# Fitness Chatbot

SIDDHARTH ADVANI, University of Nottingham, United Kingdom

**ACM Reference Format:**
Siddharth Advani. 2024. Fitness Chatbot. 1, 1 (December 2024), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

The emergence of conversational and generative AI during the past two years has radicalized how users interact with such systems. A demonstration of this change is through the transformation from regular Siri conversations towards using LLMs to produce generative answers for general questions. Thus, utilizing this ideology, this chatbot implements NLP as its base, for primarily guiding users to achieve their fitness goals, whether it be muscle building, losing fat, or powerlifting.

The implementation of this chatbot incorporates Natural Language Processing (NLP) approaches to build a system that is able to classify different user intents to produce a plethora of different generated responses to these requests. For example, the chatbot can engage in small talk, calculate maintenance calories, provide personalized diet and exercise plans, and produce responses to a variety of questions. Furthermore, this chatbot employs sentiment analysis, being able to determine the user's emotions and produce an appropriate response.

Using basic principles of conversational design, the chatbot was built with the implementation of prompt design, contextual awareness, and custom features. The chatbot uses user goals to provide different fitness plans based on their goals and preferences. Furthermore, this chatbot was intended to provide users with a seamless experience to deploy general functionality purposes, as well as user emotions.

This report will provide a comprehensive analysis on the architecture, design, and implementation of the chatbot. Furthermore, this report will also evaluate the performance and usability of the chatbot, identifying its robustness and flaws. Ultimately, this analysis will generally cover the design and user interaction of a fitness chatbot, with a variety of potential implications in the fitness industry.

## 2 Chatbot Architecture

This chatbot is specifically designed for these purposes

- **Answer Small Talk Questions:** Users are able to engage in general small talk
- **Questions and Answers:** Users are able to engage in question-answer feature
- **Fitness Questions:** Users are able to ask any general fitness questions
- **Fitness Recommendations:** Users are able to purchase different fitness plans
- **Analyze Sentiment:** Queries involving user sentiment is detected and responses are adapted depending on this
- **Tracking User's Fitness Data:** Users are able to store their fitness goals, log their progress, such as their weight, diet, and workouts.

Author's Contact Information: Siddharth Advani, psysa15@nottingham.ac.uk, University of Nottingham, Nottingham, United Kingdom.

ACM XXXX-XXXX/2024/12-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

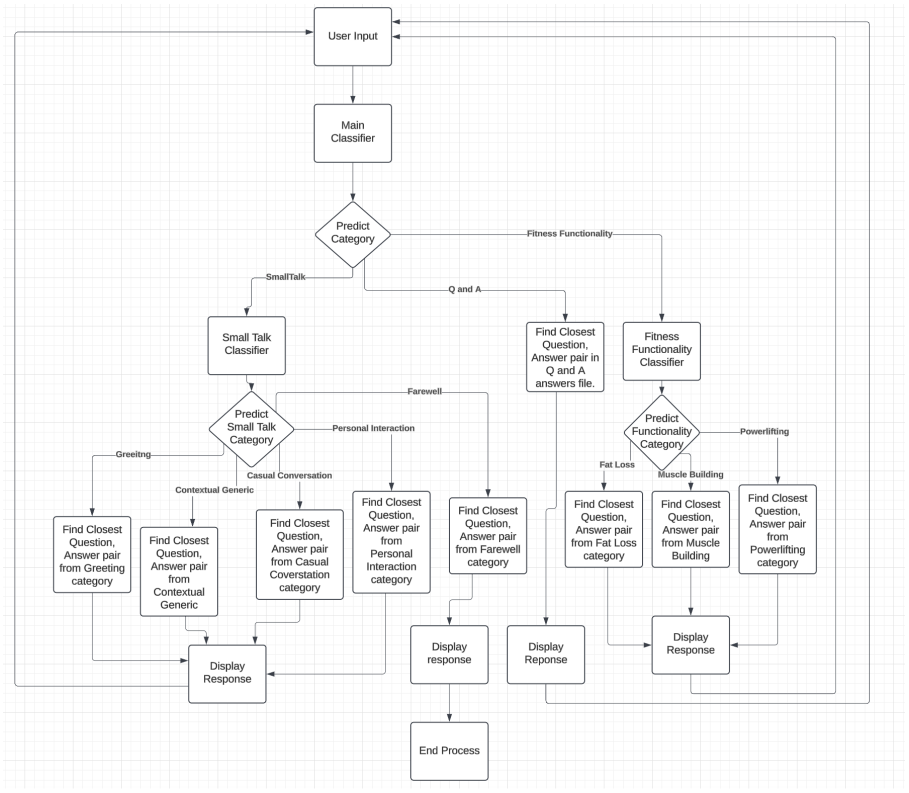

User Input
Main Classifier
Predict Category
Fitness Functionality
SmallTalk
Q and A
Small Talk Classifier
Find Closest Question, Answer pair in Q and A answers file.
Fitness Functionality Classifier
Farewell
Predict Small Talk Category
Personal Interaction
Predict Functionality Category
Powerlifting
Greeitng
Casual Conversation
Contextual Generic
Fat Loss
Muscle Building
Find Closest Question, Answer pair from Greeting category
Find Closest Question, Answer pair from Contextual Generic
Find Closest Question, Answer pair from Casual Coverstation category
Find Closest Question, Answer pair from Personal Interaction category
Find Closest Question, Answer pair from Farewell category
Find Closest Question, Answer pair from Fat Loss category
Find Closest Question, Answer pair from Muscle Building
Find Closest Question, Answer pair from Powerlifting category
Display Response
Display response
Display Reponse
Display Response
End Process

Fig. 1. Input Classification

| | |
|---|---|
| Q&A Dataset (qanda.csv): | Consists of primarily different generic questions |
| Fitness Functionality Dataset (functionality.csv) | Comprises of different general fitness queries |
| Small Talk Dataset (small$_t alk.csv$) : | Encompasses general conversational small talk phrases |

Table 1. Datasets

## 2.1 Classifiers

To answer general queries, the chatbot implements a main classifier and sub-classifiers, to categorize questions, thereby providing appropriate responses.

*Figure 1* portrays a high-level representation of how user input is processed and guided through the main classifier pipeline, which categorizes queries as three different categories - small talk, fitness functionality or Q&A. The initial steps start from the main classifier, which determines if a query is categorized as small talk, fitness functionality or Q&A. To identify the category, the main classifier is trained on three different distinct datasets incorporating questions relating to the three categories, ensuring a broad coverage of user intents.

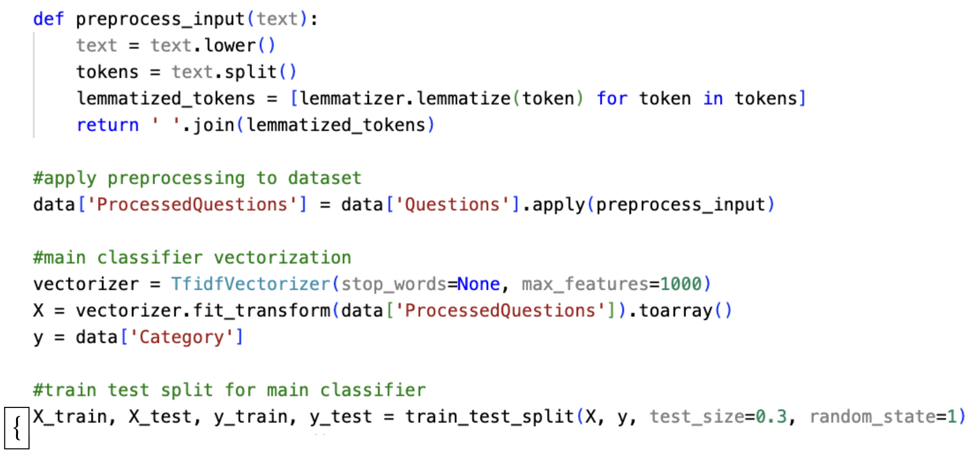
All three datasets were combined into a single dataset, then lowercased, tokenized, lemmatized, and vectorized using TF-IDF (with a vocabulary of 1000 terms), as seen in the snippet below. Each question was assigned a

$$idf(n, N) = 1 + log_2\left(\frac{N+1}{n+1}\right)$$

$$tf(t, d) = log_2(1 + freq(t.d)$$

$$tf - idf = idf(n, N) * tf(t, d)$$

Fig. 2. TF-IDF Equation



```
def preprocess_input(text):
    text = text.lower()
    tokens = text.split()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(lemmatized_tokens)

#apply preprocessing to dataset
data['ProcessedQuestions'] = data['Questions'].apply(preprocess_input)

#main classifier vectorization
vectorizer = TfidfVectorizer(stop_words=None, max_features=1000)
X = vectorizer.fit_transform(data['ProcessedQuestions']).toarray()
y = data['Category']

#train test split for main classifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Figure 3. Preproces Input

category (small talk, fitness, or Q&A). Subsequently, the resulting dataset was split into training and test sets for model evaluation. Testing the accuracy of the model involved three classifiers trained on the data – Supported Vector Machine, Naïve Bayes, and Random Forest.(Reference from Lab 3)

## 2.2 Fitness and Small-Talk Subclassifier

**Sub-classifier Datasets**

The following datasets serve as training material for their respective sub-classifiers.

- Small Talk Classifier (small_talk_classified.csv) – includes queries regarding the following categories – greeting, farewell, personal_interaction, contextual_generic and casual_conversation
- Fitness Classifier (combined_fitness.csv) - includes queries regarding the following categories - powerlifting, fat loss and muscle building

Following the main classifier pipeline, the design implements a secondary subclassifier, which refines fitness questions or small talk questions. This two-tiered approach improves the accuracy and likelihood that user interactions are correctly categorized, and appropriate answers are provided. Assembling these models involved similar steps to the main classifier.

The process involves the main classifier categorizing a query as a fitness related one, the query moves through the pipeline to the fitness sub-classifier, which categorizes the query into either one of the three fitness categories. Refining these categories into smaller sub-categories, ensures consistent accuracy when identifying an appropriate question, answer pairing, thereby allowing users to specify their queires, to emphasize their fitness goals. Similarly,

Table 2. Frequency of Special Characters

| Classifier Model | Accuracy |
|---|---|
| Supported Vector Machine | 99.2% |
| Multinomial Naïve Bayes | 97.3% |
| Random Forest | 98.1% |
| Supported Vector Machine Fitness Sub-Classifier | 93.4% |
| Supported Vector Machine Small Talk Sub-Classifier | 86.2% |
| Linear Regression Sentiment Classifier | 80% |

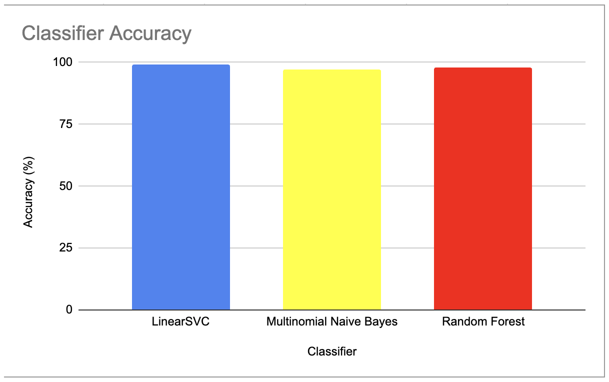


Fig. 4. Comparison of classification models for the Main Classifier

categorization of queires as small talk, follows the pipeline to the small talk sub-classifier, further categorizing input into either one of the different categories.

## 2.3 Classifiers Evaluation

**Analysis**

Referencing Table 2, the results of the main classifier outline the LinearSVC model obtaining the highest accuracy of 99.2%, effectively making it the most successful amongst the other models, in terms of classifying user intents into the three different primary categories.

## 2.4 Data Storage and Management

Figure 5 highlights the Entity-Relationship diagram outlining the structure of the chatbot's database. Builidng the database incorporated the SQLite library, due to its simplicity. The diagram depicts how the database stores essential user data, such as handling user data, their plans, and goals. This figure is essential for guiding data to accessing different fitness features.

The functionality abides by the following pattern - The *Users* table stores basic user data - username, weight, height, age, gender and maintenance calories. This table has a **One to One** relationship with the *User_Goals* table, linked together by the primary key *username*.

The *User Goals* store the user's current fitness logs and goals, with the purpose of retrieving, storing and deleting these values, allowing users to track their progress. Additionally, the *purchases* table stores purchases completed by users, with its foreign key referencing the username key from *Users*. This table also has a **Many to**

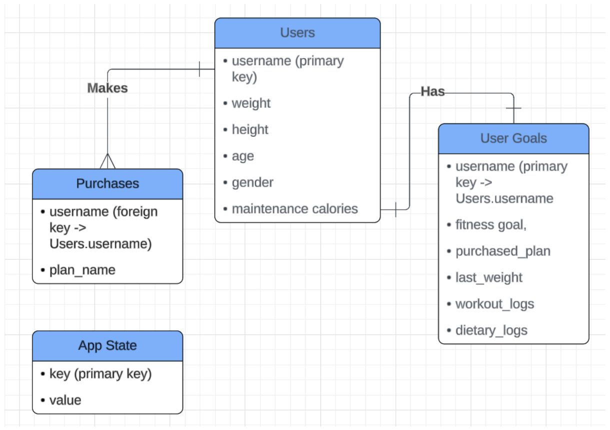


Fig. 5. Enter Caption

**One** relationship with the Users table. Lastly, the *app_state* table is independent and stores the current application data such as last_user, accessing the user from the previous conversation.

### 2.5 Sentiment Analysis

The implementation of sentiment analysis provides the ability to enhance the chatbot's responses, by interpreting the different user sentiments, through specific queries. This includes the ability to classify user sentiment as positive or negative, allowing the chatbot to respond appropriately.

**Use Cases**

**Examples**

- "I am so happy"
- "I am so sad"

These phrases call upon the sentiment analysis classifier, to categorize these sentiments, providing appropriate responses.

**Limitations**

Despite the bot being able to identify emotional keyword triggers and sentence structures expressing sentiment, the model isn't intended to analyze sentiment for general questions, due to lack of development and training to identify the following queries.

**Examples**

- "Show me the time!"
- "Can you please tell me the time?"

**Datasets**:

- Positive Sentiment Data (positive_sentiment.csv) - positive phrases
- Negative Sentiment Data (negative_sentiment.csv) - negative phrases

**Preprocessing Sentiment Data**

The preprocessing steps involved extracting the *text* column, comprising of the different queries, where the adjacent column contains the sentiment labels (positive or negative). The classifier development involved a 75-25 training and testing split. Other techniques involved **Count Vectorizer** for transforming raw text data to a bag-of-words representation, to exclude general stop words, emphasize key features and phrases in the data. (Reference to Lab 4). Furthermore, the bag-of-words counts is converted into TF-IDF scores, addressing the significance of meaningful less commonly used terms.

**Model Training and Accuracy**

Referencing table 2, a Logistic Regression classifier is implemented and trained on the transformed TF-IDF training data. The classifier obtained a score of 80%

## 2.6 Chatbot Features

**Managing User Profiles**

The implementation of managing various different user profiles, is depicted initially through the bot questioning whether it is talking to the same user. Building this involves functions such as confirm_name(), which SELECTS the last active user from the independent table *app_state*. However, if the user would like to change their name, the bot prompts for a new name, calling the create_user() function, initializing a new blank profile in the Users table. The following function references the name, carefully running SQL operations to check for any name conflicts, and allows them to overwrite existing records or select a new name, thereby deleting any persistent fitness data. Similarly, users are able to change their name mid-conversation, calling the *change user name()* function. Current users are therefore stored in the app_state table, using the set_last_active_user function, which identifies the current user interacting with the bot. Ensuring the chatbot to seamlessly continue conversation with the previous user. Ultimately, the chatbot prevents any conflict in the database, and ensuring persistent data and consistency with creating users, ensuring that identity changes are identified throughout the tables, preventing any duplicates.

**Fitness Transaction:**

One of the cornerstone features of this chatbot is the addition of fitness plan recommendations and transactions. The process initially invokes the function collect_fitness_details() , if user data is missing, checked by the start_fitness() function. Collecting fitness details is responsible for obtaining user data, to calculate maintenance calories, ensuring that the system has all the necessary data prior to making any recommendations. Within the start_fitness() function, after collection of user data, fitness goals are set through the set_fitness_goal function, separating user's general data with their fitness objectives. Subsequently, the offer_fitness_plans() function presents users with a choice mechanism, allowing them to select a plan based on their goal. Once selected, confirm_plan_purchase() finalizes the transaction, by checking if the user already owns a plan – user_has_plan() function, and subsequently confirming the purchase, and storing the plan into the database, through record_purchase(). This approach depicts a streamlined approach to purchasing, through the presentation layer depicting the options, the transaction layer incorporating the purchasing of these plans, and the data access layer, which interacts with the database to store information.

Once users have purchased a plan, users can view either their workout or diet plan, using the display_csv_plan function. The architectural logic involves determining the different plans associated with the user – get_plan_files, and then outputting the plans from the CSV file. The diagram below identify the transaction process and plan viewing process.

**Logging Progress** The implementation of this references the user_goals table in the database, which links the users and the user_goals, via the primary key – username. Other keys represent metrics for identifying user progress. For this instance, users can log workouts and diet through the log_workout and log_diet functions, where both initially SELECT the existing workout or diet logs from user_goals. These logs are converted into a list, using commas, allowing new workout or diet logs to be added. Furthermore, these are stored back into the database using the UPDATE call. Additionally, users are able to delete logs, with the delete functions retrieving workout_logs or dietary_logs from the user_goals table. Similarly, these are split using commas and portrayed

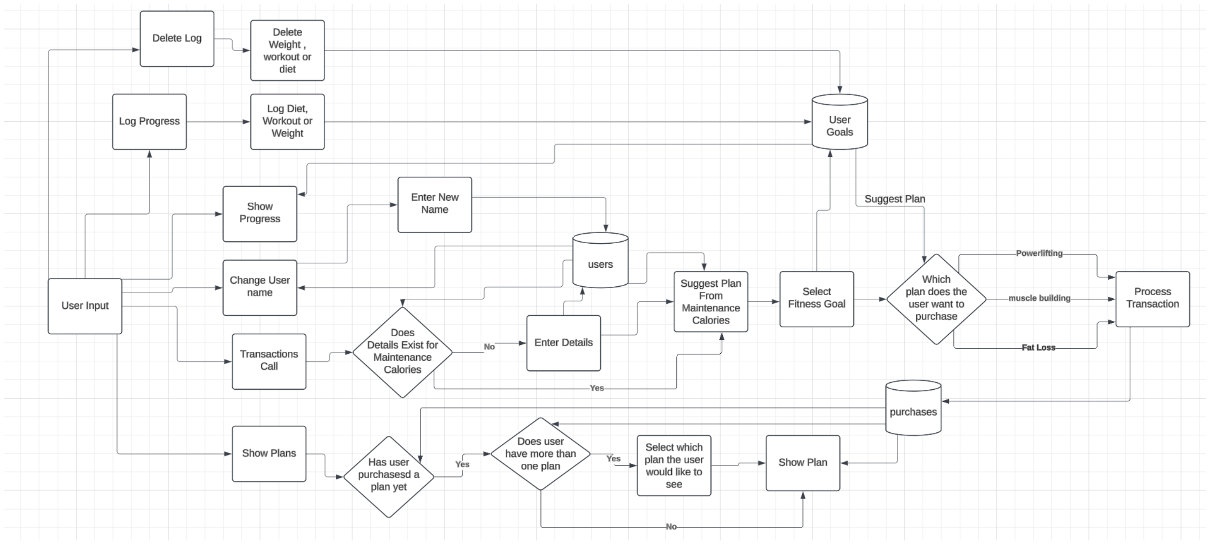

Delete Log
Delete Weight , workout or diet
Log Progress
Log Diet, Workout or Weight
User Goals
Show Progress
Enter New Name
Suggest Plan
users
Change User name
User Input
Suggest Plan From Maintenance Calories
Select Fitness Goal
Which plan does the user want to purchase
Powerlifting
muscle building
Fat Loss
Process Transaction
Transactions Call
Does Details Exist for Maintenance Calories
No
Enter Details
Yes
purchases
Show Plans
Has user purchasesd a plan yet
Yes
Does user have more than one plan
Yes
Select which plan the user would like to see
Show Plan
No

Fig. 6. Chatbot Feature Activity Diagram

to users with index numbers corresponding to entry. Users are able to delete these entries one at a time, with additional measures confirming that the user enters a valid number or can cancel. Similarly, weight is also recorded, allowing the user to consistently update their weight, depicted *log_weight()* function. Lastly, progress viewing is incorporated, calling the display_progress() function, selecting keys – *last weight*, *workout logs* and *dietary logs* for display, ensuring the chatbot serves helpful in guaranteeing achievement of goals.

**Calculating Maintenance Calories**

The ability for the bot to calculate, retrieve and store maintenance calories is implemented, allowing the bot to provide fitness recommendations. As seen in the calc-maintenance_calories() function, which involvs using the Mifflin-St Jeor equation to calculate this value based on user weight, height, age and gender. Storoed in the Users table, the function get_or_compute_maintenance_calories ensures maintenance calories are already added, otherwise re-computes them. Furthermore, *recommend_fitness_plan()* obtains the maintenance calories, recommending specific fitness plans based on the value. Ultimately, the architecture involved in building this ensures a more personalized experience of users, through the ability to store persistent data and provide recommendations.

**Error Handling**

Various instances of error handling is implemented throughout the code, ensuring that the chatbot remains consistent, robust and reliable. These mechanism include schemes such as invalid inputs, database conflicts, etc. Portrayed in the collect_details() function, incorporating try/except validations ensures users enters recognized and appropriate values for either weight, height, age and gender input fields, with retry mechanism involved to ensure the right values are entered. Additionally, handling empty data, such as users not having a plan, is resolved with the offering of plans for the user to purchase. Essentially, these mechanisms are crucial to handle to ensure data integrity and a seamless user experience.

## 3 Conversational Design

The implementation of conversational designs for the chatbot intends to serve as a pipeline to guide users throughout the plethora of features the chatbot can cover. These instances include general question and answers,

$$\text{similarity}(q,d) = \frac{q \cdot d}{\|q\| \cdot \|d\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

Fig. 7. Cosine Similarity

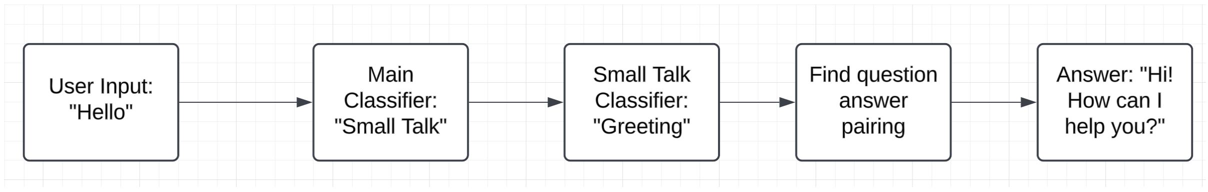

User Input: "Hello"
Main Classifier: "Small Talk"
Small Talk Classifier: "Greeting"
Find question answer pairing
Answer: "Hi! How can I help you?"

Fig. 8. Small Talk Response In Conversation

small talk, functionality, purchasing fitness plans, logging their goals, etc. The implementation of techniques such as personalization and accessibility enhances the user experience, creating a sense of assistance and support.

The structure of conversation is loosely wrapped around these interactions

(1) Small Talk
(2) Q and A
(3) Fitness Functionality
(4) Specified instruction guided features

## 3.1 Greeting

When first interacting with the chatbot, the confirm_name() function is called checking whether it is talking to the same user as the previous conversation, emphasizing techniques such as memory management. Furthermore, the user can select yes or no, to this question, where they will be prompted with a new name, storing it in the Users database, through the create_user() function.

## 3.2 Small Talk and Fitness Queries

Referencing the main classifier, user queries are identified as small talk, moves through the pipeline to the small talk classifier, assigning a small talk category to the input. Once the category has been identified, the chatbot is able to produce a response. As seen in the find_best_response() function, inputs are lemmatized, filtering the responses dataset to include only questions based on the category. The input and all other questions in the training data, are vectorized using calling *vectorizer_response* . Thereby converting text into numerical vectors to compute similarities, through cosine similarity as seen in Fig. 7. Once the similarity scores has been identified, if the similarity value is greater than 0.2, the index of the response with highest score is obtained, and the corresponding response is acquired and returned. (See figure above for cosine similarity equation)(reference from Lab 2) Lastly, a conditional statement checks that if no category was found, an output of ‘I couldn’t understand that’ is returned.

Another feature implemented into conversational design, is the addition of the farewell category within small talk classifier, where the user can enter any farewell query, such as ‘goodbye’ or ‘bye’, which essentially allows the user to quit and end the conversation.

Similarly, the fitness functionality incorporates a similar pipeline, where user input is processed, and the fitness category is predicted, where it will search for similar questions in either one of the fitness categories, and returns the answer which corresponds to the highest cosine similarity score with the fitness question.

## 3.3 Establishing Goals

User's have a plethora of features regarding fitness functionalities they can access. This interaction is initiated when the user intends to start their fitness journey. The following dialogue initiates the start_fitness and collect_details functions, prompting the user to enter their details, such as height, weight, age, and gender to calculate maintenance calories and suggest goals and plans the user should purchase. The addition of these sequentially requested prompts ensures transparency in the conversation, ensuring the bot is able to assist the user. The technique of keyword and trigger recognitions is depicted in the table below, with natural language activation words, such as 'change my name' or 'show my progress', to identify the different functionalities users attempt to access.

## 3.4 User Choices and Commands

Instances of command based intents is depicted, in the transaction process, where the chatbot will initially ask the user to calculate their maintenance calories (if not already calculated), and then ask the user to select their fitness goals. Furthermore, once they have proceeded, another menu displays where the user can then select a plan they would like to purchase based on their goals. The offer_fitness_plans() implements this, which displays a numeric menu (1. Powerlifting 2. Muscle Building, 3. Fat Loss), integrating it for simplicity. Additionally, confirm_plan_purchase() ensures confirmation for user plans, provided they have already purchased a plan prior, thereby reducing errors and enhancing clarity.

## 3.5 Personalization and Contextual Responses

The addition of personalization portrays the seamless and helpful nature of the chatbot. Certain requests, such as "show me my plan", displays user's workout/diet plans from their purchased plan. Through the function - *user-has-plan()*, users are prompted to select either one. Furthermore, this feature is implemented in instances where no transaction has been completed, prompting the bot to offer the different plans for them to purchase. Users are also able to document and delete their current workout, diet and weight logs, emphasizing accessibility and personalisation, by the user being able to track their goals. Referencing table 3, portrays the different user commands to access these features, outlining the ability to store data and retain relevant user data.

## 3.6 Sentiment Analysis

Emphasizing a more human-like conversation, the implementation of sentiment analysis is crucial to provide users with more appropriate empathetic responses. For instance, expressing negative emotions, prompts the sentiment classifier, to identify user sentiment, depicted by the *analyze-sentiment-intial()* function. Assisted by the *mood-response()* function, which responds appropriately based user mood. Negative sentiments are usually responded to with jokes, tips and general expression of empathy, to potentially improve user mood. The interaction design humanizes and personalizes the chatbot for users.

## 4 Evaluation

Evaluating the usability of the bot was performed through task-based usability testing incorporating 4 individuals. The primary objectives involved assessing the ease of use, conversation flow, etc. To evaluate the chatbot the following tasks were performed, with the user assessing the ability to handle these tasks through a survey.

**Tasks Performed**

1. Create a user profile
2. Calculate Maintenance Calories and receive Fitness Plan recommendations
3. Purchase a fitness plan, and view workout and diet plans
4. Log workout, diet or weight and show progress

Table 3. Chatbot Features

| User Command / Keyword | Functionality Action |
|---|---|
| “Change my name” | Changes user name |
| “Show me my diet plan” | Shows diet plan |
| “Show me my workout plan” | Shows workout plan if user has already purchased a plan |
| "I would like to purchase a plan" | Transaction Process, offers workout plans |
| “Calculate my maintenance calories” | Returns maintenance calories if data already exists, otherwise collects user data |
| “Log my workout” | Log workout into user goals table |
| “Log my diet” | Log diet into user goals table |
| “Update my weight” | Log new weight into user goals database |
| “Show my progress” | Shows user progress such as diet, workouts, weight and fitness goal |
| “Delete diet” | Allows user to delete diet entry |
| "Delete workout" | Delete current workout entry |

5. Delete workout, diet entries
6. Small talk questions
7. Q&A questions
8. Fitness questions
9. Sentiment Analysis

**Results**

The survey identified insights into the conversations between users and the chatbot. The survey highlighted the chatbot’s simplicity and conversation flow were rated “easy” to “very easy”, highlighting its ability to guide users effectively. Similarly, conversation flow was also deemed to be between somewhat intuitive and very intuitive. However, issues emerged with spelling, phrasing and command-based instructions, with users claiming issues such as “Needing to phrase questions in a specific way”, and “not recognizing out-of-database questions”. Despite this, error messages and the ability to purchase plans, view plans and other features were also deemed helpful and useful.

**Testing**

To identify and analyze different errors users faced**,** four different types of error categories were outlined – Spelling Errors, Wrong Answers, which occurs when the chatbot isn’t able to answer with an appropriate response to a query, Wrong Commands when calling features, and Name Errors. The graph below higlights the most frequent errors as Wrong Commands, potentially due to rigid command requirements, increasing errors. Such errors caused frustration and obstructed the natural flow of conversation.

**Improvements**

Improvement suggestions include enhancing the error handling mechanism with offering corrective suggestions, and incorporating spell check procdeures. For example – “Did you mean ‘calculate my maintenance calories’, or ‘Please rephrase your question’, instead of the regular try/except blocks with basic error handling. Additionally, improvements through providing a wider range of responses, and the implementation of weekly goals settings, would enhance usability, therefore inceasing user satisfaction with the product.

## 5 Discussion

Based on the testing results, the chatbot was able to effectively complete the listed tasks, thereby increasing general usability. However, the bot struggled with errors such as the rigid command requirements due to its limited natural language processing. Additionally, errors regarding questions providing wrong answers stemmed from the lack of training data in development. The responses of the bot are reliant on lookups from CSV files, reducing the natural flow of conversations. Furthermore, Sentiment also analysis lacks complexity and is only capable of identifying general positive or negative sentiments, also due to its lack of training data.

## 6 Conclusion

The fitness chatbot is generally valuable through instances such as profile development, purchasing fitness plans, recommendations of fitness plans through calculating maintenance calories, viewing fitness plans, tracking progress, basic interaction, and basic sentiment analysis. However, there is room for improvement, by enhancing NLP features, improving error handling mechanism, incorporating a larger training dataset, and general improvements in conversation flow. Ultimately, the foundations of this project include room for potential enhancements in the product, producing a fitness chatbot tailored to anyone based on their goals.

## 7 Citations and Bibliographies

Fischer, Joel. *COMP3074 Human-AI Interaction Lab 4: Conversational Design.*

Clos, Jérémie. *COMP3074 Human-AI Interaction Lab 1: Modelling and Generating Language.* 2023.

—. *COMP3074 Human-AI Interaction Lab 3: Text Classification.* 2023.

Clos, Jeremie. *COMP3074 Human-AI Interaction Lab 5: Evaluation.* 2024.

Clos, Jeremie. *COMP3074 Human-AI Interaction Lab 2: Information Retrieval, Representation and Similarity.* 2024.

Daniel, Jurafsky, and James Martin. *Speech and Language Processing.*