A Framework for Explainable Multi-purpose Virtual Assistants: A Nutrition-Focused Case Study

Berk Buzcu[1] [0000−0003−1320−8006], Yvan Pannatier[1] [0000−0002−4890−3327], Reyhan Aydoğan[2,3] [0000−0002−5260−9999], Michael Ignaz Schumacher[1] [0000−0002−5123−5075], Jean-Paul Calbimonte[1] [0000−0002−0364−6945], and Davide Calvaresi[1] [0000−0001−9816−7439]

[1] University of Applied Sciences and Arts Western Switzerland HES-SO, Switzerland
[2] Computer Science, Özyeğin University, Turkey
[3] Interactive Intelligence, Delft University of Technology, Delft, Netherlands

Abstract. Existing agent-based chatbot frameworks need seamless mechanisms to include explainable dialogic engines within the contextual flow. To this end, this paper presents a set of novel modules within the EREBOTS agent-based framework for chatbot development, including dialog-based plug-and-play custom algorithms, agnostic back/front ends, and embedded interactive explainable engines that can manage human feedback at run time. The framework has been employed to implement an explainable agent-based interactive food recommender system. The latter has been tested with 44 participants, who followed a nutrition recommendation interaction series, generating explained recommendations and suggestions, which were, in general, well received. Additionally, the participants provided important insights to be included in future work.

Keywords: Chatbot Framework· Explainable AI· User Study

1 Introduction

Chatbots are conversational AI systems designed to interact with humans on specific topics and offer benefits across various use cases in a common dialogue form. In the fast-paced world of interactive digital systems, chatbots are rapidly evolving from simple assistants to sophisticated companions [1]. For instance, chatbots can direct users to solutions in customer support, reducing waiting times and boosting satisfaction [16]. In marketing, they can recommend products via personalized recommendation strategies [3]. In healthcare, they can even provide basic diagnoses and guidance for specific conditions [13, 9]. To hold a pleasant dialogue, chatbots must understand well their users' needs and intentions and respond accordingly. Solutions often include state-of-the-art AI technologies to understand the users' requests and responses. On the one hand, the chatbots can rely on flow-based solutions [34], Natural Language Processing (NLP) [19] and lately also on Large Language Models (LLMs) [27]. On the other hand, the off-the-shelves chatbots commonly employ finite state machines with reliable and transparent rule-based transitions between the states [35].

Given the inherently dialogic nature of chatbots—*i.e.*, they converse with the user toward a particular goal—we envision conversational agents that not only **answer** questions but can also **explain** their reasoning in plain language during an interaction. To do so, chatbot agents may potentially employ explainable AI (XAI) techniques to enhance the dialogue and consequently lead to a better persuasion strategy for the system (*e.g.*, a recommendation system or a virtual assistant). Furthermore, building and sustaining effective chatbots presents significant challenges that exceed the scope of a specific industry or topic. Researchers must consider the challenges of seamless integration with diverse devices, the user experience, interfaces, while ensuring user privacy and data security (especially in fields where privacy is critical, such as healthcare).

Thus, the complexity of building such chatbots powered by agents for user-tailored goals calls for a framework that can (R1) facilitate communication with agents and users flexibly, (R2) provide the necessary libraries for agent-based chatbot developers to foster focused development of modules, (R3) create user interfaces for the conversations between an agent and its users. Additionally, the dialogic nature of the chatbots with explanatory mechanisms opens the door to the potential risks of chatbot explanations being untrustworthy and/or misleading [10]. Existing frameworks (*e.g.*, EREBOTS [9]) address R1-R3. However, to the best of our knowledge, they do not include explainable engines with validation—which is crucial to complete the holistic chatbot framework.

Hence, in this study, the EREBOTS framework has undergone a significant overhaul to enhance its modularity and adaptability, introducing (i) context-agnostic customizable functionality that is extendable with plug-n-play custom modules—*e.g.*, finite state machines (FSMs); (ii) a no-code auto-generation of the user profiling FSM; (iii) a completely revised proxy agent seamlessly binding agnostic front and back end; (iv) a chatbot store collecting all the thematic chatbots hosted by the framework, and (v) a Flutter-based[4] multi-device and multi-purpose chat-like interface.

To test the above-mentioned novelties, we developed a nutrition virtual coach (NVC) focusing on health-oriented food recommendations. It relies on the adoption and extension of a food recommender from the literature [8]. To this end, we have (i) adapted the recommender flow towards a dialogic structure, (ii) developed a new heuristic to calculate recipe healthiness based on a profile–food estimation system, (iii) composed conversational explanation-based sentences, (iv) included the user feedback into the recommender engine, and (v) enabled the user to redefine their preferences/constraints. User experiments have been conducted to assess the system's acceptance and efficacy in terms of user experience. The results show that the majority of the participants appreciated the chatbot recommendations and explanations and were pleased with the interac-

[4] https://flutter.dev/

tions. Nevertheless, in the post-test survey, they gave valuable insights to be included in future work.

The remainder of the paper is organized as follows. Section 2 presents the state of the art. Section 3 presents the EREBOTS framework and its new modules and functionalities. Section 5 evaluates the use case and discusses the experiment results. Finally, Section 6 concludes the paper and presents future works.

2 Related Work

This section briefly elaborates on the literature regarding the chatbot ecosystem, as well as on the selected use case of food recommendation systems under the umbrella of nutrition virtual coaches.

2.1 Persuasive Food Recommendation Systems

Food recommendation is one of the most tackled problems in the recommender systems literature [7]. Given the difficulty of evaluating preferences and health considerations for a person, automated systems play a huge role in optimizing a goal-based diet. These systems may be rule-based, meaning that they rely on a set of predefined rules on user preferences, dietary restrictions, and the time of the day. These systems may make recommendations based on user similarities (Collaborative Filtering) [31] or recommending food based on the users' previous actions (Content-Based Filtering). For instance, a study expands upon the personalization of the recommended recipes where they decompose a user's preference into his preference of ingredients [17]. Additionally, the system may also utilise a knowledge base acquired from the user prior to the recommendation session where the user submits their constraints and preferences [15]. Food recommendation systems within the chatbots literature often leverage existing food recommendation research in a conversational structure with the user. Chatbot design lets a recommender system resemble a free-flowing conversation, similar to a dietitian, while learning user preferences and suggesting relevant options, thus boosting system adoption [26]. A similar conversational approach not only personalizes recommendations but also improves users' lives, as evidenced by high satisfaction among diabetic patients in a recent study [37]. A chatbot can also be used to log the conversation data for the user, making it simpler to converse their progress on a goal. Chatbots can be a valuable tool for individuals with specific dietary goals. For example, a fitness-oriented person can track their calorie intake and preferences over time, and the chatbot can use this information to recommend personalized recipes that align with their goals. This approach not only increases the accessibility of healthy eating but also makes it more engaging and persuasive [32]. Additionally, chatbots can continuously gather data and learn from user interactions, allowing them to adapt their recommendations over time. This ensures that users receive suggestions that are increasingly relevant to their evolving needs and preferences [12].

2.2 Chatbot based Recommendations in Healthcare

Chatbots within the umbrella of healthcare are not only applied under the guise of a dietitian, but may also be employed in various other fields of healthcare. Their use within the healthcare domain is perceived as important and adaptable, with concerns related to trust and skepticism among the system users [29]. Chatbot screening for Covid-19 reduced healthcare burden by guiding users, allowing them to stay home if appropriate [23]. The study involves collecting single-sentence symptom descriptions from the an online medical platform and performing exploratory data analysis to balance classes and identify corresponding features. They evaluate several deep learning models including LSTM with and without attention mechanisms and the BERT model. Ultimately, the chatbot interacts with patients via text to recommend appropriate medical specialties based on their presented symptoms. Another study similarly focuses on reducing the load on the healthcare system by offering a chatbot based general practitioner that can diagnose general sickness with remarkable accuracy [33].

2.3 Chatbot-based Recommendation Frameworks

In order to tackle the chatbot trust problem presented in the literature, some studies attempted to create uniform platforms to run the chatbots effectively [9, 39]. Some industrial applications exist too, such as Amazon Lex [4], Google Dialogflow [18], and ChatGPT [30]. However, most of the available frameworks only provide support for the dialogic aspects of the chatbot interaction, while they barely address user conversation management, profiling, and other agent-related functionality. Additionally, users lack the control of their private information that may be utilized in the system [22]. Recent works in the frameworks has included functionalities to enable agents to be more proactive in engaging in dialogue with the user while nudging them toward their goal [28]. The architecture employed in the study comprises two main components: a chatbot interface and a recommendation engine. The chatbot is deployed via a widely utilized mobile app and performs proactive functions such as sending adherence reminders (nudges) and continuous monitoring the users' health data, as well as reactive functions such as responding to patient queries and providing aggregated health data on demand. The recommendation engine follows both daily and long-term adherence profiles and generates personalized motivational messages to ensure patient compliance with the prescribed therapy. To do so, they use various NLP and NLG methods fine-tuned to the context of hypertensive patients. However, the framework is still context dependent with minimal concern for privacy.

3 A Novel XAI-enabled Chatbot Platform

The newly proposed version of EREBOTS provides a set of core modules for chatbot development including (i) context/front end agnostic gateway agent(s) binding the user interfaces and back ends, (ii) personal agents management APIs, (iii) internal messages routing, (iv) a no-code profiling finite state machine, (v) seamless connection with databases), and (vi) new customizable (plug-and-play) modules that enable explainable recommendations capabilities (*i.e.*, Recommender Engine, XAI Engine, and a Feedback Management module).

3.1 Architecture

Figure 1 provides a schematic representation of the framework. From a technical perspective, we classify it as EREBOTS v3.0. However, for the sake of simplicity, it is referred to hereafter as EREBOTS. The objective of such a framework is to enable the creation of personal virtual assistants (mapping one user to one virtual agent). EREBOTS is intended to be multi-purpose. Hence, it leaves to the developers the possibility of plugging their contextual (thematic) finite state machine (FSM). Such an FSM will describe and enact the dynamic of the intended storyline. The back-end is modular (leveraging Docker Compose[5]), and its main components are:

- **[C1] Database System(s)**: This component stores and manages relevant data for the system's operation. In particular, EREBOTS uses Pryv[6] to store the user data (a privacy-preserving and GDPR-compliant DB), and Mongo DB[7] to store functional and non-sensitive data.
- **[C2] Internal Agent Communication Server:** This component enables communication and message exchange among system entities using the Extensible Messaging and Presence Protocol (XMPP). In particular, EREBOTS uses Openfire[8], which enables near real-time messaging capabilities.
- **[C3] MAS Framework:** This component relies on SPADE[9] to generate and manage the multi-agent container.

This design fosters flexibility and allows for the integration and adaptation of alternative solutions based on specific sub-module requirements. EREBOTS is designed to be as technologically and application-agnostic as possible. Indeed, C1 and C2 are fully modular. For example, concerning the databases, Pryv could be replaced by Enzuzo[10] and MongoDB by MySQL[11]. The XMPP module (openfire) could be seamlessly replaced by Prosody[12].

C3 is the sole unique and irreplaceable component. Nevertheless, thanks to the core functionality, it leaves room for full customization as well as for the development/extension of custom modules. Such a core component encompasses a

[5] https://docs.docker.com/compose/
[6] https://www.pryv.com/
[7] https://www.mongodb.com/
[8] https://www.igniterealtime.org/projects/openfire/
[9] https://github.com/javipalanca/spade
[10] https://www.enzuzo.com/
[11] https://www.mysql.com/
[12] https://prosody.im/

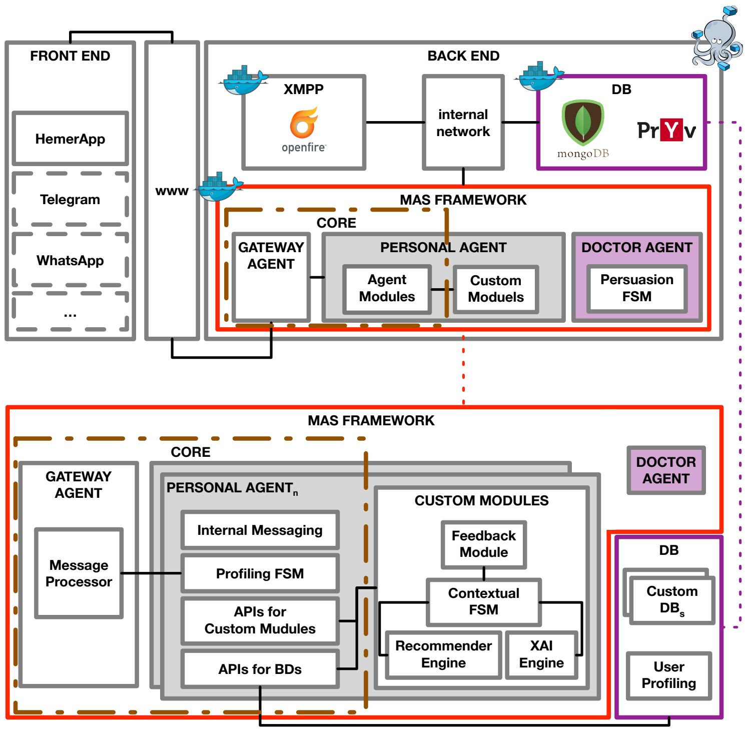


Fig. 1: High-level view of the architecture of EREBOTS v3.0

gateway agent and the foundational functionalities of the personal agent. Alongside the Agent Management System (AMS)—responsible for creating and keeping track of all the agents in the platform—and the Directory Facilitator (DF)—responsible for keeping a list of the provided services—the gateway agent (GA) plays a crucial role. By default, it is intended to be unique. Nevertheless, in case of necessity (*i.e.*, high messaging traffic or multi-campaign deployment), it is possible to instantiate multiple GAs. The Personal Agent (PA) selects the GA via the DF. The GA is responsible for (i) managing the WebSockets of the personal agents registered to them, (ii) parsing and adapting the incoming messages from the front-end format to the EREBOTS format, (iii) parsing and adapting the outgoing messages from the PA to the format of the front end. By doing so, the PA can be fully agnostic with respect to the front-end technologies, and the PA developer can focus on the needed internal strategies/logic.

Moreover, the PA has an underlying message routing mechanism that forwards the incoming messages to the given FSM (managing the PA's logic). The PA can allocate multiple custom FSMs (*i.e.*, thematic and persuasion-related) delegated to the developers of the given use cases. However, implemented among the PA's core functionalities, the user profiling FSM is the initial PA's behavior when first deployed. As different FSMs/modules require user information, the profiling FSM offers a zero-code customizable list of questions and answers (free text and multiple choice), configured through a single YAML file at development time. The contextual state machine (CSM) is the configurable chatbot flow manager, in the form of a FSM too.

Within our use case, we make use of a *Recommender Engine*, an *XAI Engine*, and a *Feedback Module*. The *Recommender Engine* operates as a recommendation selection module, identifying the most probable appropriate item to recommend to its user (provided that the PA is a recommendation agent in a use case). The *XAI Engine* is a post-hoc explanation generation engine that utilizes user information (*i.e.*, profile, negative/positive preferences, cultural constraints, and the user's feedback) and the recommendation information to try and explain *why* a recommendation has been made. Note that we have chosen to use post-hoc explanation systems due to their modular and decoupled uses. By doing so, the explainability module can be replaced by other post-hoc approaches in the literature rather seamlessly. Moreover, explainers and rule generators such as DEXIRE [14] and Psyky [24] could be integrated by default in the future. Additionally, *the Doctor Agent* is an agent that virtualizes a domain expert (in our use case, it could be a nutritionist). This domain expert can review the progress of the users in the campaign/platform, offer persuasive feedback, and, in the future, refine/validate the semantics and correctness of the XAI modules.

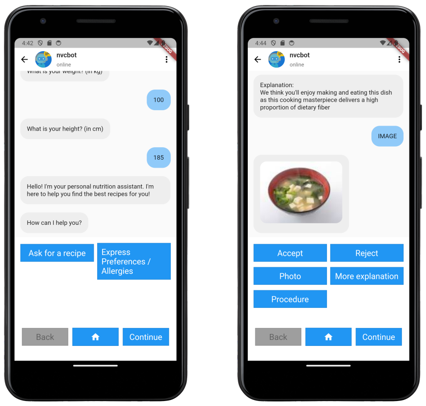
Regarding the Personal Agent and its communication pipeline, it operates as follows. Once the Personal Agent is enquired by the Core, following a request from the system front end to run the agent, the Personal Agent starts the profiling step. After profiling is complete, the agent engages the customized CSM, note that a chatbot within the EREBOTS framework **must** have this state machine. Within the CSM, the chatbot has a pre-defined flow of the dialogue where the respective Custom Modules are invoked accordingly. For instance, in our nutrition use case, we invoke the recommendation and the XAI engines within the recommendation state. Later, the agent gathers feedback utilizing the Feedback Module in a specific feedback state. Recall that these modules are decoupled, so the system developer can modify in which state these modules may be utilized in the CSM and change the modules entirely, according to their use case without having to re-design the architecture.

3.2 User Interface

Figure 2 illustrates a sample user interface for our front-end. The front-end has been implemented in Flutter to enable multi-platform/device support. Additionally, EREBOTS supports integration with well-known messaging platforms such as Telegram[13]. By connecting directly to our back-end, it removes the risk of sensitive information (*e.g.*, medical report) induced by the usage of a third-party platform. In addition, it comes with a basic set of features, such as free text, text-to-speech, and speech-to-text messaging, customizable keyboards al-

[13] https://telegram.org/

lowing the developer to cover a large panel of the use cases. The following section elaborates on our implementation of an explainable NVC as a proof of concept.



What is your weight? (in kg)

100

What is your height? (in cm)

185

Hello! I'm your personal nutrition assistant. I'm here to help you find the best recipes for you!

How can I help you?

Explanation:
We think you'll enjoy making and eating this dish as this cooking masterpiece delivers a high proportion of dietary fiber

IMAGE

Fig. 2: An example front-end from the use-case implementation

4 Nutrition Use Case

To test the EREBOTS framework and its newly added explainable and recommender modules, we have targeted the nutrition domain [10, 11]. In particular, we built a chat-based virtual coach recommending food recipes equipped with contextual explanations. We chose this scenario given that nutrition virtual coaches evaluate user-specific data, including dietary preferences, health conditions, fitness goals, and lifestyle choices, which are considered private information (C1). By using clear explanations for dietary advice generated via the helpers of the framework, users may gain a clearer understanding of the recommendations, increasing the likelihood of following them [8]. The modular structure of the EREBOTS framework spreads our use case into multiple submodules that can be tested independently (C2), and the nutrition coach is fully customizable per person with a dedicated agentic structure (C3). The building blocks of this case study are (i) a dataset of food recipes, (ii) a brief user profiling stage, (iii) a food recommendation selection algorithm implementing linear utility estimation and corresponding aggregation functions, and (iv) an explanation generation engine supporting the recommendation. Below, we present the procedure to build a food recipe database (Section 4.1), implement the Profiling FSM (Section 4.2), the Recommendation & XAI engines (Section 4.3), and finally the Feedback Module (Section 4.4).

4.1 Food Dataset Generation

ChatGPT4[14] has been used to generate sets of recipes according to several distributions of calories, ingredients, nutritional values, cost, allergens classification, type of cuisine, etc. In turn, the resulting set of recipes (a total of 7000) has been systematically cleaned and properly structured. The generated food recipes are composed of a sub-set of the fields in the nutrition ontology generated in the context of the European Project EXPECTATION [11], and they include dishes from all over the world (*i.e.*, West/East-Europe, North-South America, Africa, and Asia). The features of the food dataset are briefly presented in Table 1. Ultimately, this data is stored in the Mongo database as the main recipe database which is further personalized per user as explained in the following sections.

Table 1: Dataset features description

Feature name	**Description**	**Data type**
recipe_id	The unique ID of the recipes	String
title	Recipe name or title.	String
raw_text	Raw answer from LLM.	String
cultural_restriction	Cultural (*e.g.*, vegan, vegetarian) or religious (*e.g.*, halal, kosher) categories.	String
calories	Recipe's calorie value in kilocalories (kcal).	Float
allergens	Describe if the recipe contains one of the ten most common food allergens.	Categorical
price	Estimation for the recipe's cost as a categorical value in $\{1, 2, 3\}$, where 1: lowest, and 3: highest.	Categorical
taste	Category representing one of the five basic taste profiles in $\{sweet, sour, salty, bitter, umami\}$).	Categorical
Cuisine	String variable representing the recipe's cuisine	String
Ingredients	Describe the recipe's ingredients and quantities.	String
Preparation	Describes the recipe's preparation steps.	String
Carbohydrates	Carbohydrates amount in grams.	Float
Proteins	Protein amount in grams.	Float
Fats	Fat amount in grams.	Float
Fibers	Fiber amount in grams.	Float

4.2 User Profiling Stage

To acquire the user information needed for the process (*e.g.*, name, age, gender, weight, height, sports level, explanation preference), user profiling is executed as the first mandatory procedure right after registration. This process occurs prior to the Recommendation Stage as per the EREBOTS flow. The data acquired from the user profiling state is used by the heuristic calculating the *health score*

[14] https://cdn.openai.com/papers/gpt-4.pdf

property (HS, see Section 4.3). HS is an adimensional numerical representation of how healthy a recipe is considered for a system user's profile. Note that the profiling state machine (part of the EREBOTS core) is extended to calculate the HS property following the no-code YAML user profiling section. Once the registration is completed, the user reaches the home state. At that point they can decide to ask for a recommendation or to express their preferences (liked and disliked ingredients, constraints, etc.). Such tasks can be executed at any time and in any order.

Expressing the preferences, the user allows the system to acquire those pieces of information that are crucial to compute the *preference score* (PS, see Section 4.3). PS is an adimensional numerical representation of how much the user might like a recipe given if proposed. Having such a comparative metric allows for a quick comparison of the food recipes w.r.t. the user preferences. For instance, if *recipe A* is scored "1" it is assumed to be preferred by far over *recipe B* that is scored "0.5". The final outcome (i.e., the recommended recipe and the corresponding explanation) and the input leading to the outcome (i.e., users' BMI) are intended to be private information. Thus, they are stored in the Pryv database, where the data can only be accessed by agents that are explicitly authorized by the user.

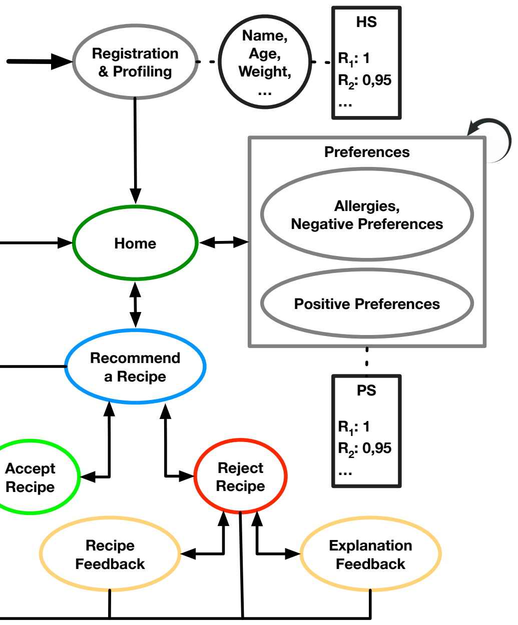


Fig. 3: The NVC states FSM diagram

4.3 Recommendation & XAI Engines

For the generation of food recommendations and explanations, we derive the protocol and the agent architecture from the literature [8]. The main difference is the decoupling of recommendations/explanations and their feedback. In our setting, the user can set to receive explanations concerning a given recommendation with it or on demand. Moreover, the user can ask for additional explanations concerning the recommended food recipe. This enhances the user's understanding and helps them in making an informed decision (*e.g.*, on whether either-or recommendation/explanation is clear, convincing, incorrect, unclear, etc.). Such new dynamics shift the approach borrowed from literature towards an iterable and bi-directional pipeline (see Figure 3).

Recommendation Mechanism Personalizing dietary recommendations is crucial for promoting healthy eating habits. Traditional methods often rely on *generic* guidelines and *subjective* preferences yet neglect individual needs—definitely of higher importance. This approach is composed of a heuristic method to estimate the healthiness of the given recipes based on the distance between a user's dietary profile and the recipe's nutritional composition (Health score, H_s) and a Jaccard similarity-based [6] preference modeling approach (Preference score, P_s) using an additive utility function with equal weights.

Health Score: A user profile can be represented using various methods, such as recommended daily nutrient intake. Recipes, on the other hand, can be represented as vectors containing their nutritional content. For instance, let us assume recipe R and a user profile P has the following vectors of nutrients:

$$R < protein, fat, carbohydrates, fiber, calories >$$

$$P < proteins, fat, carbohydrates, fiber, calories >$$

Thus, we can model the final healthiness estimation as a minimum distance problem, $||\mathbf{P} - \mathbf{R}||$ (Euclidean distance). Then we apply linear scaling to clamp the distances within the range of $(0, 1]$ where the closest distance is represented by 1 and the rest of the values are linearly distributed within the same range.

Preference Score: A user preference can be related to the ingredients of a food recipe [25, 36]. The system is initiated during the user profiling step with the information on whether they like certain ingredients. For instance, if a user specifies to dislike *onions*, then the system filters out all the recipes involving *onions*. Conversely, if they state they like *tomatoes*, then we apply Jaccard Similarity [6] in the following manner: Assume that a user has specified preferences for certain ingredients (*e.g.*, i_1, i_2, i_3), and considering a recipe denoted as $R = i_1, i_2, i_5, i_6$ (note that i_5 and i_6 are ingredients with no data), each liked ingredient is assigned a value of 1, and 0 otherwise. The average of these values results in a score of 0.5 for recipe R. Subsequently, all recipe scores are normalized to a range of (0, 1], establishing a relative importance for each recipe. Ultimately, the recipe's score is calculated as a weighted sum of the health score and the preference score as shown in Equation 1. The *recommendation* with the maximum of the *recipeScore* is elected as the next recipe to recommend as described in Equation 2.

$$recipeScore = w_p * PreferenceScore + w_h * HealthScore \tag{1}$$

$$recommendation = argmax_{recipe} recipeScore \tag{2}$$

After the user receives the recommendation at the Recommended Recipe State, the user can take the following actions, as also shown in Figure 3:(i) Accept the recipe and return to home state, (ii) Deny the recipe and proceed to feedback state, (See Section 4.4), (iii) Ask for the recipe's picture, (iv) Request more explanations, and (v) Get the cooking procedures of the recipe.

Explanation Generation This study adopts a technique to generate post-hoc explanations borrowed from the literature [8] to enhance the transparency and user-friendliness of the nutrition virtual coach, with the aim of persuading users to follow healthier eating habits. We chose post-hoc explanations due to their flexibility (applicable to any decision-making model) and modularity whilst making the framework as modular as possible. The main intuition of such an approach is to reduce the complex or otherwise unexplainable decision function to an inherently explainable method. Thus, we implement Decision Trees, which are inherently explainable models to determine the important features of our recommendations [2]. The decision tree is constructed from the outcomes of the recommendation selection algorithm (Section 4.3). The recipes are labeled according to whether they should be recommended (as "1"), not (as "-1"), or neutral (as "0"), then are used to form the decision tree with the features of nutritional information, and derived features of preference score and health scores. The decision tree is regenerated after the live input is processed by the recommendation engine. Ultimately, the chosen three features from the decision tree are used in a grammar-based structure. Figure 4 shows an example of an explanation generation grammar. This grammar structure picks the sections of the sentences according to the features selected by the explanation tree.

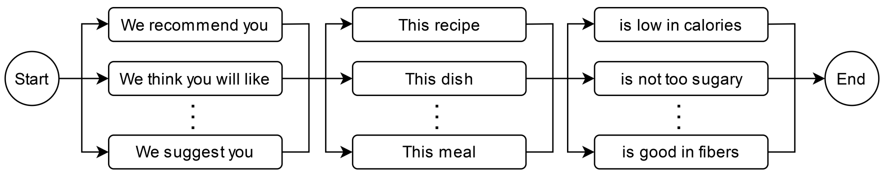


Fig. 4: Example of an explanation grammar.

4.4 Feedback Module

The feedback module enables the user to express their opinion on an agent's recommendation, hence enabling the user side of the dialogue. This study opts to utilize pre-defined options for feedback to reduce the computation complexity of understanding the human input, given the chance that the agent may misunderstand the feedback or branch out of the system's scope. The feedback options branch out at the base level, the user may give feedback to the recipe or the explanation. Table 2 shows the options presented to users when they wish to give feedback to a recommendation. Note that they can also specify the variable *ingredients* during their session.

Table 2: Feedback options for both recipe and explanations

Recipe Feedback Options	**Explanation Feedback Options**
I don't like the ingredient(s).	The explanation is not convincing.
I'm allergic to ingredient(s).	The explanation doesn't fit my case.
I ate an ingredient(s) recently.	The explanation is not clear enough.
I don't have the ingredient(s).	The explanation is incomplete.
I don't have time to cook this recipe.	
The recipe is unsuitable for the time/day.	

5 Experimental Results & Discussion

To test the efficacy of the developed NVC module and, consequently, the EREBOTS framework, we have conducted experiments with volunteers belonging to several categories (students, professors, researchers, and non-scientific profiles) via the EREBOTS mobile application. In total, we recruited 44 participants. They were first introduced to the study's goals, principles, privacy, and consent values (briefly presenting Pryv[15]) and then given a brief demonstration of the system using a test bench. Hands-on the device, the participants have created their own Pryv profiles and conducted the experiment for a variable duration (as long as they were willing to *play around)*. Finally, after interacting with the system, they completed a post-experiment questionnaire to capture their experiences and engaged in an informal interview focusing on their inputs beyond the structured questionnaire.

The effectiveness of self-explanatory systems is typically evaluated using two categories of metrics: objective and subjective [21, 20, 38]. Objective metrics focus on participant actions within the experiment. Examples include success rate (*i.e.*, percentage of sessions ending with agreement, or contextually accepting the recipes), number of interaction rounds per session, and analysis of potential misunderstandings or feedback provided during interactions. Subjective metrics, on

[15] https://www.pryv.com/

the other hand, capture participant perceptions through post-experiment questionnaires and focus groups/interviews. Accordingly, Section 5.1 overviews the objective metrics, and Section 5.2 summarizes the subjective metrics.

5.1 Findings with the Objective Metrics

Once registration and profiling have been completed, the system is in the *home state*. For this study, *asking for a recipe* and *expressing preferences* are the only two options in the *home state* menu. Surprisingly, at most 33 out of 44 participants opted to express their preferences before asking for a recommendation. This behavior has later been explained by the participants' curiosity in exploring the possibilities and being surprised rather than just narrowing their possibilities at the very beginning. Indeed, some of the participants had restrictions that they should have observed, but their curiosity won the first round. Some users have indeed asked for multiple recommendations without constraints/preferences (just related to their basic profile). Only afterward, they have tested the recommendation compliance with their predilections and, ultimately, with their feedback in case of rejection. All of the participants have accepted a recipe in their session, meaning there has been no session where the user gave up. Overall, the participants acquired 130 recommendations and they accepted 76 of them. For the recipes the users rejected, the users gave the following feedback with the corresponding amounts (note that the users go back and forth the Reject and Feedback states while providing multiple kinds of feedback):(a) I don't like ingredients: 10, (b) I don't have the ingredients: 9, (c) Recently eaten an ingredient(s): 7, (d) Unsuitable time for the meal: 5, (e) I don't have time to cook this recipe: 5, (f) The explanation is not convincing: 4, (g) The explanation is not clear enough: 1, and (h) No feedback: 12.

It is worth recalling that the participants could ask for (i) additional explanations, (ii) more information about the cooking procedures of recipes, and (iii) a picture of the recipe. The participants asked for additional explanations 36 times, for the cooking procedures 37 times, and for the recipe picture 56 times. An interesting finding is that two participants opted to not receive explanations, however, they asked for explanations prior to acceptance of their chosen recipe. Finally, no food recommendation has been delivered without explanations (even having given/chosen such an option). Moreover, several participants asked for *additional explanations*, *more information on cooking procedures*, and for **the picture** of the final meal. However, the additional pieces of information requested have not always led to the acceptance of the recommendation. Table 3 collects the detailed outcomes.

Figure 5 illustrates the average amount of time users spent on each state.

5.2 Questionnaire Findings

Figure 6 illustrates the results in box-plot format, and reports the related questions. The questionnaire is comprised of 5 point Likert scale questions where we interpret the negatively worded questions better if they are scored lower, and

Table 3: Information requested by users and the amount of times it led to acceptance or rejection.

User requests	Acceptance	Rejection
Additional explanations	26	11
Cooking procedure	27	10
Picture of recipe	41	15

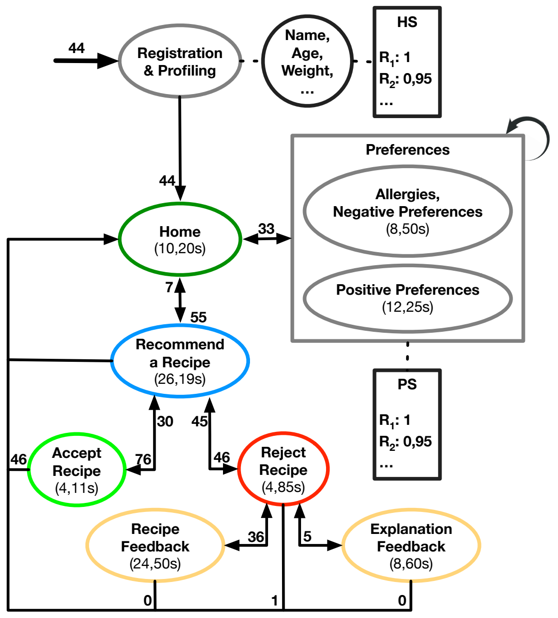
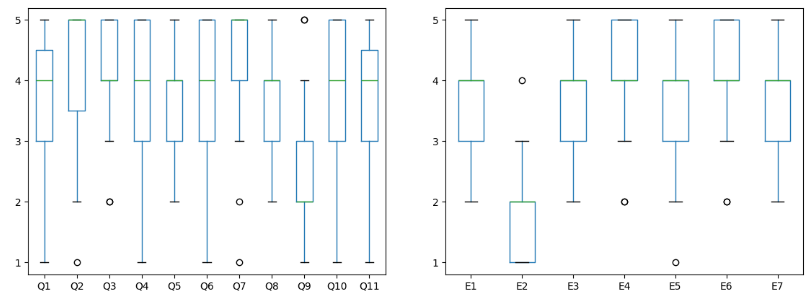


Fig. 5: NVC state diagram with average time spent on each state

the vice-versa. Additionally, to the questions in the figure, the questionnaire also included questions with the following average ratings:

- Q12: "The user interface is user-friendly and easy to use." - 4.20 ± 0.95
- Q13: "It was overwhelming to enter my food constraints such as liked and disliked ingredients and cuisine choices." - 1.71 ± 0.88
- Q14: "I feel my privacy was violated during the interaction. - 1.38 ± 0.78

As a result, we observe that participants generally enjoyed the interacting with the system (See Q11 is rated on average 3.67). They believed that the system was taking into account their feedback (Q7, rated 4.35). The participants were generally pleased with the recommendations (Q1: 3.75, Q2: 4.13 and Q3: 4.18 on average). The system was also discovered to be not too exhausting (Q9: $2.53 < 3$, Q10: 3.84, Q12: 4.2, Q13: $1.71 < 3$). Regarding the explanations made by the system, the participants found them satisfactory (E3: 3.61, E5, 3.61 on average). Although the questionnaire results support the effectiveness of the ex-



Id	Question
Q1	I am satisfied with the recommendations made by the system.
Q2	The system tried to make recommendations according to my preferences.
Q3	I believe the recommended recipes were tasty.
Q4	The images on the recommendations influenced my choices.
Q5	How do you rate the healthiness of the recommended recipes?
Q6	The interaction interface enabled me to effectively express my feedback for the recommendations.
Q7	The recommender system was considering my feedback.
Q8	The recommender system gave me enough information to decide.
Q9	It was time-consuming to find the best recipe in the system.
Q10	The interaction with the system was smooth and effortless.
Q11	I would like to use such a system in my daily life.
E1	The explanations for recommendations have helped me choose the most convenient recipe.
E2	The explanations for recommendations were too detailed.
E3	The explanations displayed during the interaction were satisfactory.
E4	The explanations for recommendations were clear and easy to understand.
E5	The explanations were sufficient to make an informed decision for healthiness.
E6	The explanations were realistic in terms of healthiness of given recipes.
E7	The explanations let me know how convenient the recipe is.

Fig. 6: The participants' responses to the questionnaire

plainability and the interaction, there is still room for improvement as discussed in the following section.

5.3 Informal Interviews & Discussion

While the experiment participants appreciated the agent's ability to recommend dishes according to their preferences accompanied by transparent explanations (See the ratings in Section 5.2), the informal interviews held with the experiment attendees after the experiment was complete reveal a desire for even further user control and options (yet, possibly leading to overwhelm the user). Participants expressed curiosity and were pleasantly surprised by the variety of recipes the

NVC generated. The system's interactivity and response times further enhanced the user experience of a Food Recommendation System. Moreover, they asked for additional features or customization features. Some examples include budget filters (some of the recipes were out of students' budget), batch cooking options (some students cook over the weekend for a batch of three or four meals, and individuals responsible for cooking for their families have required to have a portion scaling included in the system), cooking style options (some participants would not accept frying or boiling, and other have highlighted an unfortunate lack of tools to prepare the suggested recipes), and timing-related filters (most of the recipes are delicious, cannot be an option due to lack of time, at least during the week days).

Furthermore, despite the broad recipe selection for specific dietary needs and cuisines, some participants found that their specific categories were absent or underrepresented in the recommendations. This may be because the healthiness function is a discrete heuristic that shined more in some recipes than others or because they expected a wider coverage of recipes within the database (including even more local/typical food). Ultimately, the participants' trust appears to be contingent upon functionalities that foster engagement, such as the presentation of the recipe options and the ability to delve deeper into recipe explanations while providing feedback. Some issues might arise from the need for more options during the dialogue. In the case of explanations, the users found that some explanations needed to be more convincing or clear. Some participants have not fully appreciated the "over-tuning" of the explanations on the healthiness aspects of the recipes rather than the convenience. Conversely, some users have stated that they liked the explanations and their elaboration on why/how a recipe is healthy. This minor contradiction between the system participants is due to their different preferences and expectations w.r.t. the system. The key takeaway is that the presentation of the explanations and their target features should also be personalized rather than focusing on a single domain and a fixed amount of detail. Meanwhile, on the user input side, some participants thought that the options were limited. For instance, it was not possible to give feedback on the recipe in case of acceptance (even if a recipe is accepted, feedback given at that moment might directly steer the next recommendation), and some users pointed out that the dialogue was lacking since they could only express their negative thoughts.

Several participants expressed their need for clarification about specific questions during their profiling step. For example, when asked about their "sports level," they were unsure how to self-assess (some examples could be added). This highlights a missed opportunity to leverage the system's potential for on-the-fly clarification. Note that the EREBOTS framework addresses this very need through its Doctor Agent module. This module was not implemented in the current use case, but it will represent a key component for future work.

5.4 System Outcomes

Eventually, on the one hand, the findings suggest that the EREBOTS framework offers a promising foundation for personalized interactive recommender agents. However, incorporating more advanced user-driven customization features is crucial either contextually or into the core of the system. Ultimately, there needs to be some enhancements to improve user satisfaction while forming the base for the modules residing on the framework. On the other hand, trust has definitely been boosted, revealing an even more prominent desire of the participants to be engaged in the AI decision process. Nevertheless, if not properly managed, the misuse of the framework could lead to a filter bubble. Therefore, it may be necessary to implement a detection mechanism within the core to prevent this issue.

6 Conclusion

In this study, we extended our chatbot framework EREBOTS by incorporating explainability and additional transparency features and explored its potential in a nutrition related case study. We presented the usability of this implementation with a Nutrition Virtual Coach use-case adopted from the literature. The framework combines a user-friendly conversational interface, similar to popular chat applications, with a personalized dietary recommendation engine that provides explanations for inclusion of the user in decision making. Our findings suggest that this framework simplifies development for researchers while fostering positive user experiences and building trust.

In this study, we focus on only short-term of impacts of the developed systems. It would be interesting to examine a long-term usage of such chatbot interaction where the agent proactively engages the dialogue instead. In decision support systems, support mechanisms (such as explanations, as in our case) can be delivered proactively or upon request [5]. As a prospective avenue for research, we plan to intrigue to compare on-demand explanations with default explanations.

7 Acknowledgments

This work has been supported by the Chist-Era grant CHIST-ERA-19-XAI-005, and by *(i)* the Swiss National Science Foundation (G.A. 20CH21_195530), and *(ii)* the Scientific and Research Council of Turkey (TÜBİTAK, G.A. 120N680).

References

1. Adamopoulou, E., Moussiades, L.: An overview of chatbot technology. pp. 373–383 (05 2020). https://doi.org/10.1007/978-3-030-49186-4_31

2. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: Results from a systematic literature review. In: AAMAS, Montreal, Canada, May 13–17, 2019. pp. 1078–1088 (2019)
3. Arsenijevic, U., Jovic, M.: Artificial intelligence marketing: Chatbots. In: 2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI). pp. 19–193 (2019). https://doi.org/10.1109/IC-AIAI48757.2019.00010
4. AWS, A.: Amazon lex. https://aws.amazon.com/lex/, accessed: (Mar. 2024)
5. Aydoğan, R., Jonker, C.M.: A survey of decision support mechanisms for negotiation. In: Hadfi, R., Aydoğan, R., Ito, T., Arisaka, R. (eds.) Recent Advances in Agent-Based Negotiation: Applications and Competition Challenges. pp. 30–51. Springer Nature Singapore, Singapore (2023)
6. Ayub, R., Ghazanfar, M.a., Maqsood, M., Saleem, A.: A jaccard base similarity measure to improve performance of cf based recommender systems. pp. 1–6 (01 2018)
7. Bondevik, J.N., Bennin, K.E., Önder Babur, Ersch, C.: A systematic review on food recommender systems. Expert Systems with Applications **238**, 122166 (2024). https://doi.org/https://doi.org/10.1016/j.eswa.2023.122166, https://www.sciencedirect.com/science/article/pii/S0957417423026684
8. Buzcu, B., Tessa, M., Tchappi, I., Najjar, A., Hulstijn, J., Calvaresi, D., Aydoğan, R.: Towards interactive explanation-based nutrition virtual coaching systems. Autonomous Agents and Multi-Agent Systems **38**(1), 5 (Jan 2024). https://doi.org/10.1007/s10458-023-09634-5
9. Calvaresi, D., Calbimonte, J.P., Siboni, E., Eggenschwiler, S., Manzo, G., Hilfiker, R., Schumacher, M.: Erebots: Privacy-compliant agent-based platform for multi-scenario personalized health-assistant chatbots. Electronics **10**(6) (2021). https://doi.org/10.3390/electronics10060666, https://www.mdpi.com/2079-9292/10/6/666
10. Calvaresi, D., Carli, R., Piguet, J.G., Contreras, V.H., Luzzani, G., Najjar, A., Calbimonte, J.P., Schumacher, M.: Ethical and legal considerations for nutrition virtual coaches. AI and ethics **3**(4), 1313–1340 (2023)
11. Calvaresi, D., Ciatto, G., Najjar, A., Aydoğan, R., Van der Torre, L., Omicini, A., Schumacher, M.: Expectation: personalized explainable artificial intelligence for decentralized agents with heterogeneous knowledge. In: International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems. pp. 331–343. Springer (2021)
12. Calvaresi, D., Eggenschwiler, S., Calbimonte, J.P., Manzo, G., Schumacher, M.: A personalized agent-based chatbot for nutritional coaching. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. p. 682–687. WI-IAT '21, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3486622.3493992, https://doi.org/10.1145/3486622.3493992
13. Chung, K., Park, R.C.: Chatbot-based heathcare service with a knowledge base for cloud computing. Cluster Computing **22**(1), 1925–1937 (Jan 2019). https://doi.org/10.1007/s10586-018-2334-5, https://doi.org/10.1007/s10586-018-2334-5
14. Contreras, V., Marini, N., Fanda, L., Manzo, G., Mualla, Y., Calbimonte, J.P., Schumacher, M., Calvaresi, D.: A dexire for extracting propositional rules from neural networks via binarization. Electronics **11**(24) (2022). https://doi.org/10.3390/electronics11244171, https://www.mdpi.com/2079-9292/11/24/4171

15. Felfernig, A., Burke, R.: Constraint-based recommender systems: Technologies and research issues. ACM International Conference Proceeding Series p. 3 (08 2008). https://doi.org/10.1145/1409540.1409544
16. Følstad, A., Nordheim, C.B., Bjørkli, C.A.: What makes users trust a chatbot for customer service? an exploratory interview study. In: Internet Science: 5th International Conference, INSCI 2018, St. Petersburg, Russia, October 24–26, 2018, Proceedings 5. pp. 194–208. Springer (2018)
17. Freyne, J., Berkovsky, S.: Intelligent food planning: personalized recipe recommendation. In: Proceedings of the 15th International Conference on Intelligent User Interfaces. p. 321–324. IUI '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1719970.1720021, https://doi.org/10.1145/1719970.1720021
18. Google: Google dialogflow. https://www.citedrive.com/overleaf, accessed: (Mar. 2024)
19. Harbola, A.: Design and implementation of an ai chatbot for customer service. Mathematical Statistician and Engineering Applications **70**, 1295–1303 (02 2021). https://doi.org/10.17762/msea.v70i2.2321
20. Hoffman, R.R., Mueller, S.T., Klein, G., Litman, o.: Metrics for explainable ai: Challenges and prospects. arXiv:1812.04608 [cs.AI] (2018)
21. Hulstijn, J., Tchappi, I., Najjar, A., Aydoğan, R.: Metrics for evaluating explainable recommender systems. In: AAMAS, EXTRAAMAS 2023, London, England, May 29, 2023. Springer (2023)
22. Ischen, C., Araujo, T., Voorveld, H., van Noort, G., Smit, E.: Privacy concerns in chatbot interactions. In: Følstad, A., Araujo, T., Papadopoulos, S., Law, E.L.C., Granmo, O.C., Luger, E., Brandtzaeg, P.B. (eds.) Chatbot Research and Design. pp. 34–48. Springer International Publishing, Cham (2020)
23. Lee, H., Kang, J., Yeo, J.: Medical specialty recommendations by an artificial intelligence chatbot on a smartphone: Development and deployment (preprint). Journal of Medical Internet Research **23** (01 2021). https://doi.org/10.2196/27460
24. Magnini, M., Ciatto, G., Omicini, A.: On the design of psyki: A platform for symbolic knowledge injection into sub-symbolic predictors. In: Explainable and Transparent AI and Multi-Agent Systems: 4th International Workshop, EXTRAAMAS 2022, Virtual Event, May 9–10, 2022, Revised Selected Papers. p. 90–108. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15565-9_6, https://doi.org/10.1007/978-3-031-15565-9_6
25. Majumder, B.P., Li, S., Ni, J., McAuley, J.J.: Generating personalized recipes from historical user preferences. CoRR **abs/1909.00105** (2019), http://arxiv.org/abs/1909.00105
26. Mendes Samagaio, Á., Lopes Cardoso, H., Ribeiro, D.: A chatbot for recipe recommendation and preference modeling. In: Marreiros, G., Melo, F.S., Lau, N., Lopes Cardoso, H., Reis, L.P. (eds.) Progress in Artificial Intelligence. pp. 389–402. Springer International Publishing, Cham (2021)
27. Meyer, J.G., Urbanowicz, R.J., Martin, P.C.N., O'Connor, K., Li, R., Peng, P.C., Bright, T.J., Tatonetti, N., Won, K.J., Gonzalez-Hernandez, G., Moore, J.H.: Chatgpt and large language models in academia: opportunities and challenges. BioData Mining **16**(1), 20 (Jul 2023). https://doi.org/10.1186/s13040-023-00339-9, https://doi.org/10.1186/s13040-023-00339-9
28. Montagna, S., Mariani, S., Pengo, M.F.: A chatbot-based recommendation framework for hypertensive patients. In: 2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS). pp. 730–733 (2023). https://doi.org/10.1109/CBMS58004.2023.00309

29. Nadarzynski, T., Miles, O., Cowie, A., Ridge, D.: Acceptability of artificial intelligence (ai)-led chatbot services in healthcare: A mixed-methods study. DIGITAL HEALTH **5**, 2055207619871808 (2019). https://doi.org/10.1177/2055207619871808, https://doi.org/10.1177/2055207619871808, pMID: 31467682
30. OpenAI: Chatgpt. https://chat.openai.com/, accessed: (Mar. 2024)
31. Ornab, A.M., Chowdhury, S., Toa, S.B.: An empirical analysis of collaborative filtering algorithms for building a food recommender system. In: Jain, L.C., E. Balas, V., Johri, P. (eds.) Data and Communication Networks. pp. 147–157. Springer Singapore, Singapore (2019)
32. Prasetyo, P.K., Achananuparp, P., Lim, E.P.: Foodbot: A goal-oriented just-in-time healthy eating interventions chatbot. In: Proceedings of the 14th EAI International Conference on Pervasive Computing Technologies for Healthcare. p. 436–439. PervasiveHealth '20, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3421937.3421960, https://doi.org/10.1145/3421937.3421960
33. Shinde, N.V., Akhade, A., Bagad, P., Bhavsar, H., Wagh, S., Kamble, A.: Healthcare chatbot system using artificial intelligence. In: 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI). pp. 1–8 (2021). https://doi.org/10.1109/ICOEI51242.2021.9452902
34. Singh, J., Joesph, M., Abdul Jabbar, K.: Rule-based chabot for student enquiries. Journal of Physics: Conference Series **1228**, 012060 (05 2019). https://doi.org/10.1088/1742-6596/1228/1/012060
35. Singh, S., Beniwal, H.: A survey on near-human conversational agents. Journal of King Saud University - Computer and Information Sciences **34** (11 2021). https://doi.org/10.1016/j.jksuci.2021.10.013
36. Teng, C.Y., Lin, Y.R., Adamic, L.A.: Recipe recommendation using ingredient networks. In: Proceedings of the 4th Annual ACM Web Science Conference. p. 298–307. WebSci '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2380718.2380757, https://doi.org/10.1145/2380718.2380757
37. Thongyoo, P., Anantapanya, P., Jamsri, P., Chotipant, S.: A personalized food recommendation chatbot system for diabetes patients. In: Luo, Y. (ed.) Cooperative Design, Visualization, and Engineering. pp. 19–28. Springer International Publishing, Cham (2020)
38. van der Waa, J., Nieuwburg, E., Cremers, A., Neerincx, M.: Evaluating xai: A comparison of rule-based and example-based explanations. Artificial Intelligence **291**, 103404 (2021)
39. Wei, C., Yu, Z., Fong, S.: How to build a chatbot: Chatbot framework and its capabilities. In: Proceedings of the 2018 10th International Conference on Machine Learning and Computing. p. 369–373. ICMLC '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3195106.3195169, https://doi.org/10.1145/3195106.3195169