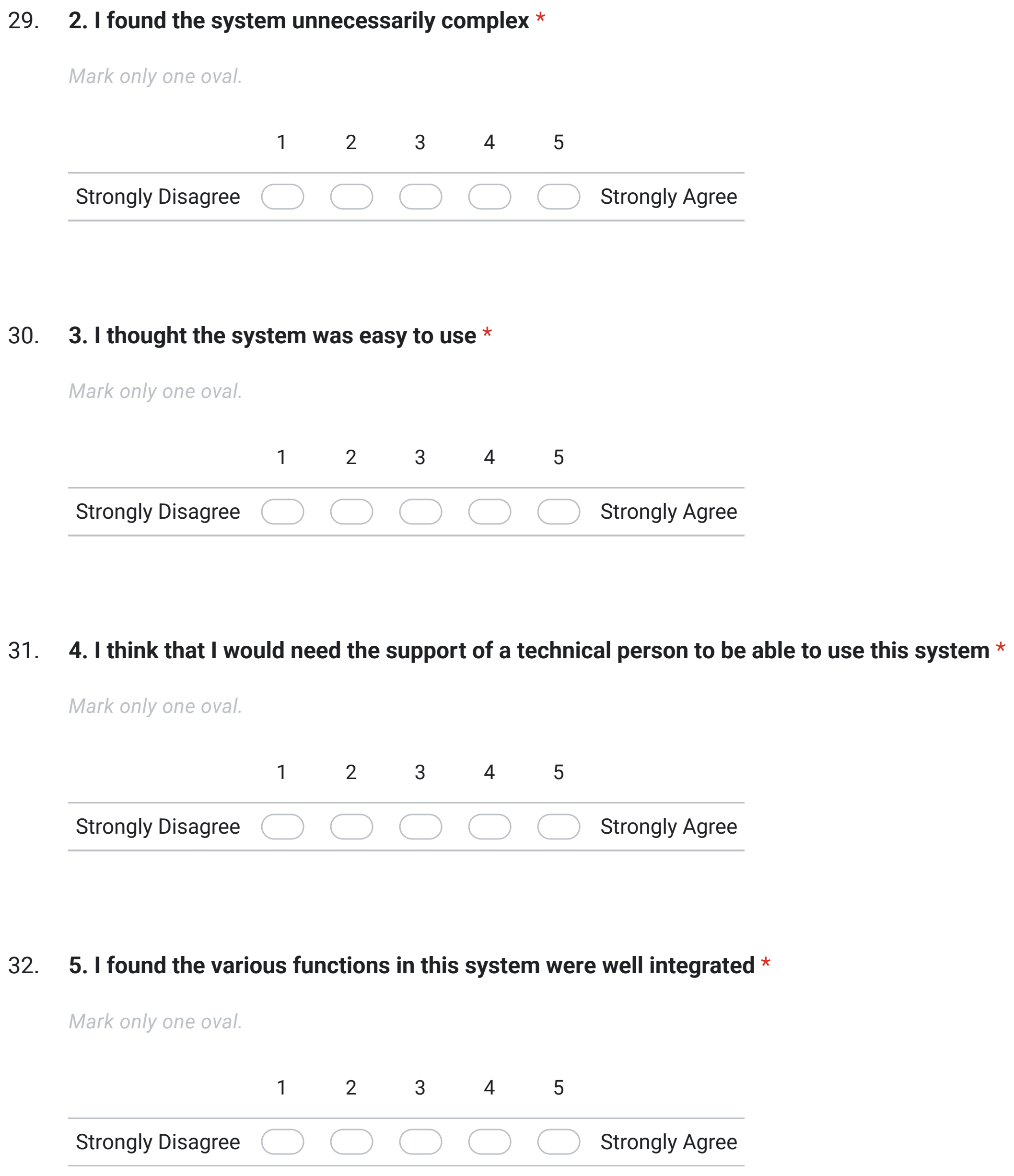
Standardized questions on the usability of HybridCRS as a platform.

28. **1. I think that I would like to use this system frequently** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

Figure B.9: Questionnaire: Page 9



29. **2. I found the system unnecessarily complex** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

30. **3. I thought the system was easy to use** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

31. **4. I think that I would need the support of a technical person to be able to use this system** *

Mark only one oval.

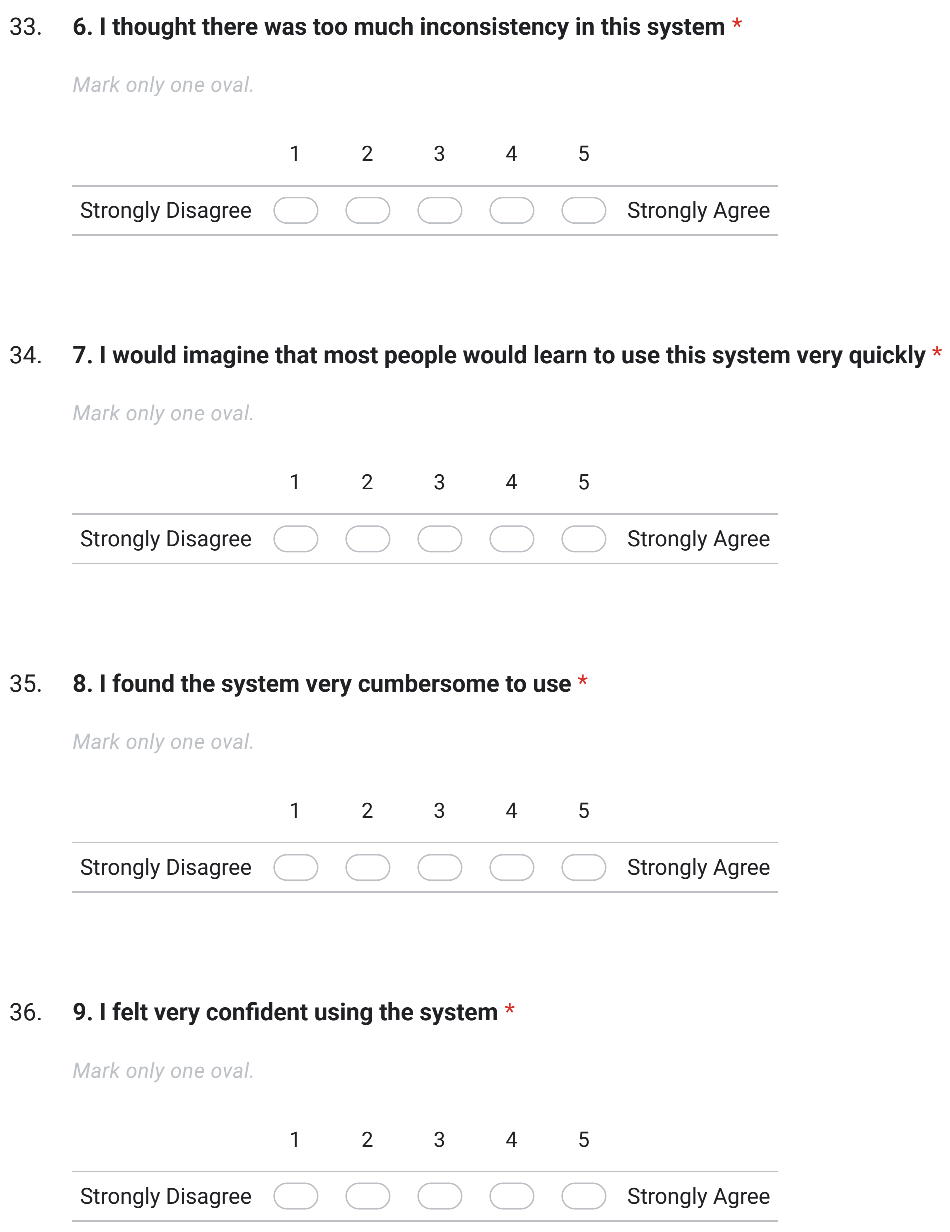
	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

32. **5. I found the various functions in this system were well integrated** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

Figure B.10: Questionnaire: Page 10



33. **6. I thought there was too much inconsistency in this system** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

34. **7. I would imagine that most people would learn to use this system very quickly** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

35. **8. I found the system very cumbersome to use** *

Mark only one oval.

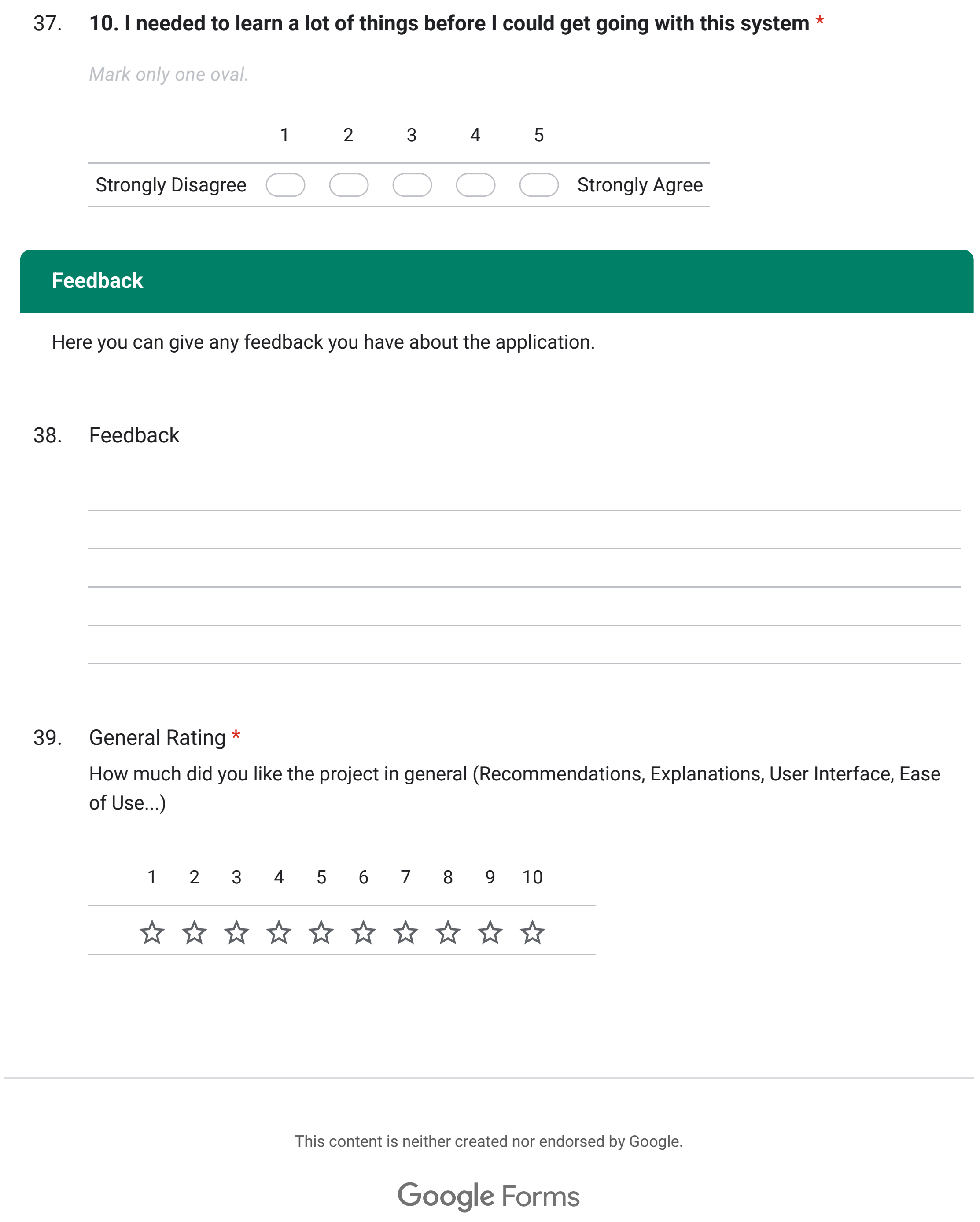
	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

36. **9. I felt very confident using the system** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

Figure B.11: Questionnaire: Page 11



37. **10. I needed to learn a lot of things before I could get going with this system** *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

Feedback

Here you can give any feedback you have about the application.

38. Feedback

39. General Rating *

How much did you like the project in general (Recommendations, Explanations, User Interface, Ease of Use...)

1	2	3	4	5	6	7	8	9	10
☆	☆	☆	☆	☆	☆	☆	☆	☆	☆

This content is neither created nor endorsed by Google.

Google Forms

Figure B.12: Questionnaire: Page 12

Universidad Autónoma de Madrid