SS2024 | KV Engineering of AI-Intensive Systems

Human or Not

Youhei Enrique Lopez Lopez (K51807823)
Severin Bergsmann (K12008683)

Table of Contents

Goal of Your System

The main goal is to determine whether Large Language Models (LLMs) actually pass a Turing Test - like game.

The main game environment is a chat room with two anonymous users (human-human or human-AI). The users can interact within the chat and their goal is to identify whether the opponent is a human or an AI. Once a user casts an AI vote, a 1-minute countdown begins and the players attempt to convince each other that they are not AI. During this time the players have the opportunity to change their votes. When a human vote is casted, the game ends and the results of the game are displayed.
In addition to the game interface, researchers can also upload LLM endpoints which will be used within Human or Not to gather data on whether their LLMs pass the Turing Test.

Requirements (Summary)

12 Reqs & 7 NfReqs

1. Interface Structure and Game Setup (Req 1-3, NfReq 3, 6)
2. Interaction Modalities (Req 4,12 & NfReq1, 2, 5)
3. Game Dynamics (Req 5-10)
4. Data Management (NfReq 4)
5. AI Endpoint Registry (Req 11 & NfReq 7)

Requirements

Req1 HumanOrNot shall have two views, a start page, and a chat page.
Req2 If a user enters the application, the game shall redirect the user to the start page.
Req3 If a user presses the button with the label “Start Game” on the start page, HumanOrNot shall create a new chat room and connect the user to it.
Req4 If a user writes a message into the chat room, HumanOrNot shall immediately display the message to all users in the same chat room.
Req5 After 30 Seconds the chat room shall display the possibility to raise a claim against other users in the chat room that they are AI or Human.
Req6 When an AI claim is raised, HumanOrNot shall send a system message into the chat room announcing which user claims the other user as an AI.
Req7 When an AI claim is raised, HumanOrNot shall display a countdown for 1 minute to allow the users to debate the decision.
Req8 As soon as a countdown is started, HumanOrNot shall allow users to change, undo, or raise claims.
Req9 When the countdown after an AI claim is finished, HumanOrNot shall stop the game, display an AlertDialog showing the outcome of the game in text form, terminate the chat room, and show a button that redirects to the start page.
Req10 When a Human claim is raised, HumanOrNot shall stop the game, display an AlertDialog showing the outcome of the game in text form, terminate the chat room, and show a button that redirects to the start page.
Req11 If a user presses the “Add Model Endpoint” Button on the start page, HumanOrNot shall include the Endpoint in the pool for chat rooms.
Req12 When no claim was made for a certain number of messages, HumanOrNot shall force both players to make a claim.

NfReq1 If no message is posted within 20 seconds, HumanOrNot shall send a system message with a call to action to communicate with each other. **[constraint]**
NfReq2 If no message is posted 10 seconds after a call to action was raised by the system HumanOrNot shall terminate the chat room and redirect all users to the start page. **[constraint]**
NfReq3 If a user presses the “Start Game” Button, HumanOrNot shall provide a running chat room within 5 seconds. **[performance]**
NfReq4 As long as a chat room is alive, HumanOrNot shall log all chat messages in memory and save them in a JSON format on chat room termination. **[constraint]**
NfReq5 On each received message, HumanOrNot shall make an API call to the Large Language Model **[external interface]**
NfReq6 When a game is started, HumanOrNot shall include at most one AI user to a chat room. **[constraint]**
NfReq7 When a model API is uploaded, HumanOrNot shall run tests to verify whether model API meets endpoint requirements. **[constraint]**

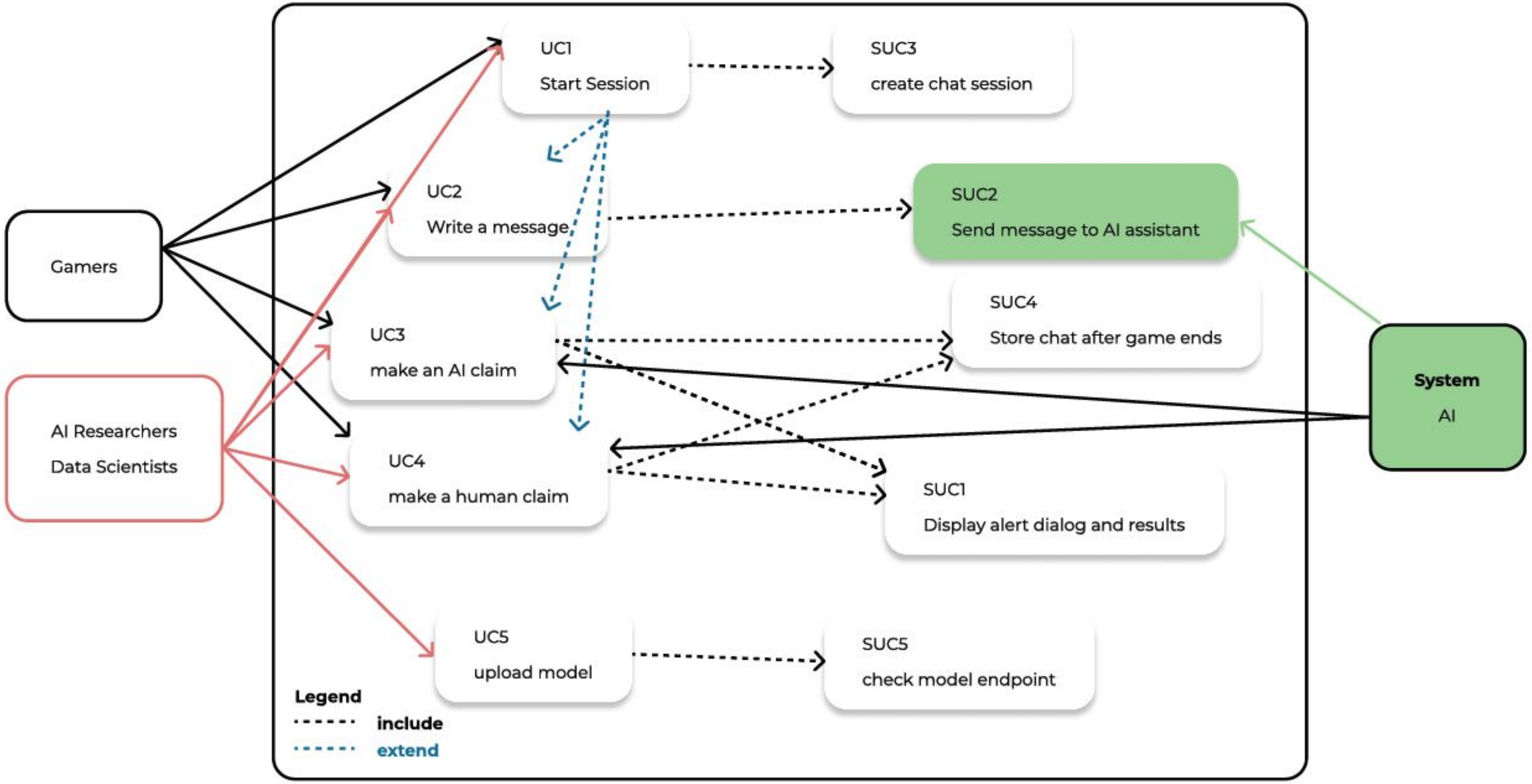
Use Case Descriptions

- UC1 (Start Session): User enters the app and starts the game.
- UC2 (Write a Message): User writes a message into the chat after successfully joining one.
- UC3 (Make an AI Claim): User claims that the other party in the chat is AI.
- UC4 (Make a Human Claim): User claims that the other party in the chat is human.
- UC5 (Upload Model): User uploads his own model.

- SUC1 (Display AlertDialog and Results): The AlertDialog informs users that a human claim was made and the corresponding results.
- SUC2 (Send Message to AI Assistant): A message is created, the message is passed to an AI endpoint and the result is displayed on the chat based on whether the result decided on responding in the chat.
- SUC3 (Create Chat Session): Create a chat session and assign another user or an AI to the chat.
- SUC4 (Store Chat After Game Ends): After the game ended the whole chat is stored in a json file.
- SUC5 (Check Model Endpoint): If a user submits an AI endpoint, then some example requests are made to ensure that the endpoint meets our requirements.

Use Case Diagram

5 UCs & 5 SUCs

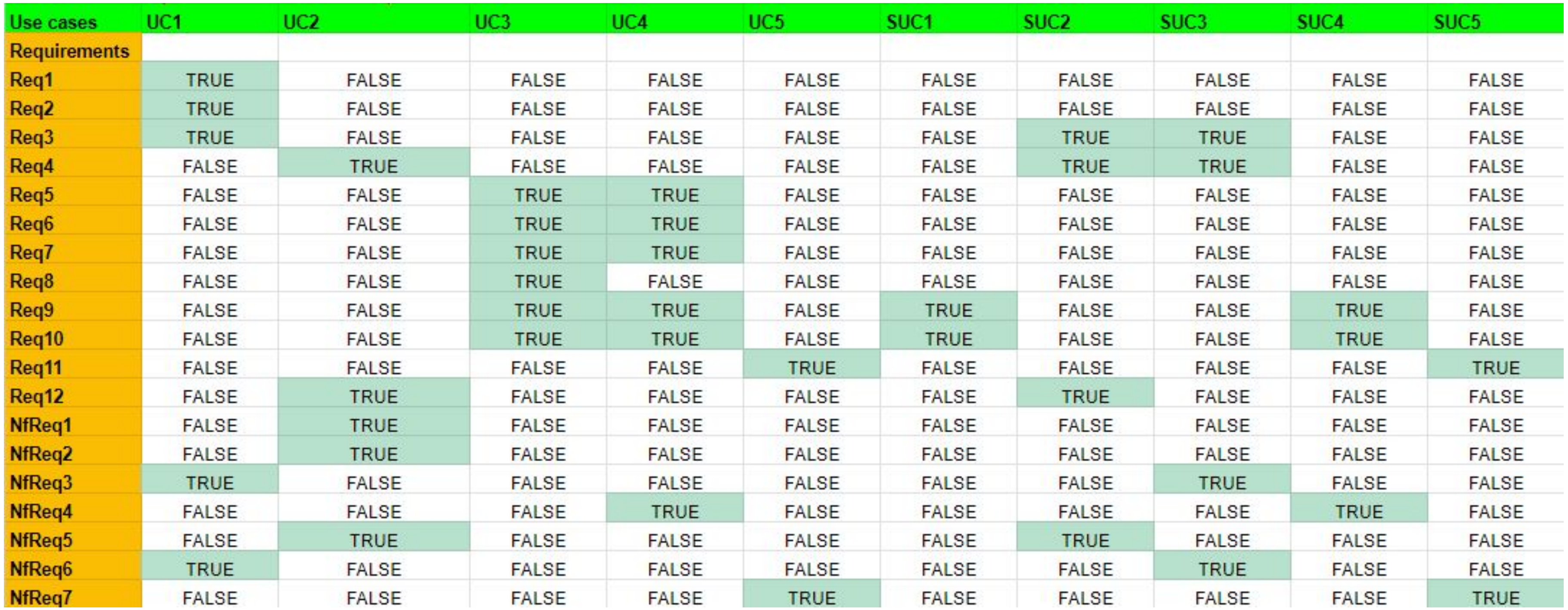


Implemented Use Cases

Link to implemented use cases:

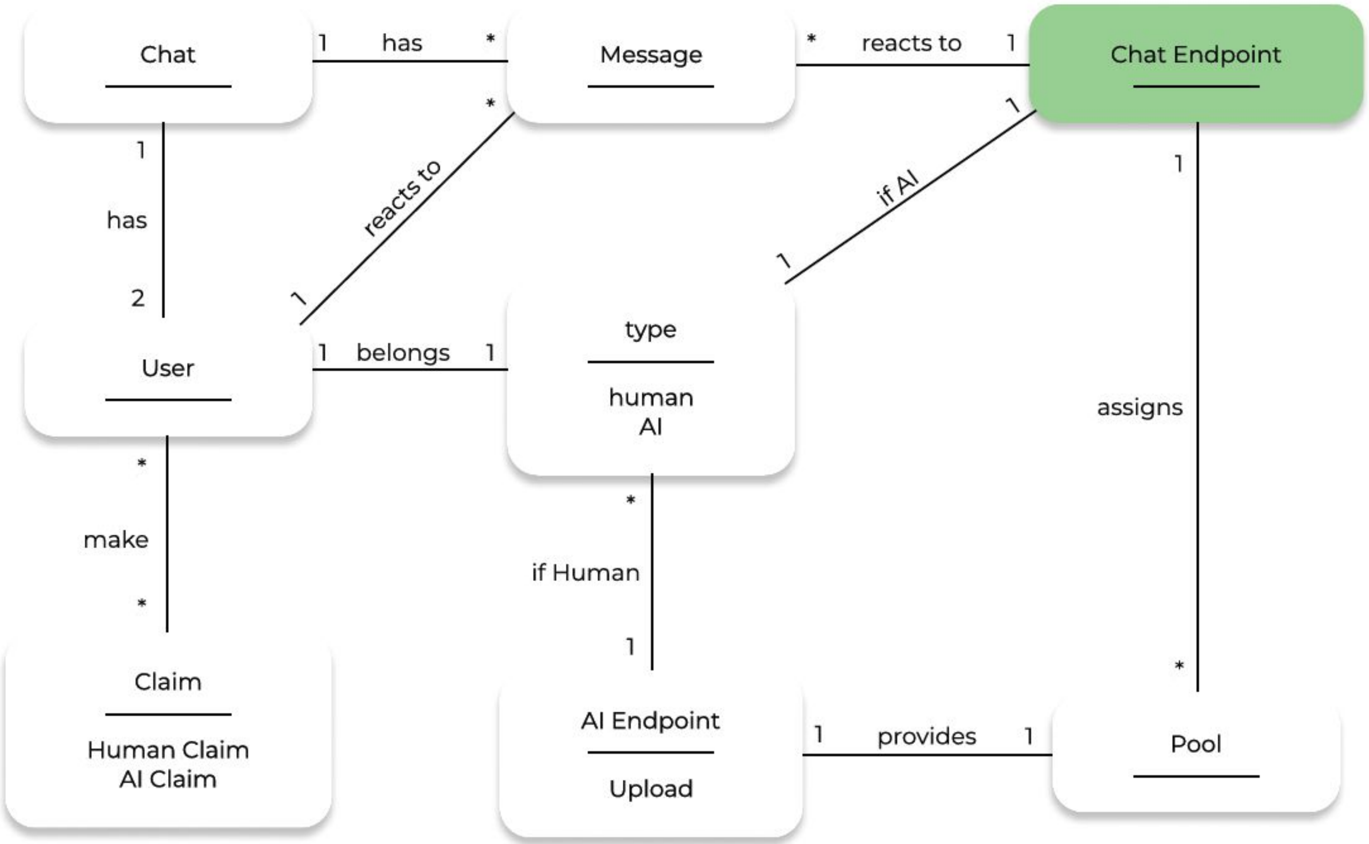
https://docs.google.com/spreadsheets/d/1KfOiY8WAP4dlo2Nggu_tkYvMxUXkt34tu91G7kVmKQw/edit?usp=sharing

Traceability Matrix

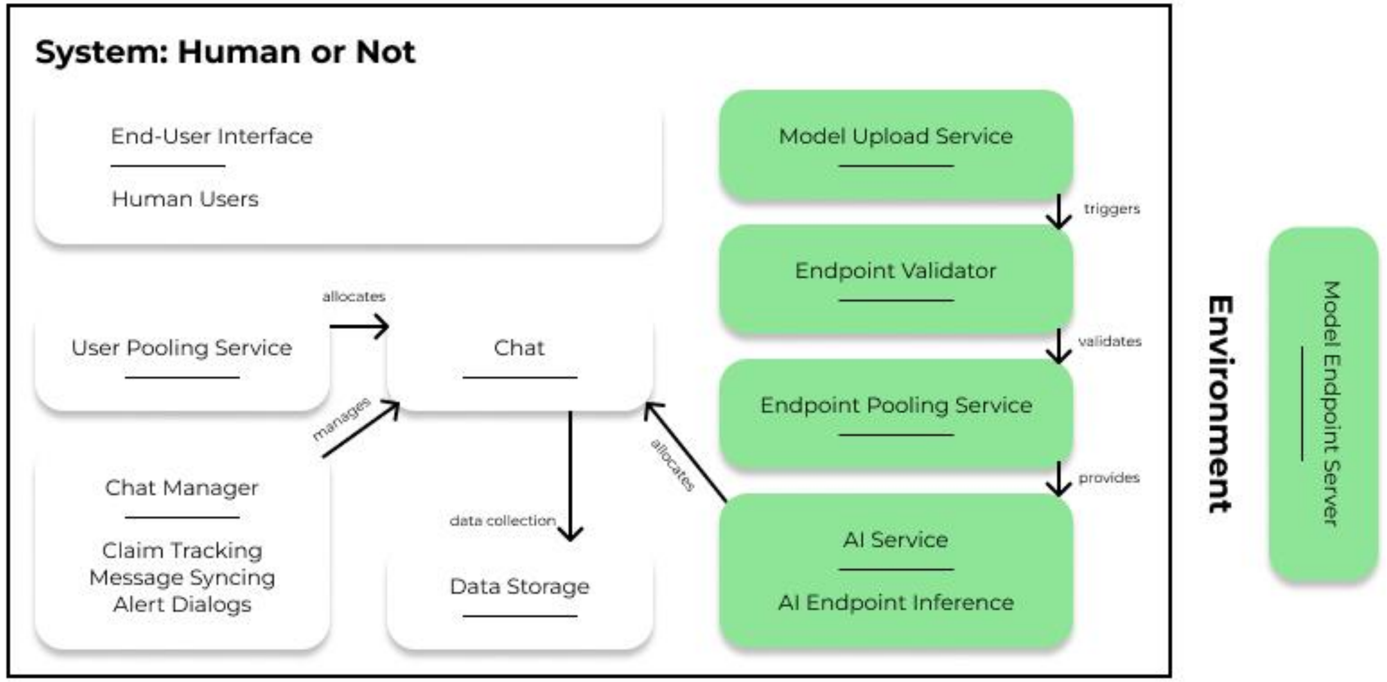


Use cases	UC1	UC2	UC3	UC4	UC5	SUC1	SUC2	SUC3	SUC4	SUC5
Requirements										
Req1	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req2	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req3	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
Req4	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
Req5	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req6	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req7	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req8	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Req9	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
Req10	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
Req11	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
Req12	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NfReq1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
NfReq2	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
NfReq3	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
NfReq4	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
NfReq5	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
NfReq6	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
NfReq7	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE

Domain Model



Architecture Diagram



Legend

⬜ Non-ML Component 🟩 ML Component □ System Boundary

Components Description

1. **End-User Interface**
   a. The end-user interface will have 2 options for the user; either to join a chat session or to upload a model endpoint. The chat session is the main page of the application which allows 2 users to play the game through a chat interface.
   b. [Req1, Req2, Req3, Req11]
2. **User Pooling Service**
   a. The pooling service assigns users to chat sessions, either by grouping 2 human users in a chat session or a human user with an AI user. The pooling service will only be triggered if a human starts a session and if there are more than 1 other active human users.
   b. [Req 3, NfReq3, NfReq6]

Components Description

3. **Chat Manager**
    a. The chat manager is triggered when certain activities occur in the chat. The manager actively syncs messages between users. The manager also triggers alerts when no user has written into the chat, displays the users claims, and shows the results after a session has been concluded. During the chat session, the manager also logs all the chat messages and saves them to a JSON format once the session is terminated.
    b. *[Req4-10, Req12, NfReq1-4]*
4. **Chat**
    a. The main instance and interface where chat messages are displayed. It provides the possibility to raise claims against the other user and write messages to the other user.
    b. *[Req3, Req4]*
5. **Data Storage**
    a. All chat histories and model scores are stored persistently after a game ends.
    b. *[NfReq4]*

Components Description

6. **Model Upload Service**
    a. The model upload service allows users to add their own model endpoints and see how they are performing.
    b. [Req11]
7. **Endpoint Validator**
    a. When a user submits an AI Endpoint via the interface, the endpoint is tested whether it meets the requirements to be used in the game. If it succeeds, it is added to the pool of model endpoints.
    b. [NfReq7]
8. **Endpoint Pooling Service**
    a. This service keeps track of all currently valid endpoints (including the ones that were submitted by a user) and provides them to the AI service.
    b. [Req11]
9. **AI Service**
    a. The AI service is occasionally assigned with another human user to a chat session and should mimic a user.
    b. [NfReq5-6]

Design Questions and Answers

1. What happens if an API endpoint is not accessible anymore during a game? Do we have a fallback solution?
   We terminate the game, display an AlertDialog telling that the game ended due to a connection error, and redirect the user back to the main page.
2. What if human users always win the game? How can the game be made more difficult?
   We can improve our LLM prompting based on gathered data (if humans always win, there must be a clear indication in the chat history on why they are convinced that the other party is an AI).
   a. Solution from 2. directly applies and other solutions:
      - We filter "good" chat histories and finetune a model based on those chats.
      - We provide chat data to researchers and let them finetune their models (and provide new endpoints thereafter).
      - We transform the AI Service into a RAG-based solution where we load the most similar chunks of chats into the context based on the current chat message.
      - Set up a continual learning pipeline to train LLMs on-the-fly with the recent chat histories.

Design Questions and Answers

3. How can we control the model conversation behavior in terms of discretion and sounding human-like?
   Prompting and Guardrailing
4. How do we minimize random claiming after the possibility is unlocked? How can we prevent this?
   The model has the possibility to end the game within the first 30 seconds if there is reason to believe that the other party is not complying to our terms of use: harmful language, non-sense to spare time, etc.
   In this case the model can trigger the “human claim” and end the game before the user can make a claim. The model is also allowed to decide during the game whether any violation of terms is indicated. If it is the case the game will be terminated as well.
5. How can we distinguish between intended claim and random claiming?
   Letting an LLM decide whether chat histories are compliant with our terms of use