A Crowd-sourcing Tool for Collecting Task-Oriented Spoken Dialogues

by

Rishi Salem, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2023

A Crowd-sourcing Tool for Collecting Task-Oriented Spoken Dialogues

APPROVED BY

SUPERVISING COMMITTEE:

David Harwath, Supervisor

Eunsol Choi, Supervisor

Acknowledgments

As my time at the University of Texas comes to an end, I find myself indebted to a great many people. It is due to their support that I have accomplished what I have, and they all have my eternal gratitude.

To my thesis advisors, Dr. David Harwath and Dr. Eunsol Choi, thank you for your guidance over these past few years, and for accepting me into your labs. Additional thanks to Dr. Devangi Parikh, who was a steadfast mentor and friend during my time as an undergraduate.

To my parents (Hemanth and Priya Salem) and sister (Ananya), thank you for supporting me and keeping me happy, healthy and sane throughout my education. Thank you for raising me, supporting me, and helping me grow.

Thank you to my friends who helped me make a home at UT: special thanks to Carmen, for halcyon days of cooking and strategy games; to Isabelle, a beacon of support and giver of great life advice; to Chris, the best project partner and listener to my rambling; and to Anuj, my guide to the world of academic research.

To everyone else, who I will thank in person; while I cannot acknowledge you here, know how much you mean to me. Thank you for helping me become the person I am today!

A Crowd-sourcing Tool for Collecting Task-Oriented Spoken Dialogues

Rishi Salem, M.S.
The University of Texas at Austin, 2023

Supervisors: David Harwath
Eunsol Choi

Dialogue skills are crucial in everyday life, and machine learning tools that model and analyze data can be essential in helping humans with dialogue-based tasks. However, most dialogue models rely solely on text data, given the lack of spoken dialogue datasets. In this thesis, we build the DARE interface, which is designed to collect spoken dialogues. While it can be used with any task-oriented spoken dialogue, we focus on the example case of collecting dialogue on simple negotiation. This paper details the tools used in the development of the DARE interface, the design choices made, and the initial pilot study of using the system to collect spoken negotiation dialogues.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Dialogue is present in every facet of our lives. We engage in dialogue when we stop someone in the street to ask for directions, when we order food at a restaurant, when we have discussions or arguments over disagreements with friends and families, and when we negotiate pay raises with an employer or better car prices with a car salesperson. Since dialogue is such a pervasive element in our everyday lives, knowing how to communicate effectively is a crucial skill.

1.1 Motivations

Since dialogue is so important, it can be valuable to create language models that are able to assist with or evaluate different forms of dialogue. Previous studies have created language models that examine human dialogue and identify useful communication techniques. Such models are useful in assisting humans communicate and improve their communication skills.

However, while it is easy to scrape the internet for text-based dialogues and communications or for single-person audio data, it is much harder to find examples of dialogues in the public domain. As such, the training data for

most dialogue models tend to be purely text-based rather than audio-based.

There is a great deal of paralinguistic information to be perceived from audio data of communications, which models trained purely on text data cannot properly process. Examples of audial cues include tonal inflection, speed of speech, or accents. These can be used to identify sarcasm, the speaker's energy levels and emotional state, and whether the speaker likes the person they are speaking with. For example, consider a chatbot trained to have conversations with human users. A human performing the same task would easily be able to identify sarcasm or wry jokes using their conversation partner's tone of voice. However, a the chatbot may struggle to identify sarcasm without those cues. A chatbot trained with audio data would be better suited to noticing such things.

A model trained to process audio dialogues could help humans learn to verbally negotiate more effectively, or could generate its own audio to perform negotiations on its own. Such a model would have myriad applications in business, politics, and basic conflict resolution, among others.

1.2 Project Goal

Models that are trained on textual data can use existing open-source datasets (Such as Craigslistbargain, discussed in the "Data Analysis" section) and data-collecting tools (i.e. Mephisto, discussed in "Related Works") to obtain training data. However, such open-source datasets and tools do not currently exist for obtaining spoken dialogue data. The purpose of this project

is to create a web interface for collecting audio recordings of goal-oriented dialogues (dialogues in which multiple people are speaking with a specific goal in mind). This interface will be called the Dialogue Audio Recording Environment, or DARE.

For proof of concept, this project focuses on the concrete scenario of negotiations over online purchases. This system can easily be adapted to work with other types of goal-oriented dialogues (for more information, see the "Ongoing and Future Works" section).

DARE attempts to remedy the lack of audio dialogue recording software by providing a platform for laypersons to engage in negotiations, while capturing audio data for research purposes on the server side[1]. It also captures several negotiation statistics, including the length of each negotiation, the number of messages sent, and the ratio of each participant's goal price to the final price agreed upon.

DARE is built using several common frontend and backend tools that make it useable on almost any system. The frontend code, written primarily in Javascript and HTML, is useable on all desktop browsers. While the backend was built and tested on Ubuntu, it is primarily built using Python libraries and is therefore compatible with any system that has downloaded python 3.0 or higher. As such, DARE is easily useable by any researcher who wishes to collect audio recordings of negotiations for future study.

[1]The code used for this project can be found at the following link: `https://github.com/rishiUT/NC-FastAPI`

1.3 Outline

This paper is divided into several sections. The "Background" section will provide a brief overview of the field of dialogue dataset collection and discuss the reasons why an audio negotiation dataset can be valuable. It will also cover the specifications used in the design of the DARE system. The "Technologies Used" section will provide background on the modern web technologies, database tools, and audio recording software used in the creation of this project. The "System Components and Design" will provide an overview of the DARE architecture, detail the structure of a negotiation task, and describe the design choices made in the development of the tool. The "Data Analysis" section will examine test data collected using the tool and compare the collected data to the data statistics in a previous paper. The "Future Works" section will discuss other potential uses for the DARE interface, as well as improvements that can be made to increase the potential use cases and improve accessibility. The "Conclusion" provides a brief recap of the topics covered in this paper.

Chapter 2

Background

2.1 Related Works

There have long been efforts to model human dialogue and communication tasks. To train machine learning models to understand and replicate human dialogue behavior, researchers have needed to collect datasets of dialogues.

2.1.1 Text-based Information Seeking Dialogue Datasets

There are many dialogue-based datasets. Two datasets that stand out for modelling interactive conversations through long back-and-forth dialogues include the Question Answering in Context (QuAC) dataset [2] and the Wizard of Wikipedia dataset [4].

QuAC is a dataset of dialogues where one participant answers questions from another. Each dialogue has several questions, some of which do not make sense without the context of the full exchange. This dataset is used to train models that can keep track of an entire context and use it to inform their responses.

Wizard of Wikipedia is a dialogue dataset in which one user answers

in-depth questions on a specific topic from another user. The answerer uses a Wikipedia page as a source for their responses. This study also requires models that can keep track of context, but specifically require that the model look up context from outside the conversation.

2.1.2 Task-Oriented Dialogue Applications

The next step from context-based dialogue models is models that can use context-based dialogue to perform a task. One example of this is Cicero, an AI agent created by researchers at Meta to play the game Diplomacy with human players at a high enough level to pass as a human [7].

Diplomacy is a game requiring players to negotiate in a multi-turn format to form alliances and agreements. Players negotiate, make decisions, and negotiate again. An agent that performs well at this game would need to keep the conversation context in mind for each negotiation, and the game context to respond properly to kept agreements and betrayals. Cicero was able to play 40 games on an anonymous Diplomacy server without getting noticed as an AI and performed in the top 10% of players.

2.1.3 Spoken Dialogue Applications

While many datasets and models currently work with text-based data, audio dialogues are far less explored. That said, audio communication can be very valuable in conveying information that words alone cannot. For example, McFarland et al. [20] discovered that models trained on audio data of speed-

dating events could use several cues, both within the dialogue and in the audio recordings themselves, to determine when someone was attracted to the person they were speaking with. Some of these purely audial cues include raised volume and laughter for men and raised and varied pitch for women; these features indicated attraction independent of the words used.

2.1.4 NLP Crowdsourcing

Training models to make use of audial cues for predictions and communication is clearly valuable, but such models would require datasets for training purposes. However, collecting datasets for use in training models is difficult, especially when considering the large quantity of data necessary for machine learning.

A popular solution is crowdsourcing, or getting non-experts to perform a task to collect data. Amazon Mechanical Turk (AMT, or Mturk) is an incredibly popular crowdsourcing platform; it was used to gather data for QuAC, and is common enough that papers are written on the best practices to use for tasks published on Mturk [14]. In fact, an ACL meeting on crowdsourcing also focused on best practices for designing Mturk tasks [1].

Since AMT is such a popular platform, there are many additional tools and templates that exist to facilitate data collection using Mturk Tasks. One such tool is Mephisto [27], which not only allows for combining data collection, model training, and model testing in a single tool, but also provides several useful templates for Mturk tasks. Mephisto even includes a template for a

text-based chat task, which could feasibly be used for negotiations.

Another tool that can be used with Mturk is the Web-Accessible Multimodal Interface (WAMI) toolkit [9]. The WAMI toolkit provides a framework for creating and deploying Web-Accessible Multimodal Interfaces. It can be used to easily create and use Graphical User Interfaces (GUIs) alongside existing multimodal dialogue collection tools. Additionally, the interface can be used to record spoken dialogue from a single user.

Spoke is another framework for spoken language technologies which can be used to create websites for spoken language research purposes [26]. Spoke uses a modular approach to make it easier to build tools; for example, frontend modules can help create a functional, easy-to-understand UI, while backend modules can handle audio recording or speech recognition, etc. Spoke can be used to build tools compatible with Amazon Mechanical Turk; the example in the paper collects utterance data by having turkers record audio of given sentences.

However, such tools are infeasible for use in a multi-user audio dialogue collection setting. Mephisto in particular is very inflexible and difficult to use for tasks that do not match any of the templates. Changing Mephisto's text messaging task to collect audio dialogue would be impractical to the point that creating a platform from scratch is more feasible. WAMI and Spoke allow for single-user audio collection, but cannot collect audio data of multiple users engaging in a common task.

2.1.5 Negotiation Data Collection

The task of interest in this paper is negotiation. Datasets exist for text-based negotiations; one such dataset is the CraigslistBargain Dataset created by He et al. [10]. This dataset contains negotiations by human participants over Craigslist items, where both participants were shown a real listing scraped from Craigslist and provided a goal price. The researchers found that, as compared to other dialogue datasets, CraigslistBargain resulted in "longer dialogues and more diverse utterances".

The tool described in this paper is intended to create a dataset equivalent to CraigslistBargain with audio messages instead of text. As such, more details about the results found in the CraigslistBargain dataset can be found in the "Data Analysis" section along with comparable data collected using the DARE interface.

2.2 Project Goals

The goal of this project is to create a tool that can easily be used by researchers to collect negotiation data from volunteers. To achieve this goal, DARE needed to meet a series of requirements.

DARE needed to have a simple interface, both on the backend and the frontend. Researchers must be able to install and run the backend server without a deep understanding of the underlying code, and the installation process should be streamlined and as quick as possible. Additionally, the

frontend and user interface should be simple enough for general users to be able to negotiate easily and efficiently, so that users would be willing to repeatedly use the service to provide data.

There are also performance requirements. The server must be able to handle user loads as high as possible; it is expected that the server be able to serve hundreds of users without overloading. Additionally, negotiations can already take several minutes due to the time required to record and listen to audio messages, so it is critical to reduce message latency as much as possible. Finally, since negotiations require near-constant connectivity and lots of time, the server must be able to handle unstable connections, user refreshes, and lost users in a timely manner.

Finally, it is ideal to have as few dependencies as possible. Code with lots of dependencies is difficult to maintain, as such dependencies will not always be maintained and future users will need to find workarounds to manage dependencies that are no longer usable. Additionally, dependencies often form dependency chains, in which imported code libraries require other libraries to function; this can make debugging failing dependencies very difficult. Reducing the initial number of dependencies reduces the workload of future users and developers.

2.3 Summary

In this chapter, we discussed related works in the area of dialogue data collection and audio recording. We also discussed why previous, readily-

available resources are insufficient for gathering multi-user, task-oriented dialogue. Finally, we discussed the goals and requirements of the DARE interface, which was created to fill this niche.

We will now discuss the tools, technologies, and code libraries used to meet these requirements.

Chapter 3

Technologies Used

The DARE interface was created using a combination of existing tools, languages, and code libraries. In this section, we will provide an overview of the components and technologies used for the project and the reasons why the technologies in question were chosen. We will discuss backend technologies used to build and maintain the web server, frontend coding languages used to create user-friendly webpages, interfacing tools to connect frontend and backend, and additional tools used to gather participants and create negotiations.

3.1 Backend

The backend of the system is responsible for handling connections and pairing, collecting and recording the audio sent between users, processing the audio so it's in an acceptable format, and automatically generating transcripts of the messages sent. It uses the following components to do so.

3.1.1 FastAPI

The backend server is built using FastAPI. FastAPI is a high-performance web framework with several built-in features optimal for multi-user chat plat-

forms. It has levels of performance comparable to NodeJS and Go, and is one of the fastest Python frameworks available [19]. It is asynchronous, has built-in support for websockets and cookies, and is easy to use for development. It is also easy to integrate python libraries such as FFmpeg and Whisper, since the framework uses Python for the backend.

FastAPI can handle large numbers of connections using asynchronous code execution. This allows a single server to handle several users concurrently, but unlike multithreading and load-balancing (the most common alternatives), this ensures that all users are handled by the same processor. This helps prevent bugs in the pairing system and ensures that users looking for negotiation partners will be able to find partners.

FastAPI also provides built-in support for websockets and cookies, both of which are essential for managing a chat application. Built-in functionality drastically reduces the overhead caused by dependencies. Other frameworks like Flask require add-ons like flask-websockets, which makes development and maintenance more complicated as time goes on. Dependencies require constant maintenance, can proliferate by requiring dependencies of their own, and can result in dependency conflicts in which two dependencies require different versions of the same software. Built-in functionality avoids all these issues.

Since FastAPI is built on Python, it is easy to slot in additional libraries as necessary. It is also easy to remove and replace them if they are deprecated in the future.

Finally, FastAPI allows for quick and simple development. The base functionality of the server, including pairing, recording, byte-data sending, and asynchronous connections, were completed in the first week of development. Additionally, the code can be run on a local server using uvicorn for efficient testing. The ability to code quickly and test locally makes it much easier to develop new features, which is useful when developing a brand-new tool from scratch.

3.1.2 FFmpeg

FFmpeg is a powerful open-source tool used for video and audio processing[8]. It can be used on virtually any platform and can be used to edit, trim, and compress audio and video files, etc. It is used in this project to convert incoming audio from the .ogg format to .mp3, as well as to reduce the sample rates and the number of channels. This allows the audio to be stored more compactly while maintaining enough sound quality for machine learning purposes.

3.1.3 Whisper

Whisper is an automated speech recognition (ASR) system used for speech-to-text purposes[29]. It is used in this project to automatically generate transcripts of all incoming audio from the negotiations.

3.1.4 Nginx

Nginx is an open-source web server that can be used for reverse proxying and load balancing, among other purposes. It can handle hundreds of thousands of concurrent connections [24]. When compared to the Apache servers that are the default on Linux machines, it has far higher performance [15].

It is used in this project to handle incoming traffic to the hosted server. As a load balancer, it allows a far greater number of users to access the server at once and grants it greater stability. Reverse proxying is also helpful from a security standpoint and allows for better caching for static files [25].

3.1.5 Long-Term Storage

3.1.5.1 PostgreSQL

The project stores all user data in a SQL database. For this project, a PostgreSQL database was used.

PostgreSQL is fully ACID compliant [6]. Of the guarantees this provides, Isolation and Atomicity are the most essential for the purposes of this project due to the pairing system; at any given moment, the system must know which users are already paired and which can still be paired, to ensure that no user is paired with multiple other users. Eventual consistency, the weaker guarantee provided by many other SQL databases, will not be sufficient.

3.1.5.2 File Systems

All audio is stored in the server's file system in an output folder as MP3s (as described in the "Audio Processing" section). Metadata for each negotiation is also stored in the file system; this metadata includes a list of all voice messages in the negotiation in the order they were sent, as well as the users involved in the negotiation, the amount offered, and whether the negotiation was successful.

3.2 Frontend Tools

The frontend is responsible for creating user interfaces and handling user interactions. The tools described in this section are used for creating aesthetically-pleasing, intuitive webpages, collecting audio, and guiding users through the task.

3.2.1 HTML, Javascript, and CSS

3.2.1.1 HTML

The frontend user interface is made of several webpages. These webpages are created using HTML, a markup language often used in web development. Browsers use HTML files to create the textboxes, buttons, images and other visual elements commonly found on webpages.

The user interface requires the creation of several webpages, many of which share common elements. Creating individual HTML pages for each webpage would be inefficient; as such, the website uses HTML templates to

create several HTML pages from a similar templated layout. The templates used in this project were created and managed using the templating language Jinja. Jinja uses just-in-time (JIT) compiling to reduce memory usage and allows for template inheritance to provide more flexibility in template creation [17].

3.2.1.2 Javascript

While HTML is useful for creating static webpages that display text and images, it is less useful for the creation of dynamic elements such as clickable buttons, pop-up messages, timers, etc. JavaScript is a scripting language that works with HTML to provide more dynamic elements [16]. Javascript is a critical element in this project; it is used for everything from recording audio messages (see "Audio Recording") to handling websocket connections (see "Interfacing").

3.2.1.3 Bootstrap and CSS

Cascading Style Sheets, or CSS, is a language used to completely control the visual appearance of a webpage. CSS can specify the styles used in an HTML document; Specific fonts, colors, background colors and sizes can be defined for all elements in an HTML document, including titles, text, buttons, etc. Additionally, CSS can be used to control where and how text loops around, divide the webpage into sections, and even to create animations [3].

Bootstrap is an open-source CSS library commonly used to create co-

hesive, aesthetically pleasing webpages with minimal setup. Bootstrap uses a responsive grid system to create webpage layouts and divisions that adapt to changing browser window sizes and can be used on various device layouts [18]. This project uses bootstrap to handle the CSS for all created webpages, as well as to provide images like the "recording" and "sending" symbols used on the buttons in the negotiation user interface.

3.2.2 Audio Recording Libraries

3.2.2.1 MediaDevices

The MediaDevices API contains the getUserMedia function[21], which is used to get users' permission to use media inputs (their device's microphone, or any other microphone). This function is crucial for any platform that records audio data.

3.2.2.2 MediaRecorder

Given an audio input, the MediaRecorder API can be used to record audio without additional plugins [5]. It is used in this project to record users' voice messages at the press of a button. The basic functionality (start recording, stop recording, store data) are compatible with all web browsers [22], making this API extremely useful.

3.2.2.3 HTML Audio Element

The HTML audio element can be used in any HTML page to grant users access to an audio file[11]. It is used in this project to allow users to

play back recorded audio and to listen to audio messages sent by the users' negotiation partners.

3.3 Interfacing

3.3.1 Websockets

Websockets are a communication protocol used for bidirectional communication between two computers. It is used to allow for steady, bidirectional communication between web browsers and web servers, which otherwise only allow for communication when the browser sends a request to the server. [28]

Note that while constantly sending requests (polling) is an option, it is unwieldy for real-time communication; long-polling, or sending infrequent messages, makes it difficult to get messages from negotiation partners, while short-polling sends a lot of HTTP messages and has greater overhead. Websockets are a much more convenient solution, as they allow both the server and the browser to send messages if and only if necessary.

3.3.2 HTTPS

HTTPS is an extension of HTTP that encrypts data for greater security[12]. Webpages with HTTPS connections use the Transport Layer Security (TLS) protocol to encrypt communications between the web browser and the web-server hosting the webpage[13]. This prevents third parties from reading and editing the communications en route.

Secure connections are essential for any web platform, but are espe-

cially important when handling sensitive data such as audio recordings. Most modern browsers do not allow websites to make certain API calls when using insecure connections; these API calls include geolocation services, push notifications, and most importantly, access to audio devices. The mediaRecorder and MediaDevices cannot be used without a secure, HTTPS-certified connection.

3.4 Summary

Thsi chapter presented an overview of the tools, technologies and code libraries used in this project; these ranged from frameworks used for coding webservers to APIs provided by web browsers. Now, to better understand how these tools were used and why, we will explore the design of the system, starting with the intended user experience.

Chapter 4

User Experience

4.1 User Story

Assume you are participating in a negotiation on the DARE platform. What might your experience be?

When you first connect to the website, you will see a message asking for your permission to use all recorded data in scientific research. You'll need to consent to this to continue, so assume you do so.

You'll then have a chance to read the negotiation instructions and familiarize yourself with the task; once you're finished, you can indicate that you're ready to begin, then wait for a partner to be found.

Once you're assigned a partner, you can start the negotiation. You'll be shown a webpage with information about an item, a goal price, and an interface for sending and listening to recorded voice messages. Let's assume you're the buyer; you'll want to convince your negotiation partner to sell the item for as low a price as you can manage.

Since you're the buyer, you start the negotiation with the first message. You could start with pleasantries ("How was your day?"), ask for more details on the product ("How big is this bedframe?"), jump right into suggesting

prices ("I'll give you fifty dollars for it"), or a combination of the three. After you send your message, your partner will send a response.

You and your partner will go back and forth a few times. You both have different goals in the negotiation and will likely have reasons for sticking close to your goals (You need a decent bed within a college student's budget, and they are raising money for their sick dog's medical bills). Eventually, you'll find a compromise you can both be happy with. Once you do, you'll send an offer through the interface; depending on how the conversation went and the offer you sent, they can accept it or reject it.

At the end of the conversation, you'll be given a score for the negotiation (If you're performing this negotiation on a worker platform like mechanical turk, you'll be paid!). Since you reached an agreement and your partner agreed to your offer, you get a pretty high score! You also get to see your partner's score, which is also pretty high; it looks like reaching an agreement is pretty good for both of you.

4.2 Negotiation Structure

For the sake of clarity, let's clarify what a negotiation entails.

Each negotiation has two participants. Each will be assigned a role; one will be the buyer, and the other the seller. The buyer and seller will be negotiating over a single item from a Craigslist listing, which comes with information about the item and a listed price to start the negotiation with.

The buyer and seller also each have a goal price; the buyer's goal is strictly less than the seller's goal.

The buyer and seller will exchange voice messages until they reach an agreement. Once they have finished negotiating, the buyer will send a single monetary offer to the seller, who can accept or reject it. The negotiation will then end immediately regardless of whether the offer was accepted or rejected.

4.3 Bonus Calculation

In a real negotiation situation, there are stakes that drive conflict and encourage users to push for as good a reward as possible. However, in a simulated negotiation, the stakes are much lower. To provide stakes to the user, the DARE interface provides a bonus to users that perform well in their negotiation – a monetary bonus for users of Amazon Mechanical Turk (or any other worker platform used with the DARE interface) and a simulated, score-based bonus for non-paid users.

Once an Mturk user has completed the negotiation, they will be provided a monetary bonus that scales based on their performance. The bonus ranges from $0.00 to $2.00, with participants who reached an agreement with their partner receiving a bonus between $1.00 and $2.00.

To receive a bonus, the user must have reached an agreement with their partner. If the seller rejected the buyer's offer, the bonus for both participants is zero. If the offer has been accepted, both participants receive an initial

bonus of $1.00.

Participants also receive an additional bonus for performing well. This is defined as deciding on an offer that is closer to the participant's goal price than to their partner's goal price. The added bonus is an additional $1.00 split between the participants proportionally to the distance between the offer and each participant's goal. The bonus will be calculated as $|(P - O)/(S - B)|$, where seller's goal price is S, buyer's goal price is B, the offer price O, and the participant goal is P.

For example, consider a buyer with a goal of $1200 and a seller with a goal of $800. If they agree on a price of $1000, each will get a bonus of $1.50 ($1.00 for reaching an agreement, and $0.50 each from the added bonus). However, if they agree on a price of $1100, the buyer will receive $1.25 and the seller will receive $1.75.

Note that any bonus above $1.00 is rounded down to $1.00, and any negative bonus is rounded up to $0.00. If the buyer and seller in the previous example agreed on a price of $1250, the buyer would receive $1.00 and the seller would receive $2.00.

If this platform is used without the added layer of Amazon Mechanical Turk (i.e. users connect directly to the website hosting the task), the bonus shown to participants is no longer shown as a dollar value (since it is assumed these participants will not be paid). Instead, they are shown a "score" equivalent to the bonus they would have received, with one point given for every

cent they would have earned.

4.4 Pairing System (User Perspective)

The pairing system, from the perspective of the users, is very straightforward. Users are paired with the first available user the system can find, then given a random item for both users to negotiate over. Each user is assigned a role – one is the buyer and the other is the seller – and each is given a goal price to negotiate towards.

Users keep the same partner until they finish the negotiation or one user disconnects entirely. In the case of a user's partner disconnecting, they are asked to leave the negotiation and start the pairing process entirely from scratch.

4.5 Disconnect Handling (User Perspective)

Except in extreme cases, users are not meant to notice when disconnections occur. If a user needs to reconnect to an in-progress negotiation or refresh the webpage, the negotiation will not change at all; all the same messages will be sent and received, they will see the same item to negotiate over, and the conversation timer will still be counting down from the start of the negotiation. Additionally, if their partner sent any additional messages while they were disconnected, those messages will be visible as if they had been connected all along. If an offer has been sent, that information is also re-sent

to the user.

The only time a disconnection affects the user is if a user disconnects for longer than a given timeout period (which is variable, but currently set to two minutes). If this occurs, the system assumes the user cannot reconnect and marks the negotiation as incompletable. As mentioned above, in this situation, the partner of the user who disconnected is asked to join a new negotiation.

4.6 Summary

In this chapter, we explored the intended user journey, including how they would engage in a negotiation, what kinds of bonuses they can receive for good performance, and the user disconnection experience. Having seen the system from the perspective of the user, we will now examine the design of the underlying systems to see how this user experience is created and displayed. in the next chapter, we will examine the design from the system perspective.

Chapter 5

System Components and Design

5.1 Overview

We've examined the system from the user's perspective. How is all this implemented from the system's side? In this chapter, we will examine the system design choices that allow it to function as described. We will start with an overview of the process flow, discuss the design elements of the frontend, and clarify how the communication layer and the backend implement the necessary functionality for communication, pairing, and data management.

5.1.1 Process Flow

To better understand the system's technical design, let us re-examine the process flow described in the user story – this time, we will examine it from the perspective of the DARE interface's server. A visual diagram of this user flow is shown in figure 5.1. A flowchart version is shown in figure 5.2.

Each user will independently connect to the DARE webpage. They will all be assigned a user ID, which will be added to the server's internal database. They will also be assigned a more permanent ID which is used to identify repeat users; this can be used to identify users that often perform well

to better study their techniques.

There may or may not be another user waiting to be paired. If not, the user will be asked to wait until another user can be found. Once two users are waiting to be paired, the server will create an active conversation data structure, assign one user to be the buyer and the other to be the seller, and inform the users that they have been paired.

The users are then provided with the negotiation page, which contains information about the item and the goal price for the negotiation. The buyer sends the first voice message, and the seller can send responses; they continue taking turns until the buyer is prepared to send an offer. Note that the buyer can only send an offer after at least four messages have been sent in the negotiation. Additionally, the buyer can only send one offer; once it is accepted or rejected, the negotiation is over.

Sent messages are sent through the server. The server stores all messages internally before forwarding them. The offer is also stored internally, as is the offer response (either "accept" or "reject").

Once an offer has been sent and responded to, the server marks the negotiation as complete, saves all conversation data internally, and sends the users the "negotiation complete" page.

5.2 Client-Side layer

During the negotiation task, participants will interface with the negotiation platform website. The website uses several webpages to help users through the negotiation process described in the "Process Flow" section.

5.2.1 Home Page

When users view the task description page, before they accept the task, they will be informed that the data collected in the task will be used for research purposes (figure 1.1). They will be given the choice to accept the task and grant consent to use the data thusly, or to back out.

The website is also equipped to gather consent from users who are not using MTurk. Such users will be shown a pop-up when they first use the site; by clicking through the pop-up they will consent to the use of their data in scientific research. The exact text shown to the users is shown in Figure 1.5.

Once users have granted their consent to use the data and accepted the task, they will be shown the page in figure 1.2. This page contains a short description of the task and a reference to the task description document, which they can read for further information. Once they are prepared to engage in the negotiation task, they can click the "pair me" button to begin the pairing process.

5.2.2 Negotiation Instructions

This link is shown on every page in the site. Clicking this link opens a PDF document walking users through the negotiation process. The document is available as a supplementary document in appendix 2.

5.2.3 Connection Page

Once users acquiesce to join a negotiation, they will be given a temporary user ID and asked to wait until a partner can be found. The webpage is shown in figure 1.3.

While the pairing occurs, the webpage keeps a websocket open so the server can send an alert when a partner is found. This also allows the webpage to ping the server periodically; ensuring that all active users periodically ping the server helps the server determine when a user disconnects and should not be paired with other users. (details in the "Detecting Disconnections" section). For more information on the pairing process, see the "Pairing System" section.

Once a partner is found, the user will be shown the pop-up in figure 1.6. At this point, the user will have 30 seconds to join the negotiation. This time limit ensures that both users are active, even if a user was engaging in other activities while waiting to be paired.

5.2.4 Negotiation Page

Once users have joined the negotiation, they will be redirected to the negotiation page. Initially, the webpage will ask users to wait until their

partner has joined (figure 1.4); this is because the negotiation has a timer that should not begin until both users are present. Once the partner joins, they pop-up can be dismissed so users can view the webpage. Additionally, the negotiation timer will begin.

At the start of the negotiation, each participant will be assigned a role. One will be the buyer, and the other will be the seller. The negotiation pages shown to each participant differ slightly based on the role; each participant has a different task description based on their role, and their goal prices differ slightly. Additionally, the buyer is provided with a submission box to submit an offer for the item. The page for the buyer is shown in figure 5.3.

Figure 5.4 shows the task information section. The top part contains a short summary of the user's role to refer to if they feel any confusion. This section can be hidden by the task description toggle button when it isn't needed. Below the task description is the item description provided by the Craigslist listing, as well as the price on the listing and the user's goal price for the negotiation.

The negotiation interface is shown in figure 5.5. This contains the message panel, an interface for recording and sending audio messages and for replaying previous messages sent and received throughout the negotiation. The website uses the the MediaRecorder API to record audio (see the "MediaRecorder API" section for details). Audio is played back and displayed using HTML audio elements (See the "Audio Element" section for details).

Users are required to take turns in the negotiation. The buyer will send the first message; until the seller receives a message, they will be unable to press the "record" button to record a message. The "send" button is also disabled for both users until a message has been recorded, and disabled when a message is sent. All button enabling and disabling is implemented in JavaScript.

Since users are required to wait when it's their partner's turn to send a message, the webpage provides visual and audial cues to users when their partners are recording and sending messages. When a user presses the "record" button to start recording, their partner will see a "your partner is recording" message in the message panel. When a new message comes in, a "new message" chime is played to catch the user's attention if they are looking elsewhere.

Buyers, in addition to the message panel, will have access to the offer submission interface, which will allow them to send an offer at the end of the negotiation. Finally, the negotiation timer at the top right of the screen informs both participants of the time remaining in the negotiation; when this timer reaches zero, the negotiation is marked as a failure and concluded.

The buyer can use the submission entry to submit an offer. However, the offer cannot be sent until each participant has sent at least two messages each, and the offer value must be an integer. If either requirement is not met, the buyer is prompted to fix the issue (table 5.1). Once both requirements are met, the buyer will be asked to confirm the value they wish to send, since they can only send a single offer (figure 5.6). Once the offer is confirmed, it will be sent to the seller, who will have 90 seconds to accept or reject the offer (figure

5.7).

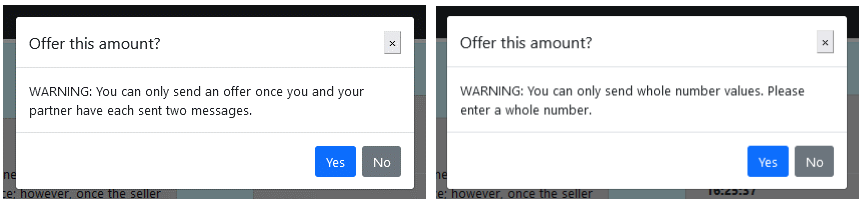


Table 5.1: Error messages shown if they buyer makes an invalid offer.

5.2.5 Negotiation Complete Page

Once the offer has been accepted or rejected, users are sent to a final page to debrief. In this page, they are given a bonus if they reached an agreement (if the offer was accepted), with a larger bonus for getting a final price closer to their goal price. For details on how the bonus is calculated, see the "Bonus Calculation" section.

Mechanical Turk users will also be given a task submission button (shown in figure 5.8). Using this button will use the AMT API to mark the user's task as "complete", as well as log the bonus earned. Should the website be used by a user not affiliated with Amazon Mechanical Turk, it will show them a different page with a "score" instead of a "bonus" (figure 5.9); this allows users to see how well they did in the negotiation without getting paid.

5.2.6 Error Pages

When something breaks the expected process flow, the user will be redirected to an error page. Typically, this occurs when either the user or the partner disconnects.

While short disconnections and refreshes are expected and handled (see the "Detecting Disconnects" section for details), it is possible for a user to leave the conversation entirely (due to technical issues, poor internet, or boredom). If a user disconnects for too long and attempts to reconnect, they will be informed that they have timed out and asked to start again from the beginning (figure 1.7).

If a user's partner disconnects and times out, this is handled one of two ways. If the disconnection occurs after the negotiation is already underway, the user will be paid despite not having completed a negotiation, to avoid punishing users for their partner's disconnections. If the negotiation is not underway, the user will not be paid. In either case, the user will be offered a chance to restart the negotiation with a new partner (figure 1.8).

5.3 Communication Layer

5.3.1 Websockets

Communications between the client-side (the website front-end) and the server-side happen almost exclusively through websockets. This includes ping messages to prevent user timeouts while waiting, voice messages and offers

sent, etc. Both the client-side and server-side use basic websocket libraries to handle websockets.

The server-side also has a websocket connection manager to handle websocket connections for paired users. The connection manager tracks which user is associated with which websocket and which users are paired with each other. The connection manager is used to forward messages through the server to their partner.

5.4 Server-Side Layer

5.4.1 Cookies

Cookies are used by web servers to maintain state across time for the same user. If a user refreshes a webpage or goes to another page on the same website, the only way to maintain information across the change is to use cookies saved by the client web browser.

The server used in this project uses cookies to store each user's negotiation ID and permanent ID. This is used, for example, to know which negotiation a user is in when they reach the negotiation page and determine who they are partnered with. This is critical for obvious reasons.

5.4.2 Data Systems

5.4.2.1 The User Database

Data about the participants in each negotiation is stored in a PostgreSQL database. Each entry in the database stores several datapoints about

the user; their IDs and the IDs of their conversation partners, the item information used in the negotiation, their role in the negotiation, the negotiation ID, their goal and their partner's goal, the number of messages sent and the average message length, the final offer, whether it was accepted, and whether the negotiation ended in a disconnection. A visual representation of a database element is shown in figure 5.10.

This information is updated throughout the negotiation and is used to aggregate data regarding each negotiation for further analysis. Additionally, it can be used to track conversation state to help with user reconnections and refreshes (see Detecting Disconnects for details).

5.4.2.2 Data Structures

The server also uses several data structures to store information about the negotiations and users. This is useful for caching information in a more accessible manner than keeping everything in a PostgreSQL database. These data structures can also store information relevant during negotiations that does not need to be stored in a database.

Message objects track the user ID of the sender, the time at which they were sent, and the name of the .mp3 file in which they are stored (see figure 5.2). These are used when collating the messages in each negotiation in the order they were sent, which can be useful for data analysis.

User objects (figure 5.2) contain the user and their partner's IDs, the negotiation's ID, and the user's role. They also store the start time of the

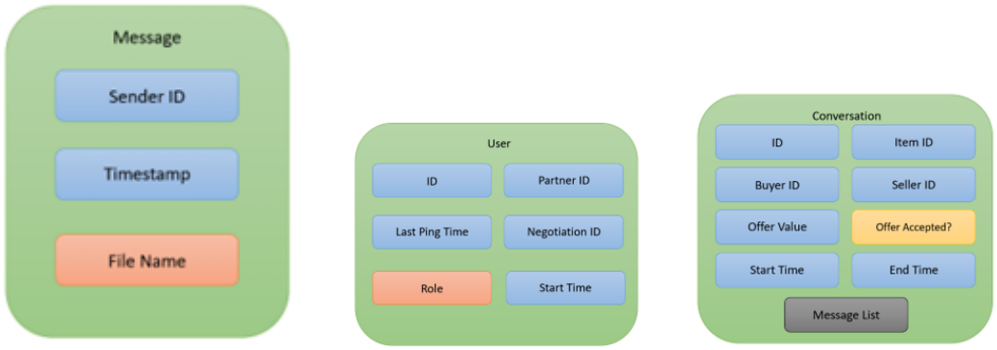


Table 5.2: Designs of the data structures used by the server.

conversation and the time since the last ping from that user, which are used for handling disconnections and refreshes (See "Detecting Disconnects").

Conversation objects (figure 5.2) track their ID, the item used in the negotiation, the buyer and seller, offer information, a list of all messages sent (in the order in which they were sent), and the conversation start and end times. The data stored in this object is printed to a file at the end of the conversation to facilitate data analysis.

5.4.3 Processes

5.4.3.1 Pairing System

When a user sends a connection request to the server, the server puts the user in a list of unpaired users. It then periodically checks for another unpaired user in the list. If such a user is found, it marks both users as partners and removes them from the list.

It is possible for the pairing system to time out if no partner can be

found. In this case, the unpaired user is asked to try again later.

Once two users are paired, a conversation object is created. Each conversation chooses an item for the users to negotiate over. Each user is then randomly assigned to be the buyer or the seller. They are also assigned pseudo-random goal prices; the buyer is given a goal price between 60-75% of the listing price, while the seller is given a goal price between 80-95% of the listing price. When the users go to the negotiation page, the conversation object is used to populate the page with the necessary item and negotiation information.

5.4.3.2 Handling Communications

The server and the client browser communicate using websockets. However, the browser and server must send several different types of messages; users will send voice recordings, offers, offer responses, and ping messages in a non-predetermined order. It is difficult to manage a separate websocket connection for each type of data for the same browser tab; as such, the server maintains a single websocket connection for each user.

To differentiate between different types of messages, all messages are sent as byte arrays and treated as FILO data structures with an identifying tag pushed at the end. This tag is used to determine the type of data stored in the data structure. Once the data is handled, the original message is forwarded by the server to the sender's partner (unless the original message was a check-in ping).

The different types of messages are handled as follows. Offers, which

are integers, and offer responses, which are Boolean values, are cast to the correct type and saved in the user and conversation data structures. Check-in pings are simply ignored, but the user is marked as "active".

Voice messages, which are sent as byte arrays and are meant to be read as .ogg files, are converted by FFmpeg into the .mp3 format and saved as files on the local filesystem. They are also fed into Whisper for speech-to-text generation. The conversation data structure is then updated to include the new message's information, including the location of the .mp3 file and the text generated by Whisper.

5.4.3.3 Detecting Disconnects

Users can disconnect for a variety of reasons. They may have a spotty internet connection, their device could lose power, or they could simply need to leave for personal reasons. Regardless of the reason, the negotiation server needs to be able to identify and handle these cases.

The server uses a ping system to identify if a user is no longer active. Each client-side webpage, as long as it is open, will periodically send pings to the server. The server will track which user the ping came from and will update that user's "last ping time" accordingly.

However, if the server does not receive a ping from a user within a certain timeout (which is currently set to 120 seconds), it will assume that user is no longer active. The server checks each user when they send a ping – if a user disconnects for over two minutes and attempts to reconnect, it will

mark them as having timed out and they will need to find another partner. The server also checks all users in the system every minute, to periodically prune all timed-out users.

It is possible for users to keep the webpages open, but still be inactive. This could be a problem if they have already been paired with another user, who needs them to be responsive; in these cases, the webpages they are on have internal timers that will report a disconnection if they run out (see the "Connection Page" and "Negotiation Page" sections for details).

It is also possible for a user to disconnect for a short period of time. In this case, it is preferred to allow them to stay in the negotiation rather than report a disconnect. For this reason, the server maintains each user's conversation state (using data structures and the user database) from the moment they are paired with another user. As explained in the "Disconnection System (User Perspective)" section, if they refresh an active negotiation page, they will be provided all sent messages. If it is their turn to speak, the recording button will also be enabled. The timer will also be updated using the "negotiation start time" provided by the user when they first connected.

Certain systems, like Amazon Mechanical Turk, handle refreshes differently. These systems connect to the DARE system's homepage when they connect, so if a user wishes to reconnect to their negotiation, the system must check the user's state at every step of the negotiation process to ensure that, if they have already been paired, they can quickly return to the negotiation and pick up where they left off. As such, the website is refresh-capable from

the home page to the conversation completion page.

5.5 Summary

In this chapter, we examined the system design that serves webpages, handles errors and ensures users have as smooth an experience as possible. We discussed how each webpage in the frontend controls the user process flow, how websockets are used to facilitate communication between users and the server, and how the server stores data and handles communications and disconnections.

In the next chapter, we will discuss the performance of the DARE interface as a data collection tool. We will discuss a pilot data collection study and examine the data collected.

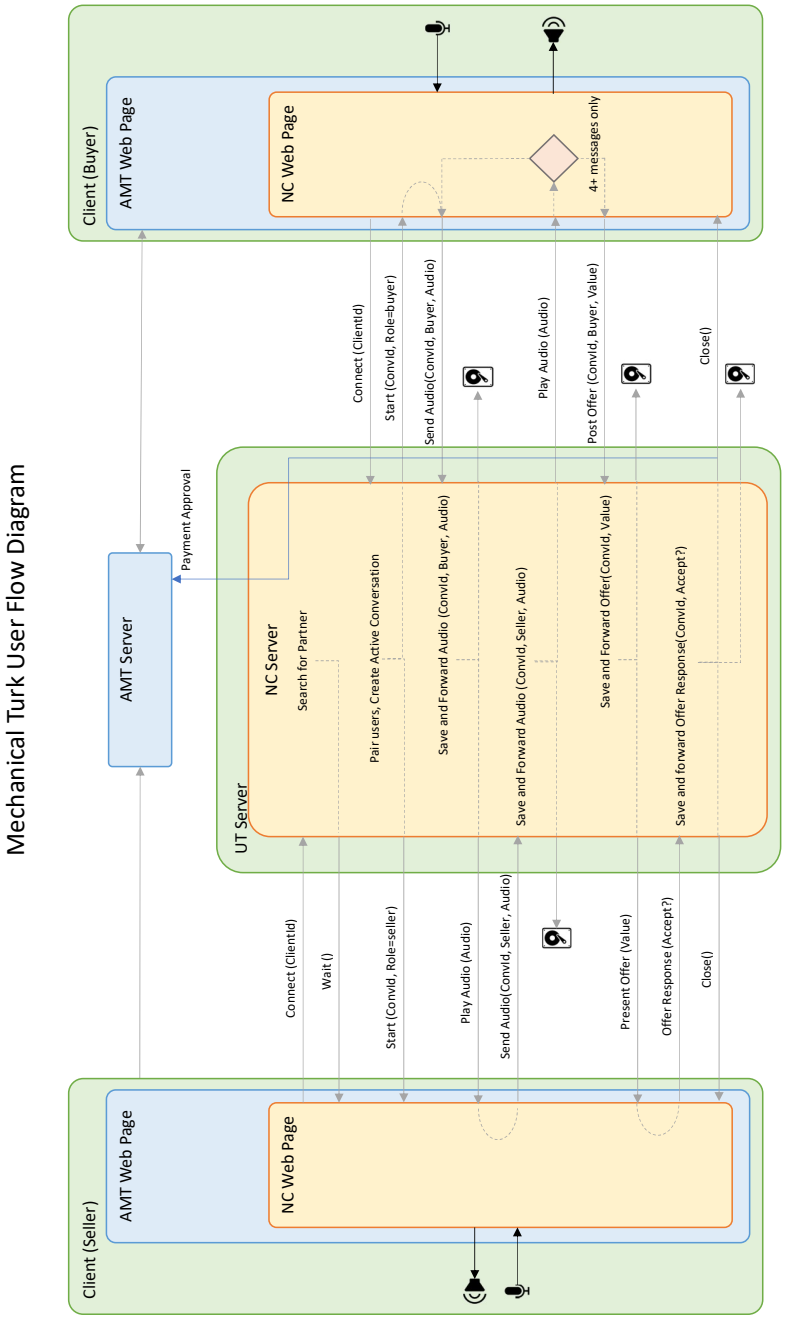


Figure 5.1: A diagram showing the typical user flow of a participant using Amazon Mechanical Turk.

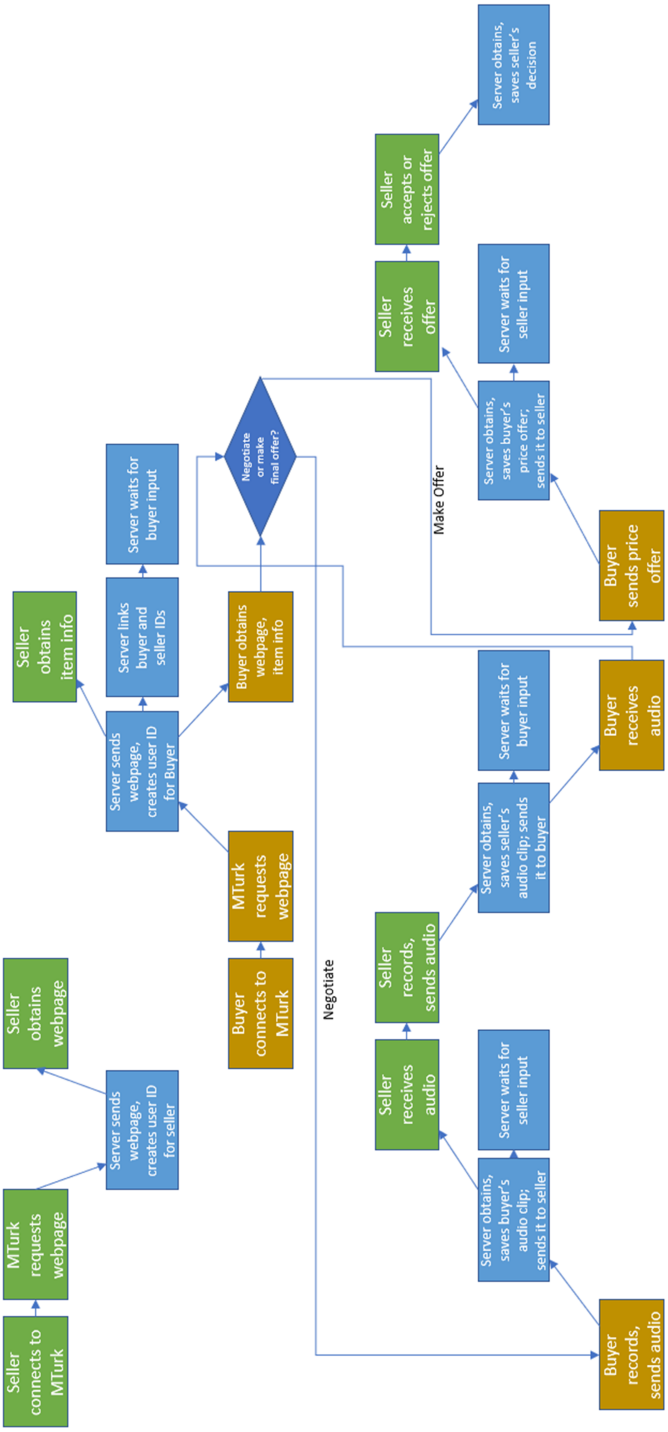


Figure 5.2: A flowchart showing the process flow of two users engaging in a negotiation.

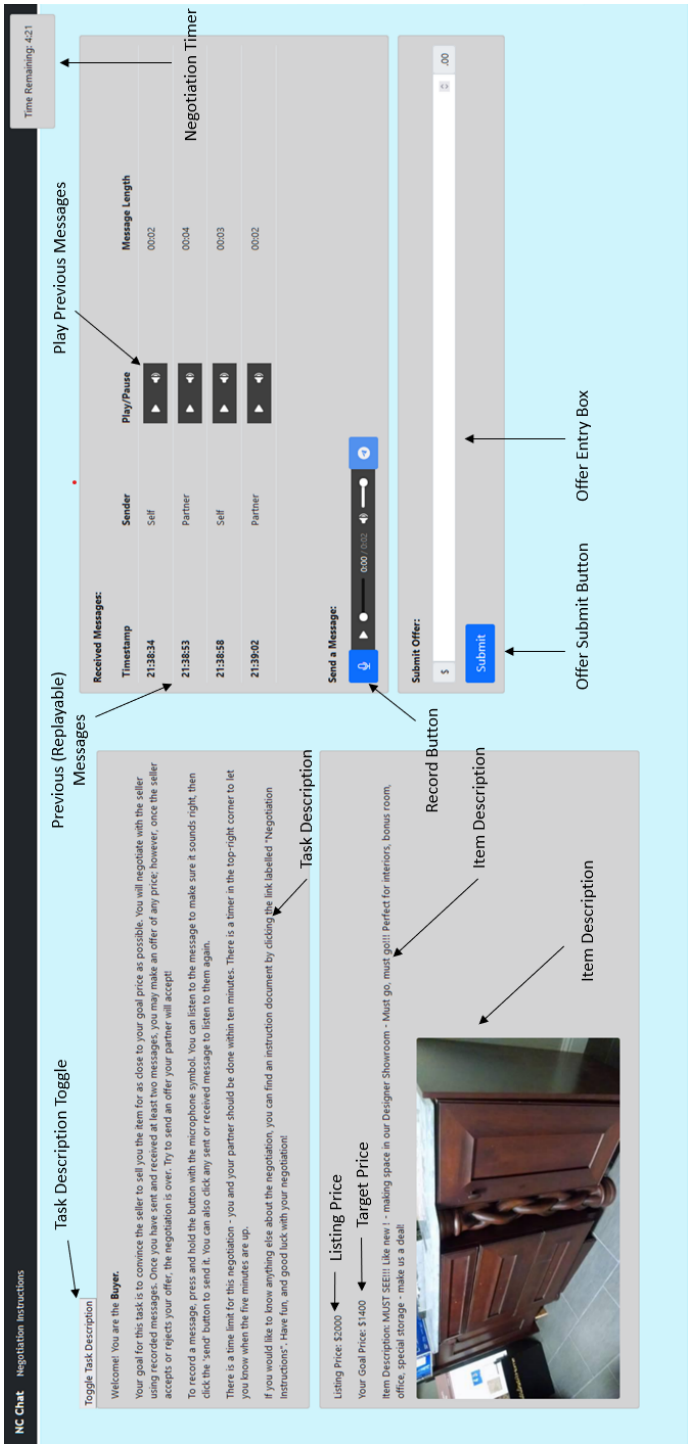


Figure 5.3: This is the negotiation interface in its entirety.

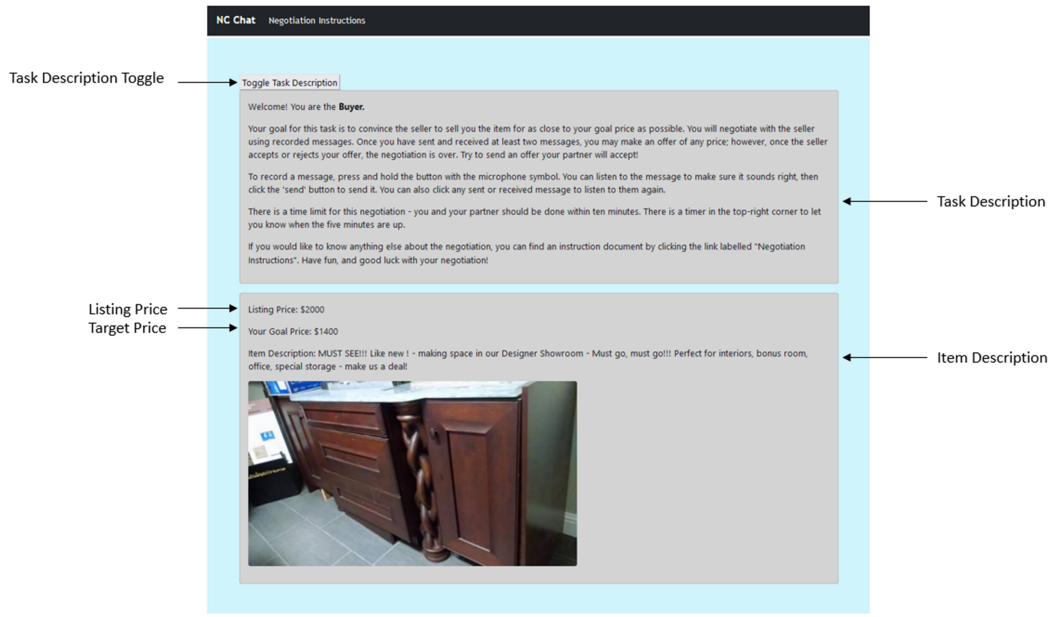


Figure 5.4: The task information provided to the user, including information about the item for sale and the goal price.

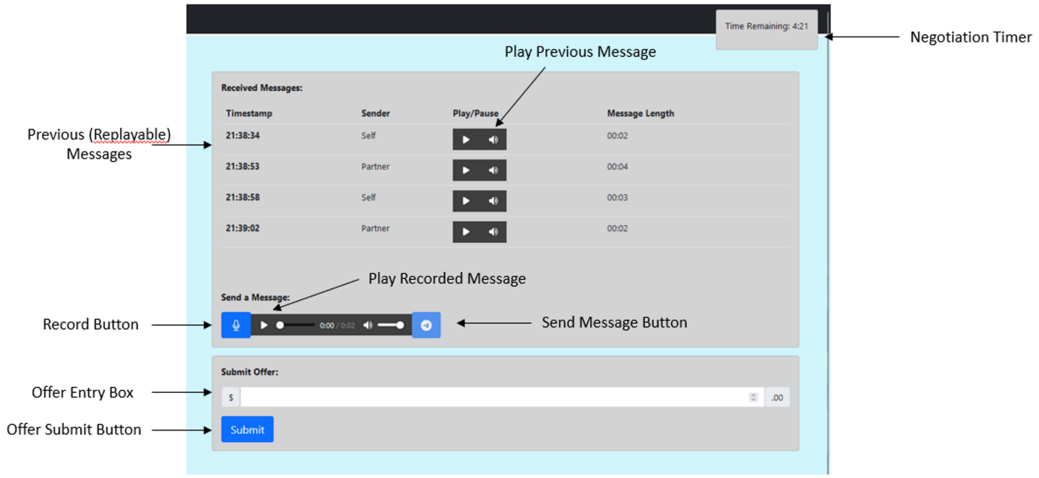


Figure 5.5: The section of the webpage used for communication, including sending and listening to audio messages. Includes the buyer's offer panel.

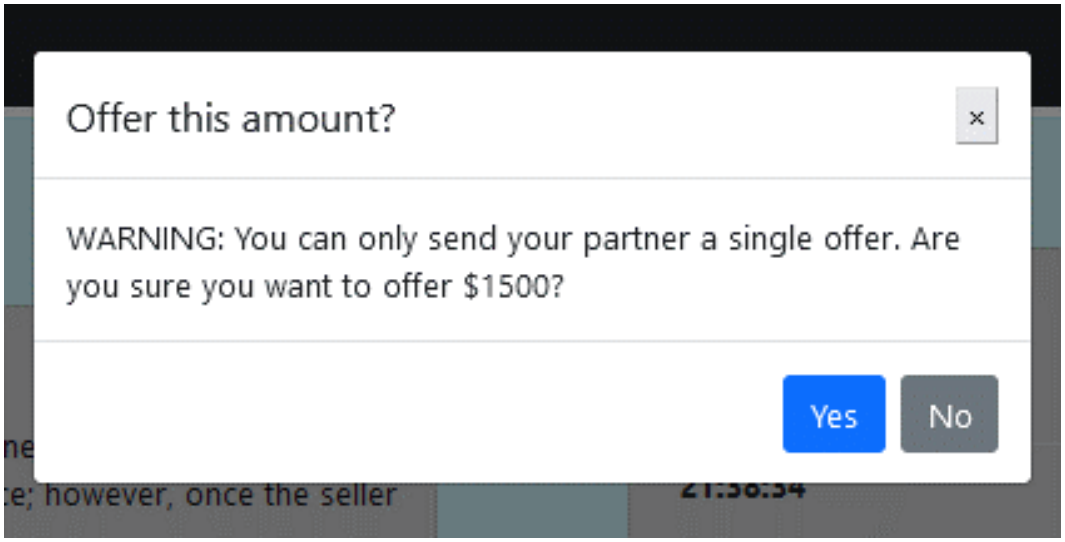


Figure 5.6: Shown when the buyer makes a valid offer.

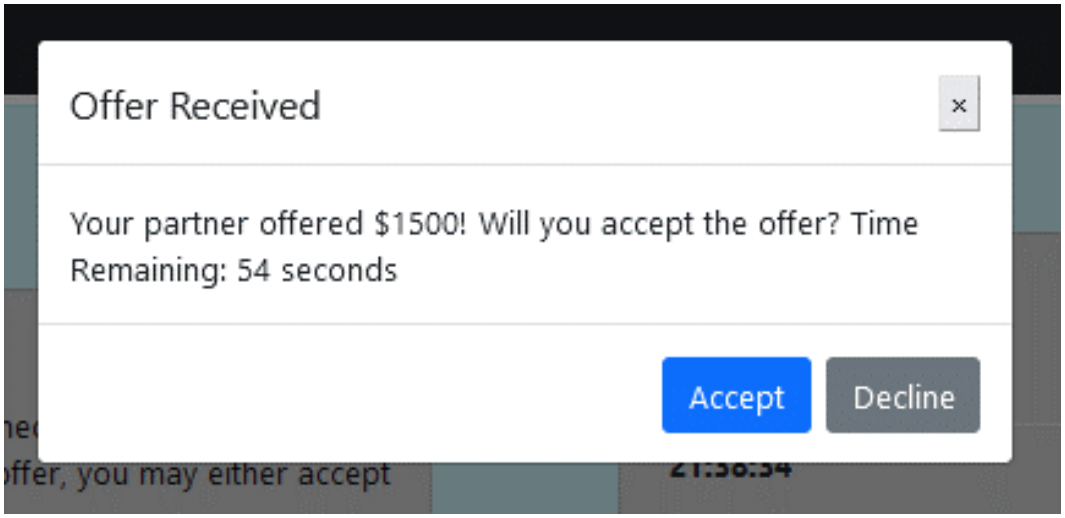


Figure 5.7: Shown when the seller receives an offer.

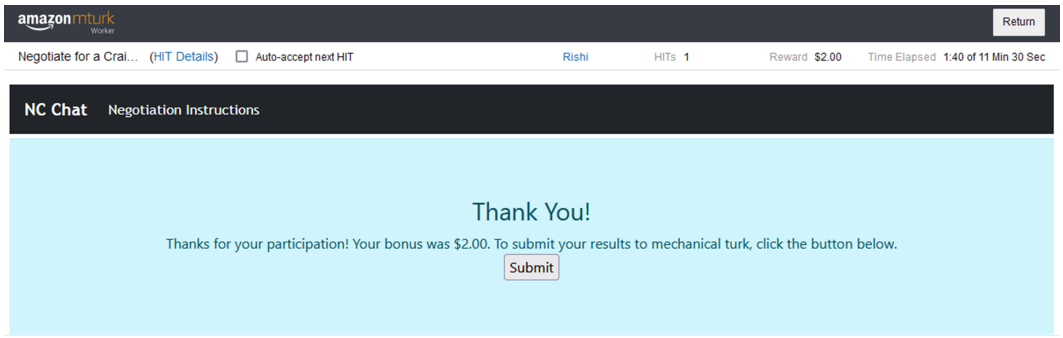


Figure 5.8: The submission page shown to Mturk users.

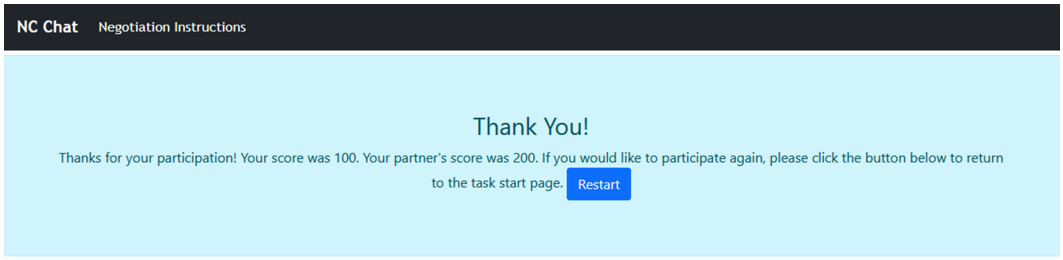


Figure 5.9: The submission page shown to other users.

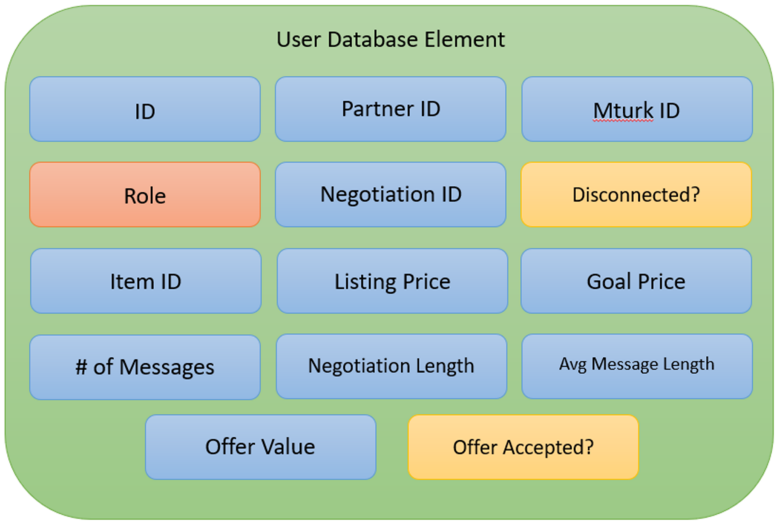


Figure 5.10: An example element from the user database.

Chapter 6

Data Analysis

Initial tests of the DARE interface have shown that the interface is user-friendly and that users are able to successfully engage in dialogue on the platform. The DARE website was set up for use in negotiations over Craigslist Listings, and multiple negotiations were recorded.

6.1 CraigsListBargain Dataset

Participants in negotiations need something to negotiate over. We used Craigslist listings from the CraigslistBargain dataset [10] to provide items and prices for participants to bargain with.

CraigslistBargain is a dataset of negotiations over items scraped from . The dataset includes all the items, as well as the goal prices for participants, the messages sent, and the results of the negotiation. For the purposes of this project, another dataset was created by scraping the items from this dataset; in addition, the dataset was filtered to remove all items without associated images (to ensure participants would be given an image to negotiate over).

The resulting dataset, which contained items, images, descriptions, and prices, was used to provide item data for all negotiations performed on the

platform.

6.2 Pilot Data Collection

An initial study was conducted by a group of graduate students at UT-Austin. This group consisted of 8 graduate students who performed negotiations over a period of two hours. The negotiations were performed concurrently, but with each participant in separate locations; all participants communicated solely through the platform while engaging in dialogue. The participants connected directly to the DARE website, rather than using a paid platform such as Amazon Mechanical Turk.

The statistics of the collected data are shown in figure 6.1.

6.3 Automatic Speech Recognition (ASR)

To facilitate data collection, the negotiations were processed using Whisper to create transcripts. These transcripts were then used to determine the numbers of tokens and vocabulary size. This allowed for the collection of data points that could be directly compared to the statistics presented in He et al.

The specific Whisper model used for transcription was the 'medium.en" model, which is fine-tuned for english speech. This model had fairly good accuracy - out of 20 recordings chosen at random, the transcriptions appeared completely accurate for 18 of them. In the remaining two, the following errors

	DARE	CraigslistBargain
Number of Dialogues	15	6682
Average Number of Messages	6.13	9.2
Average Tokens per Message	32.5	15.5
Average Utterance Length (s)	13.53	N/A
Vocab Size	669	13928
Vocab Size Excl. Numbers	612	2623

Figure 6.1: Dataset Statistics

were made:

"I got it off Craigslist" → "I got it of Gregor's"

"I did put in some tune-up work" → "I did put in some tuna pork"

In all, the sampled accuracy of 90% was sufficiently high for the sake of this study. For future works, minor assistance from a human annotator (or other transcription models) can be used.

6.4 Data Analysis

The statistics of the data collected from the pilot study are compared to the equivalent statistics from the paper by He et al. While the number of messages sent is fewer, there were more tokens per message.

Additionally, many of the phenomena identified by He et al. are also present in the data collected in this study. The phenomena, as described in He et al., are shown in figure 5.1. Examples of the phenomena found in the

collected data are shown in figure 6.2. Clearly, participants in negotiations use similar techniques whether they are speaking or typing.

Phenomenon (number seen)	Example
Embellishment (4)	Yes, it most definitely does [have books on it]. **I swear it has half my library on there.**
Cheap Talk (8)	Yeah, **this was the property I leased while I was in college**. It's newly refurbished a year before I left. It's in great condition(...)
Side Offers (4)	Now I completely understand in the sense that you want to at least break even, right? So here, I mean, **if you can throw in some of those extra pillows that you have on it**, I'm more than willing to go up to like, how about we say like 110?
Appeal to Sympathy (1)	Maybe, but **I'm also trying to buy a car for my daughter when she goes to college and she already has student loans.** I can't really pay that much for them, so we are kind of looking for something lower in the price range.
World Knowledge (5)	Did you really get that for $1000? Because **I've been seeing bikes in the store that are like $500 for about that size.**

Figure 6.2: Table of Negotiation Phenomena

Transcripts of example negotiations, along with the item details for the negotiations and negotiation statistics, can be seen in figures 3.1, 3.3, 3.5, and 3.7. All the example negotiations are shown alongside the item used in the negotiation, the goal prices for both participants, and the price that was eventually agreed upon. Additionally, for comparison, the transcripts of the negotiations can be seen alongside transcripts of negotiations for the same

items in the CraigslistBargain dataset in tables 3.2, 3.4, 3.6, and 3.8.

6.5 Summary

In this chapter, we discussed the pilot study used to gather audio data. We also compared the collected data to the results of a text-based negotiation study performed by He et al. In the next chapter, we will discuss the ongoing integration of the DARE interface with Amazon Mechanical Turk, discuss other applications of the DARE interface, and consider potential future work that can be done to expand its applications.

Chapter 7

Ongoing and Future Works

7.1 Alternate Dialogue Tasks

7.1.1 Basic Dialogue Tasks

This platform is currently built to facilitate and record spoken negotiations. However, the underlying communication capabilities and the easy integration with Amazon Mechanical Turk can be used for many other tasks. DARE can be used to record casual conversations, audio-based food purchasing, human interaction when playing online games, etc. DARE can be used, with no change to the functionality, to implement any two-user communication task that requires text and/or image displays to the user. This includes tasks such as food ordering, casual conversation, instruction-giving, joke-telling, and myriad others. This would simply require switching out the CraigslistBargain dataset with the required text and images, and changing the instructions to the user. Everything else would be identical.

7.1.2 Multi-User and Complex Tasks

Additionally, with minimal changes, DARE can be adapted to allow users to perform more complex tasks. For an example of such a task, consider the game Diplomacy. Language and strategy models have already been used

to create Cicero, an AI agent able to mimic human-level play using text-based communication [7]. Cicero would likely struggle to replicate this feat in a game where humans used voice chat to communicate. However, a model trained on audio data of humans playing diplomacy may be able to achieve that feat.

Adapting DARE to allow users to play diplomacy would require minimal changes. Users would need to be given multiple partners, which the pairing system can easily do; users would need to be able to select the recipient for each message, which requires button additions in the UI and can easily be handled in the server's websocket manager; and the UI would need to show the game state of Diplomacy. Of these changes, only the last would be difficult to implement, and has likely already been implemented for online gaming purposes.

7.2 Amazon Mechanical Turk Functionality

Amazon Mechanical Turk provides an API for programmatically creating tasks and paying workers[23]. Python scripts can be used to create a variable number of tasks with chosen payment values, time limits, worker requirements, etc. Once users complete the task, an API call can be used along with the worker's ID and the task ID to mark the task as complete and request payment. Workers can also be paid automatically using the API.

The API can be used to automatically generate tasks that lead the user to an external site using the HTML iframe element. Iframes use a link to another website to show the website within a div element; the website within

the div will work as it would in a normal browser tab. Once the task was complete, the API can also be used to submit the task to Amazon Mechanical Turk.

Ideally, DARE will be integrated with Amazon Mechanical Turk for easy access to dialogue participants and to pay participants for their data. A script has already been written to generate tasks on the Amazon Mechanical Turk platform that show the DARE website in an iframe. Further testing and fine-tuning is necessary to complete the Mturk Integration.

7.3 Additional Functionality

The DARE interface is currently compatible with the most common desktop browsers. However, there are compatibility issues with some mobile device browsers. The MediaRecorder API does not appear to be compatible with certain devices, particularly iOS handhelds. Future work may include finding libraries to add compatibility with these devices.

Chapter 8

Conclusion

This paper has covered the DARE interface in great detail. We examined the reasons why collecting audio recordings of dialogue would be useful, and why a new tool was necessary. We examined previous collected datasets and data collection platforms, as well as the types of tools in place to facilitate the use of the most popular platform, Amazon Mechanical Turk.

We discussed the requirements of the platform, including simplicity of the user interface, reasonably high performance and load tolerance, and ease of use for future developers.

We overviewed the software, libraries, webtools, and languages used to develop DARE. This includes the framework used to code the server and the server manager, the audio processing tools and long-term data storage options, the client-side webtools, the audio recording APIs, and websockets. We also discussed the AMT API and the dataset of Craigslist listings from CraigslistBargain.

We examined the system components, both from the user's perspective and from the developer's perspective. We examined the workflow of a user and discussed how the webpage and server were developed to help guide the

user through this workflow with as little friction as possible.

Finally, we analyzed data collected in the initial tests of the DARE interface. We found that many of the features found in the CraigslistBargain dataset are also present in the sample of data collected.

The DARE interface will make it much easier to collect audio dialogue data. As of now, it can collect audio data for negotiations. However, since much of the framework can be easily repurposed for other dialogue types, it can easily be updated by other developers to collect other types of dialogue datasets. It is my hope that this tool can be used by others and updated by other developers to collect a broader range of data, and thusly broaden the range of helpful models and AI agents in new fields.

Appendix

Appendix 1

Additional Figures and Diagrams

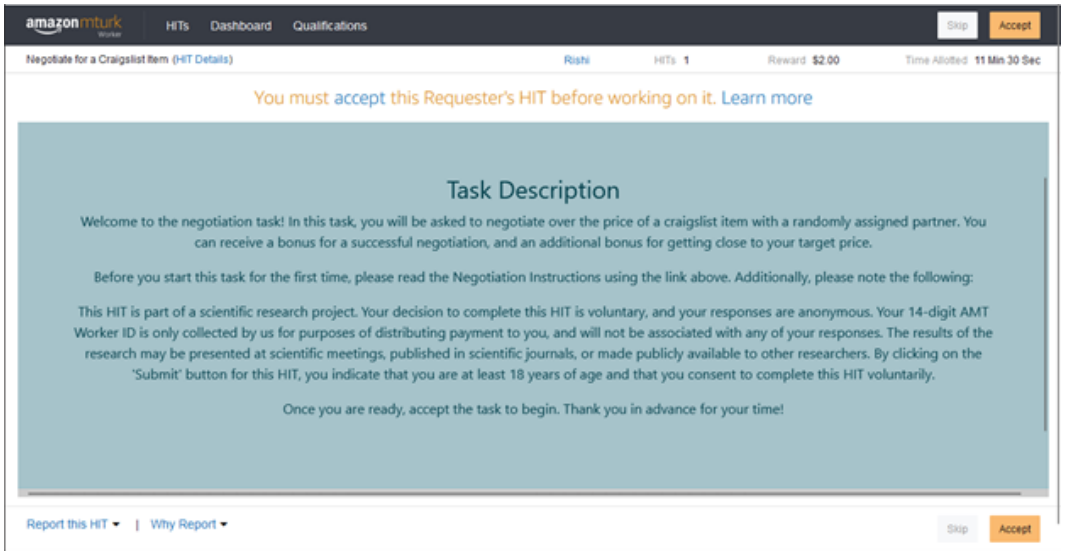


Figure 1.1: The task description page as seen by a worker on Amazon Mechanical Turk.

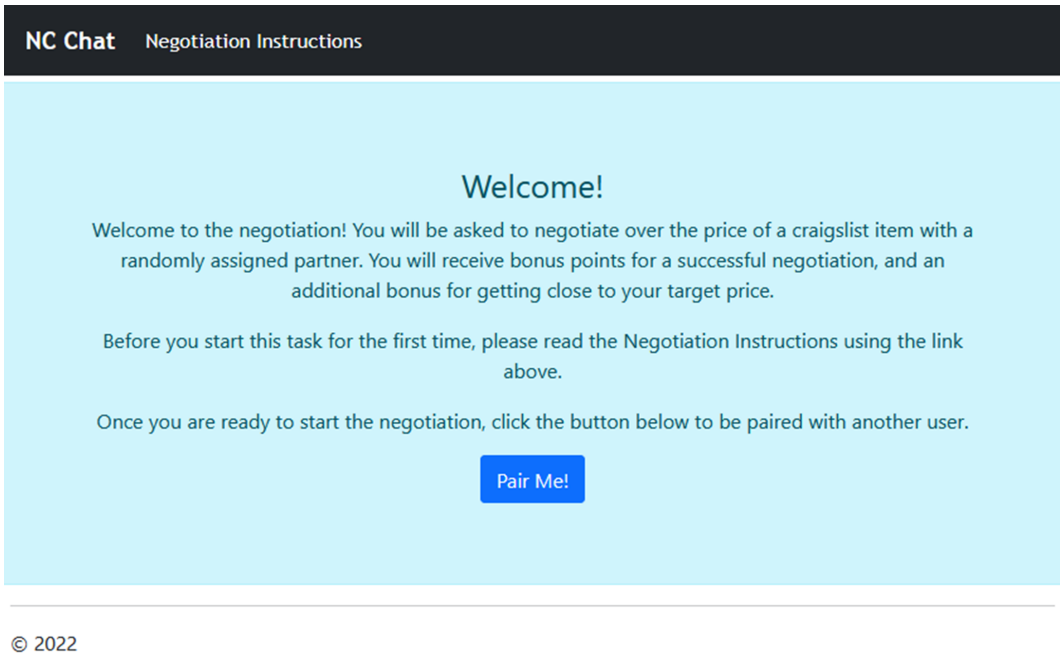


Figure 1.2: The homepage, as shown to users who have consented to providing their data.

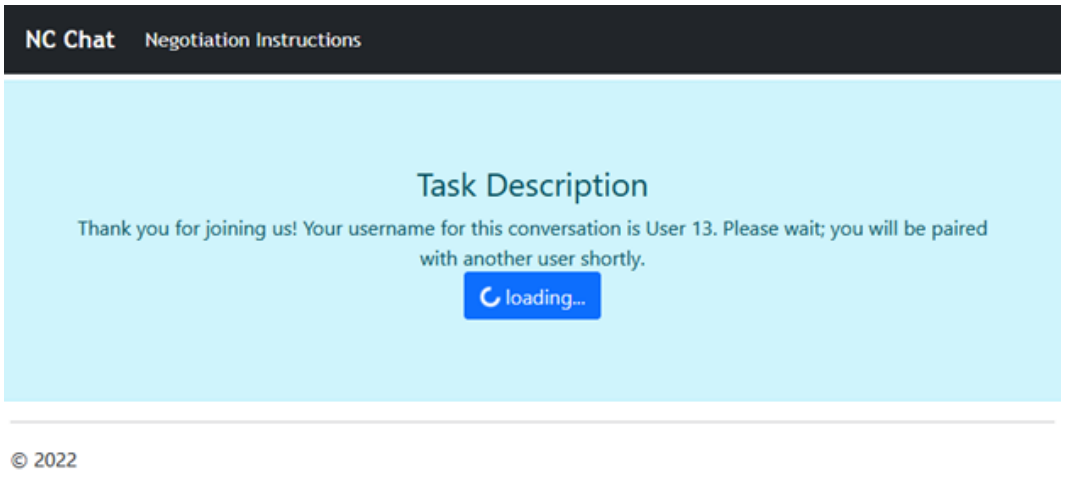


Figure 1.3: The connection page shown while waiting for a partner to be found.

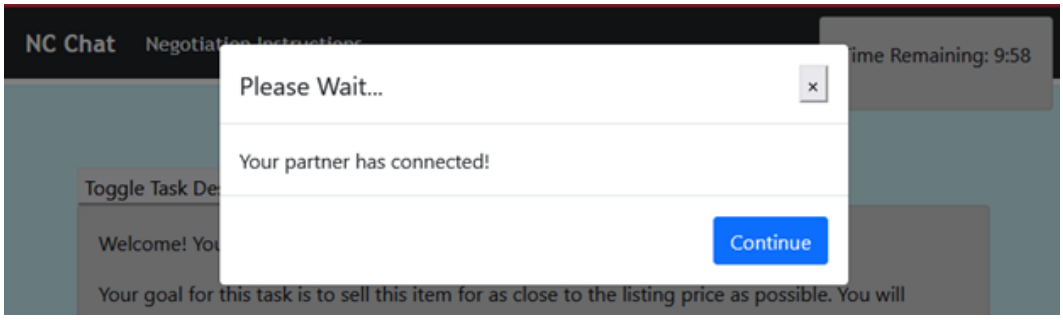


Figure 1.4: This is shown when both users are prepared to negotiate.

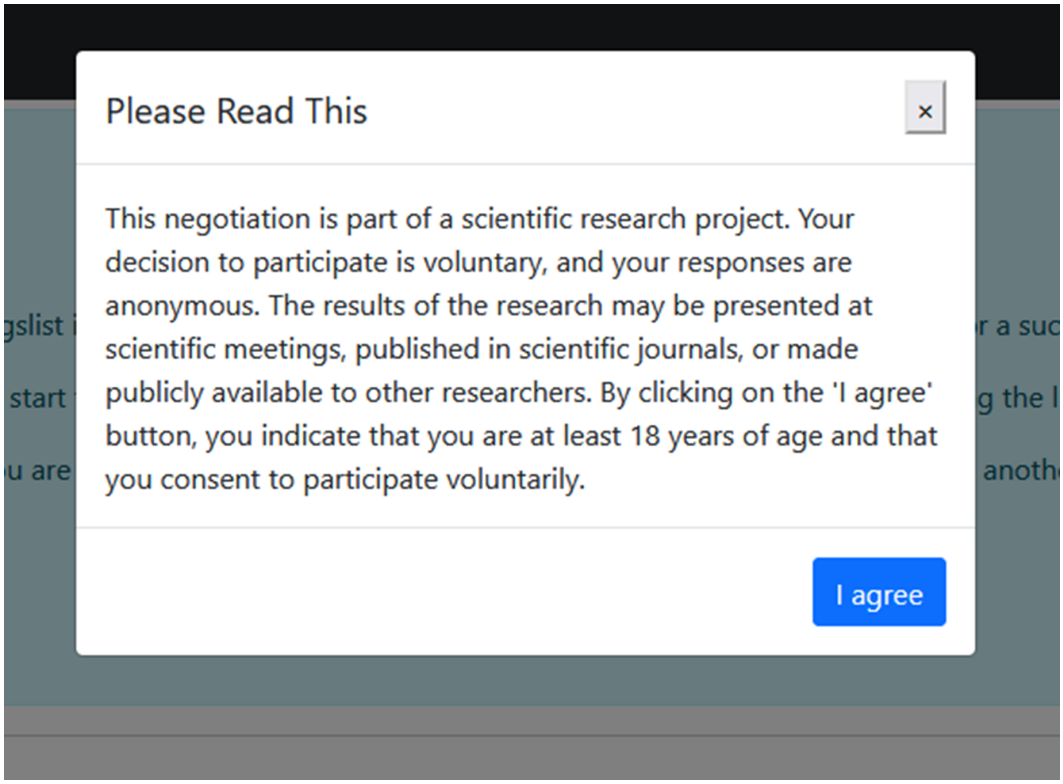


Figure 1.5: A consent form shown to users not using Amazon Mechanical Turk.

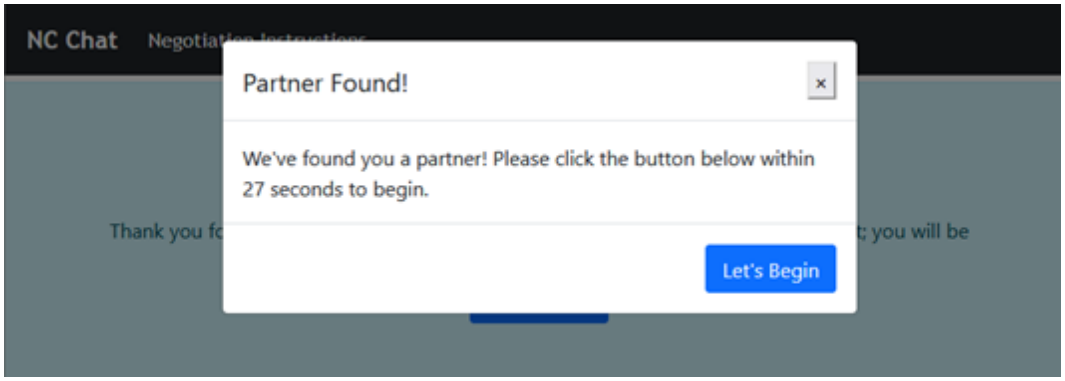


Figure 1.6: The screen shown when a partner is found.

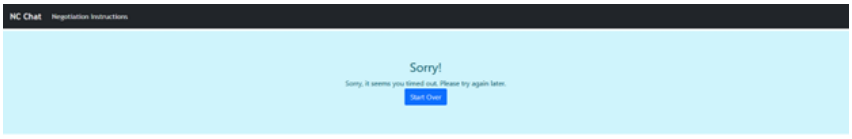


Figure 1.7: The error page shown when a user times out.

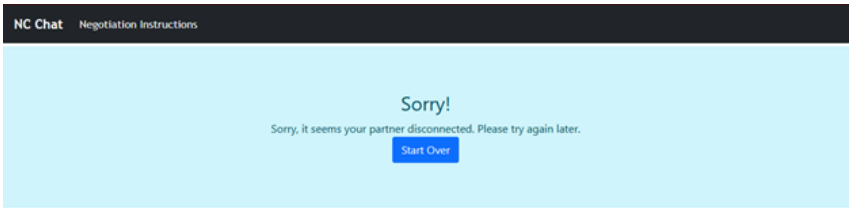


Figure 1.8: The error page shown when the user's partner times out.

Appendix 2

Negotiation Instructions Booklet

Negotiation Chat Instructions

Table of Contents

Overview

You will be negotiating on our negotiation platform. The platform will pair you with an anonymous partner. Once you are paired, you will be provided a listing for an item for sale. You will be asked to negotiate with your partner by sending voice-recording messages back and forth until you agree on a price; if you agree on a price within the time limit, you will both receive a bonus.

There will be three stages to this task:

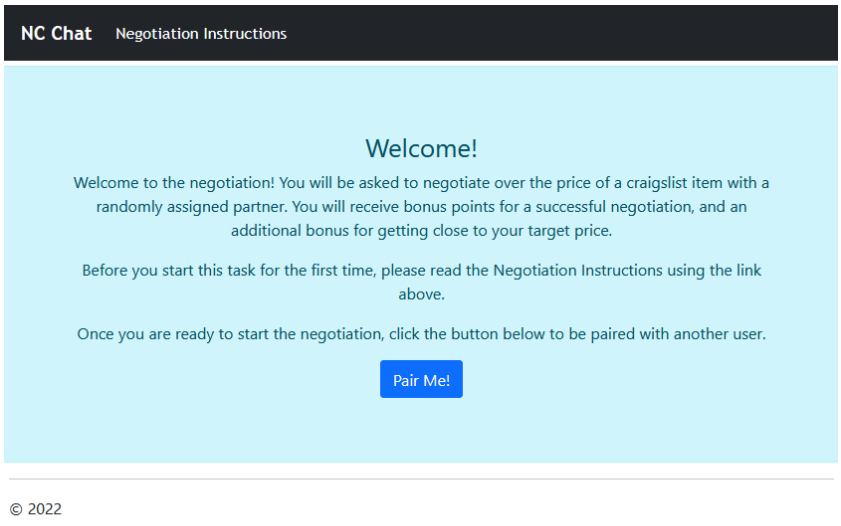
- Pairing with another user
- Negotiation through voice messages
- Making an offer / Ending the conversation

Goal

Your goal is to succeed in the negotiation, reaching a price that is closer to your target price and reaching an agreement with your partner. You and your partner will only be rewarded if there are at least 3 voice messages per person. It is okay if you do not reach an agreement, however, each partner will receive $0.50 bonus compensation if you agree on a price. You can also receive additional compensation up to $1 per conversation; this additional compensation will depend on how close you are to your target price. You should not discuss your target price – consider that, in a real-life negotiation, you are unlikely to expose your goal to the other person!

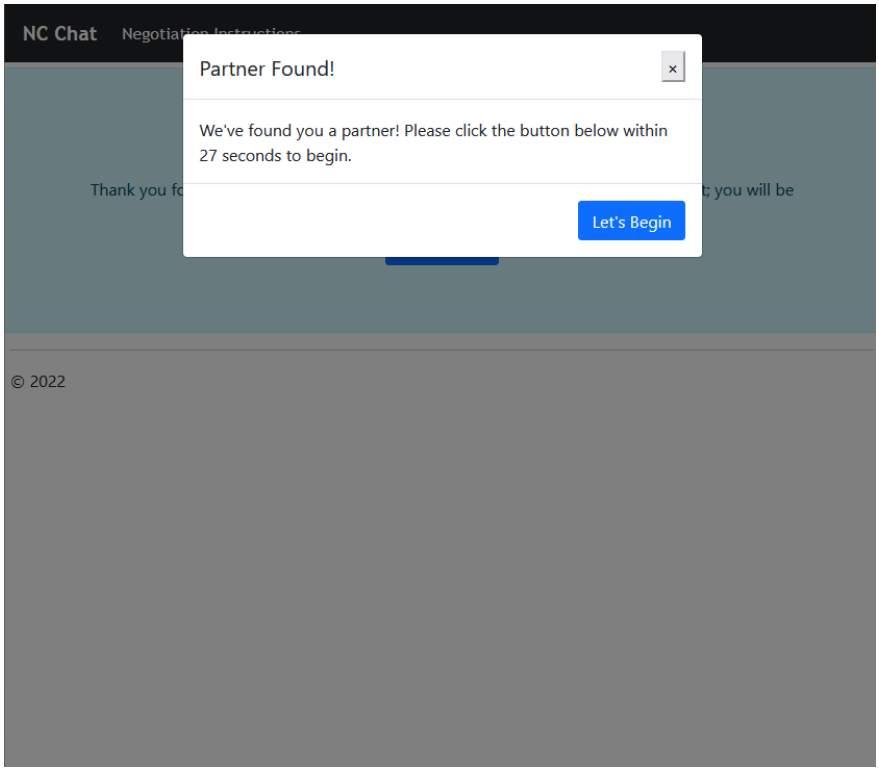
Pairing System

When you first reach the website, you will see the following page.

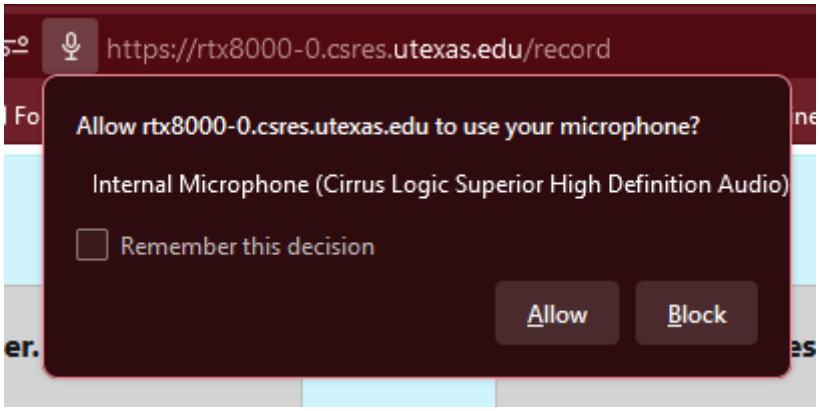


To begin the task, click the button that says “pair me!”. This will provide you with a user ID and automatically start searching for a partner. Once you are paired, **you will have 30 seconds to complete the connection and move to the next phase.** Make sure you keep an eye on it!

The following popup will appear when the connection occurs.



Click the “Let’s Begin” button to go to the negotiation page. When viewing this page for the first time, it will request permission to use your speaker and microphone:



Allow the website to use your speaker and microphone, and you’re ready to go!

Chat page overview

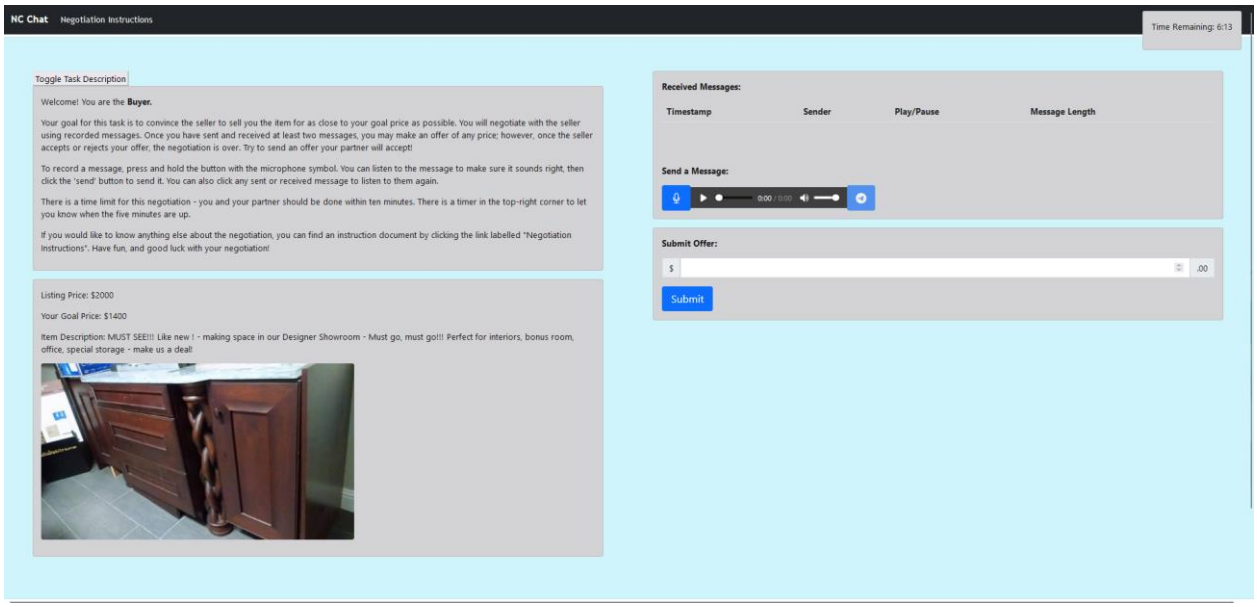


Figure 1: Buyer Chat Page

Depending on your role in the negotiation, you will have a slightly different chat page. The chat page will have two sections; the information panel and the message panel. The page shown above is the buyer's page; if you are the seller, you will have the same page without an offer submission section.

Information Panel

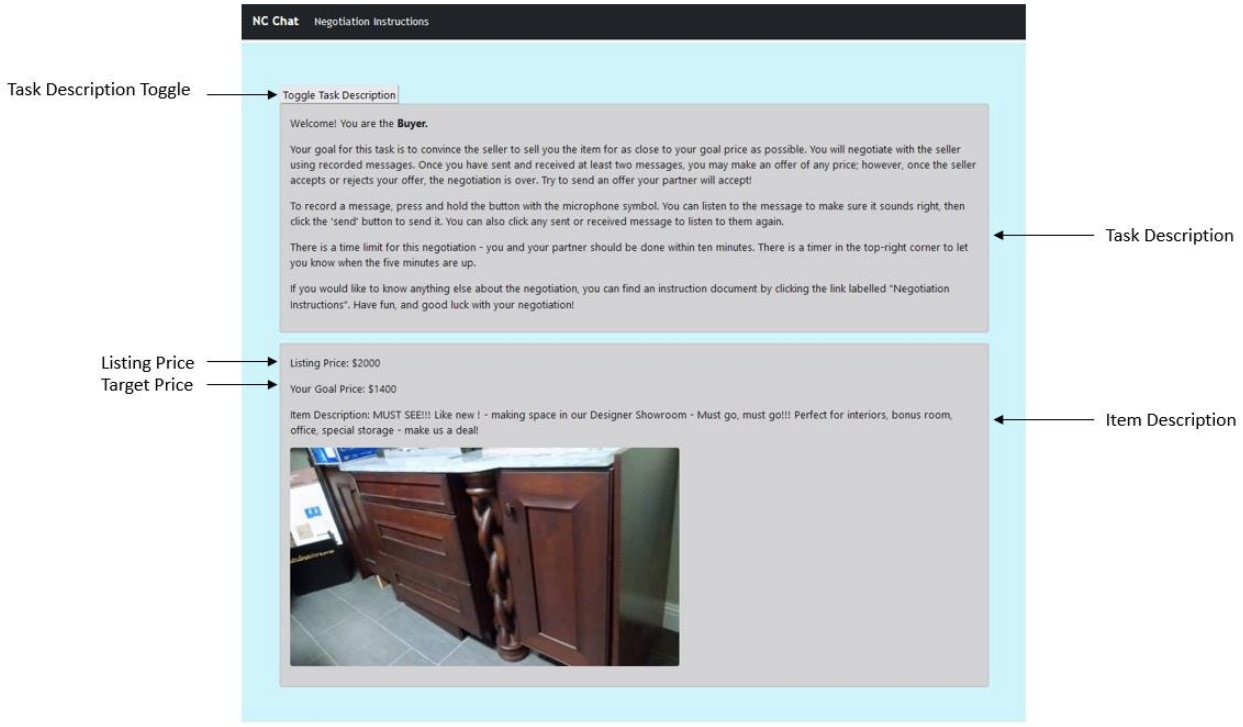


Figure 2: Information Panel

Task Description

Regardless of your role, you will be provided a short task description. Refer to this if you have difficulty remembering what you need to do during the negotiation.

Item Description

This is the listing of the item you and your partner are negotiating over. The listings vary in informativeness, so feel free to make assumptions about the item as necessary based on the provided image.

Target Price

This is the price you are aiming for in the negotiation. The closer you can get to this price, the higher the bonus you receive. That said, remember that you get a bonus for reaching an agreement, so don't be afraid to make compromises!

Listing Price

This is the price shown on the Craigslist listing. This should give you a starting point for the price of the item.

Message Panel

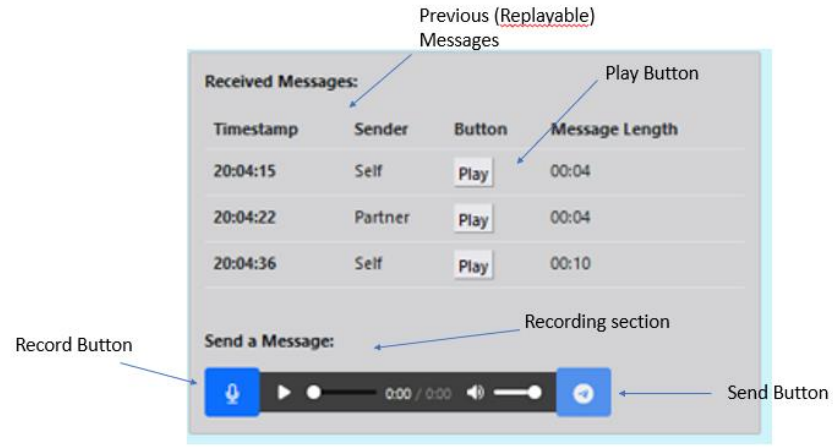


Figure 3: Message Panel

Recording Messages

This part of the panel is where you record messages you want to send. To record a message, right-click and hold down on the **record button**, say whatever you would like to say, and release the button when you are finished. **Don't release the click until you have finished recording.** Note that the record button will be green while recording is in progress, and that if it changes back to blue, it has stopped recording. If you want to re-record or change your message, you can record again; this will overwrite any existing recording.

Once you are satisfied with your message, click the **send button** to send the message.

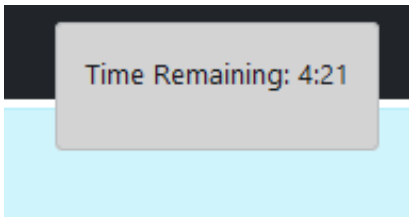
Replaying Messages

You can replay any message you have sent or received during the negotiation. Messages are displayed in the order they were sent in the **received messages** box. To replay a message, click the **play button** for that message.

Offer (Buyer only)

If you are the buyer, you may send an offer when you and your partner are done negotiating. To do so, enter your offer value in the box labelled with "submit offer". There are more details in the task description below.

Other



Time Limit

The negotiation has a time limit, which is shown in a countdown timer in the top right corner. You and your partner will want to reach an agreement before the time runs out, or the negotiation will be marked as a failure.

Task (Buyer)

Send the First Message

As the buyer, you will be reaching out to the seller with the first message. This is your chance to introduce yourself, ask for more information, and talk about your budget. Remember, you can re-record your message as many times as you like, but you can only send one message at a time! Once it's sent, you cannot send anything else until you receive a response.

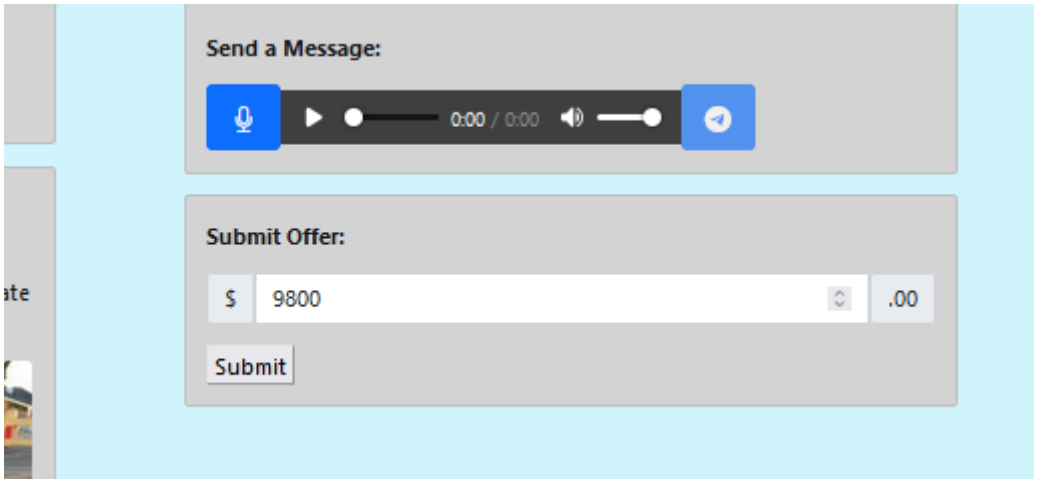
Role-Playing

- Imagine yourself in the role of the buyer
    - Try to think of compelling reasons why you would want to buy this item
    - For example… (Christmas present, low budget)
    - Basically reframe this to "play the role", not "make up whatever you have to, to win"

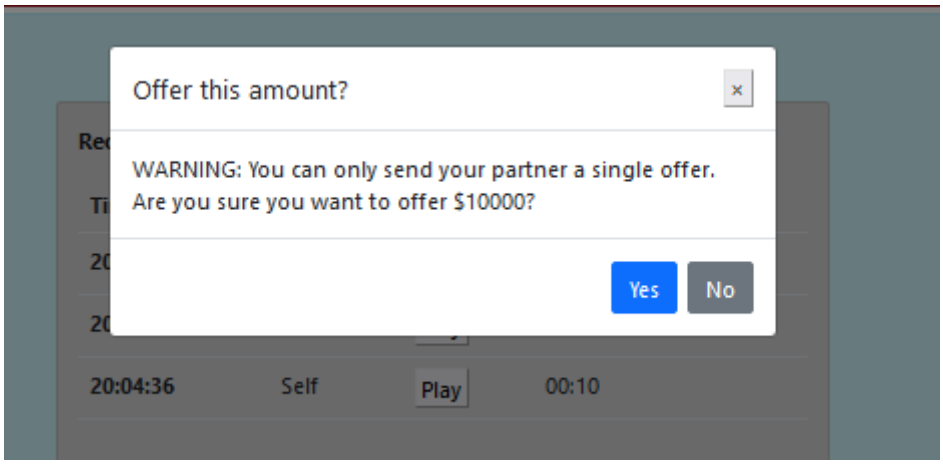
You may want to make up information about your situation as a buyer for the sake of the negotiation. For example, you may want to come up with a compelling reason for why your budget is as low as it is, or invent a heartwarming reason for why you want this particular item ("It's a Christmas present for my children!"). You are welcome to do so! Say anything you feel will be helpful for the sake of the negotiation.

Making an offer

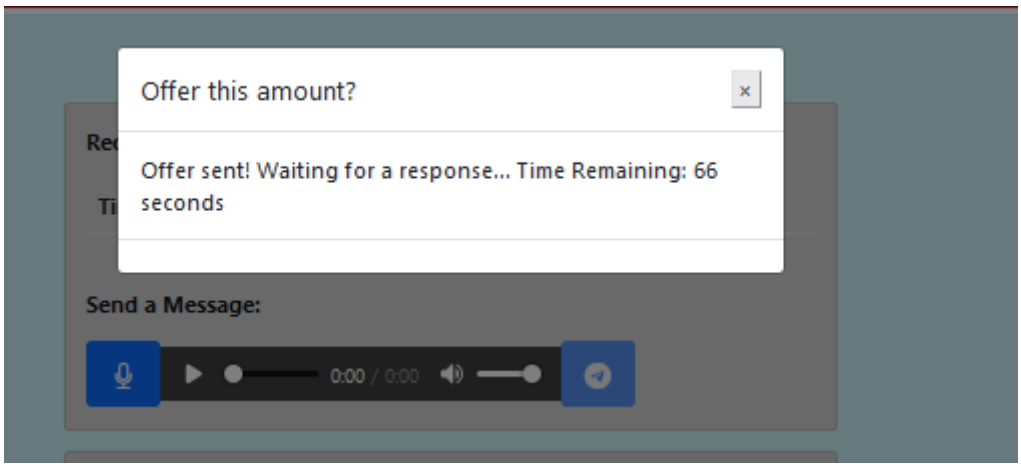
Once you and your partner have reached an agreement, you will want to send your final offer. You can do so by entering a dollar amount in the textbox labelled "submit offer". **Remember, you can only send whole dollar values, so make sure you only enter whole numbers!**



Also, **you can only send a single offer,** so make sure you're prepared for the negotiation to end after you send it.



Once you send the offer, you'll need to wait until your partner sends their response to see if it's accepted or rejected. If it's rejected, the negotiation will be considered a failure, and neither of you will receive a bonus. You'll want to make sure you send an offer you've both agreed on!



Once the offer is accepted or rejected, the negotiation is over. Thanks in advance for your participation.

Task (Seller)

Role-Playing

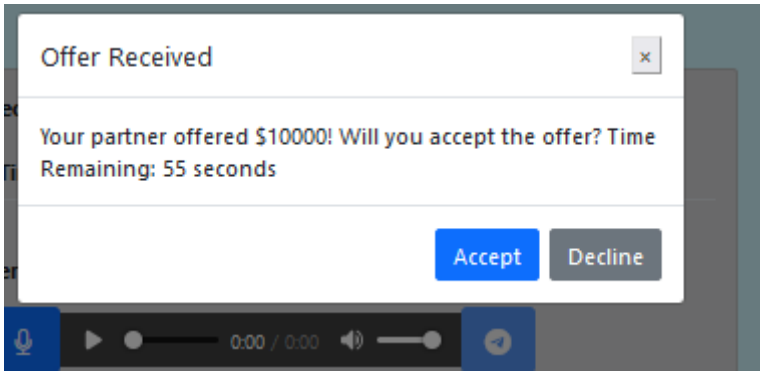
- Imagine yourself in the role of the seller
    - Try to think of compelling reasons why you would want to buy this item
    - For example… (Christmas present, low budget)

You may want to make up information about your situation as a seller for the sake of the negotiation. For example, you may want to come up with a compelling reason for why your price is so high, or invent a heart-wrenching reason for why you can't afford to sell any lower ("I need this money to pay for groceries next week…"). You are welcome to do so! Say anything you feel will be helpful for the sake of the negotiation.

Accepting or Rejecting an Offer

Once you and your partner have reached an agreement through your messages, they will send you an offer. You have the option to accept or reject it; while you both get a bonus payment if you accept, remember that your bonus is smaller the further you are from your goal price.

Also, you will have a time limit to accept or reject; if you wait too long, it will be counted as an automatic rejection.



Once you accept or reject the offer, the negotiation is over! Thanks in advance for your participation.

Appendix 3

Example Negotiations

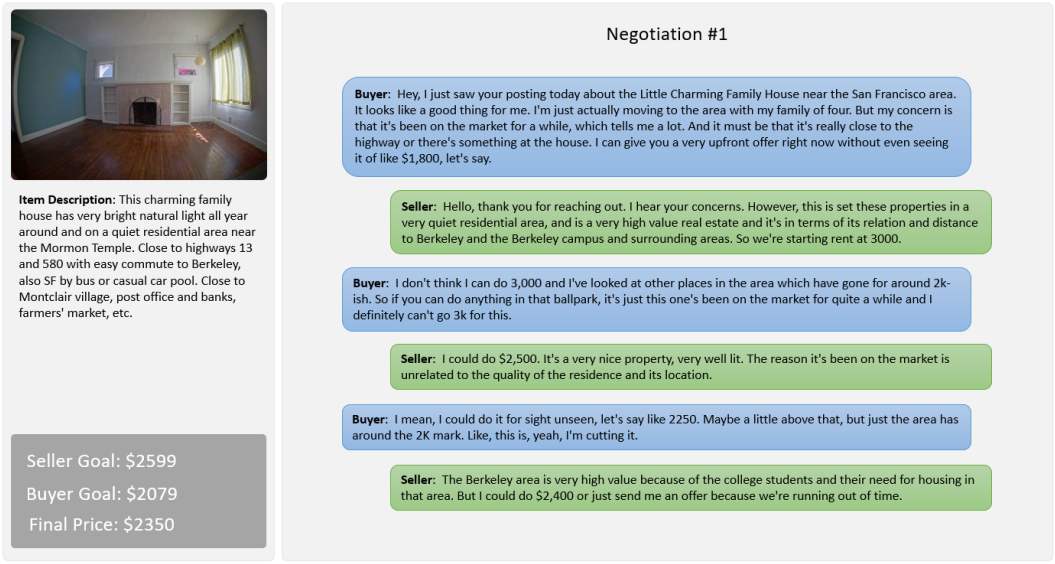


Figure 3.1: Transcript of example negotiation 1.

DARE Transcript	CB Transcript
buyer: Hey, I just saw your posting today about the Little Charming Family House near the San Francisco area. It looks like a good thing for me. I'm just actually moving to the area with my family of four. But my concern is that it's been on the market for a while, which tells me a lot. And it must be that it's really close to the highway or there's something at the house. I can give you a very upfront offer right now without even seeing it of like $1,800, let's say.	Hi are you willing to go down to $2000?
seller: Hello, thank you for reaching out. I hear your concerns. However, this is set these properties in a very quiet residential area, and is a very high value real estate and it's in terms of its relation and distance to Berkeley and the Berkeley campus and surrounding areas. So we're starting rent at 3000.	Hi there. I'd like to get closer to my asking price. How about 2400?
buyer: I don't think I can do 3,000 and I've looked at other places in the area which have gone for around 2k-ish. So if you can do anything in that ballpark, it's just this one's been on the market for quite a while and I definitely can't go 3k for this.	I don't have that much. I would be comfortable going $2300.
seller: I could do $2,500. It's a very nice property, very well lit. The reason it's been on the market is unrelated to the quality of the residence and its location.	Sure, $2300 sounds reasonable. I can do that.
buyer: I mean, I could do it for sight unseen, let's say like 2250. Maybe a little above that, but just the area has around the 2K mark. Like, this is, yeah, I'm cutting it.	
seller: The Berkeley area is very high value because of the college students and their need for housing in that area. But I could do $2,400 or just send me an offer because we're running out of time.	

Figure 3.2: Negotiation 1: DARE vs. CraigslistBargain

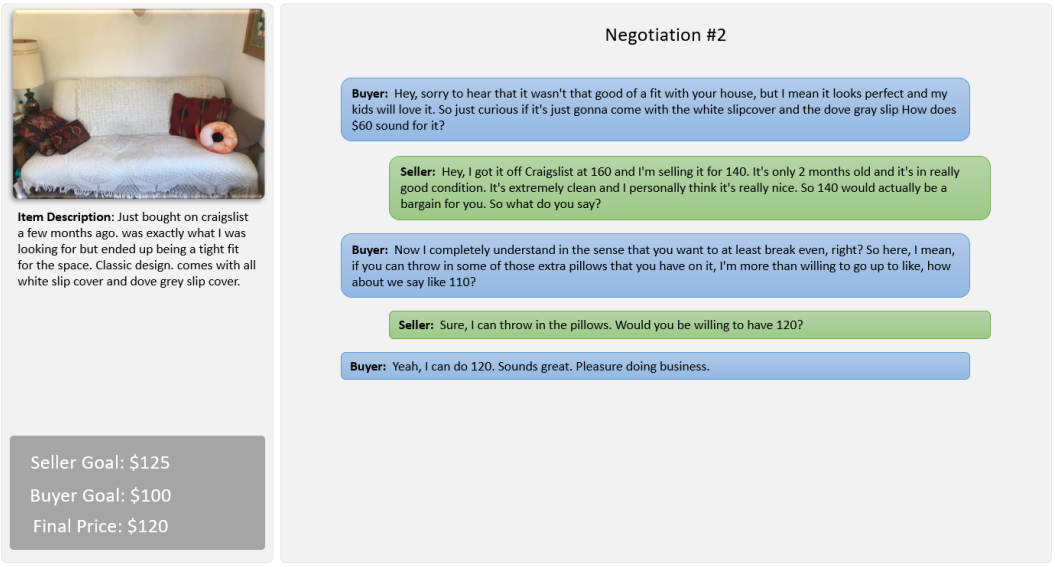


Figure 3.3: Transcript of example negotiation 2.

DARE Transcript	CB Transcript
buyer: Hey, sorry to hear that it wasn't that good of a fit with your house, but I mean it looks perfect and my kids will love it. So just curious if it's just gonna come with the white slipcover and the dove gray slip How does $60 sound for it?	What is the condition of the futon?
seller: Hey, I got it off Gregor's at 160 and I'm selling it for 140. It's only 2 months old and it's in really good condition. It's extremely clean and I personally think it's really nice. So 140 would actually be a bargain for you. So what do you say?	It's used. I got it on craigslist a couple months ao and it's in great condition. Just doesn't fit in my house.
buyer: Now I completely understand in the sense that you want to at least break even, right? So here, I mean, if you can throw in some of those extra pillows that you have on it, I'm more than willing to go up to like, how about we say like 110?	Well, at $125 it would need to come with a lot more than a slip cover. Please make me a better offer.
seller: Sure, I can throw in the lows. Would you be willing to have 120?	I will be willing to accept 100, it comes with two grey slip covers.
buyer: Yeah, I can do 120. Sounds great. Pleasure doing business.	It would need to come with a $20 bill tucked in each slip cover if you expect me to pay $100. I'll offer $75, my final offer.

Figure 3.4: Negotiation 2: DARE vs. CraigslistBargain

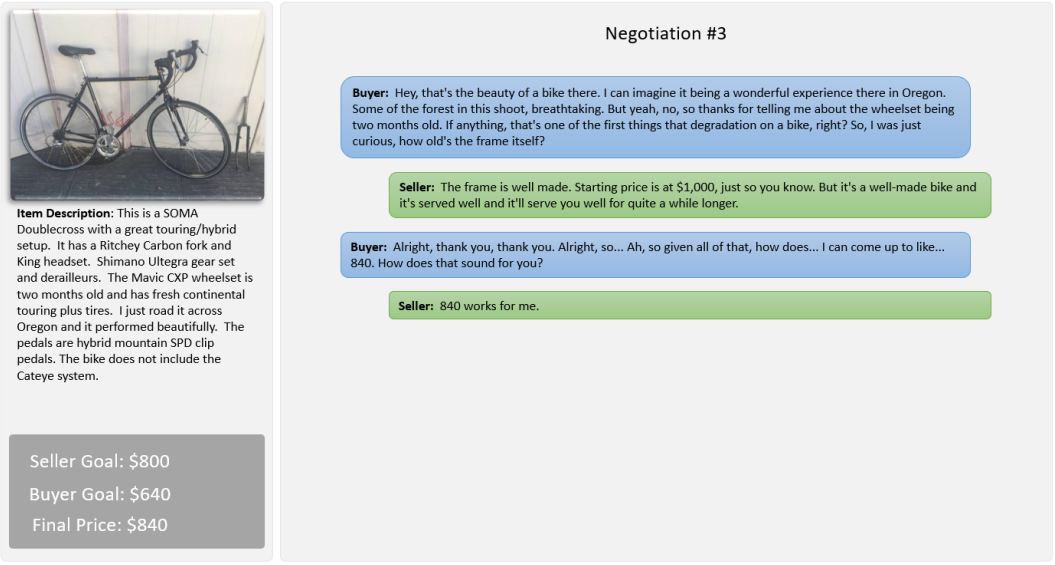


Figure 3.5: Transcript of example negotiation 3.

DARE Transcript	CB Transcript
buyer: Hey, that's the beauty of a bike there. I can imagine it being a wonderful experience there in Oregon. Some of the forest in this shoot, breathtaking. But yeah, no, so thanks for telling me about the wheelset being two months old. If anything, that's one of the first things that degradation on a bike, right? So, I was just curious, how old's the frame itself?	Hello what can you tell me about this bike?
seller: The frame is well made. Starting price is at $1,000, just so you know. But it's a well-made bike and it's served well and it'll serve you well for quite a while longer.	Hi! I have this bike and it is in great condition. The wheelset is two months old. I have thinner tires for road touring, but would also be willing to include thicker tires for riding dirt trails, if you're willing to spend $800.
buyer: Alright, thank you, thank you. Alright, so... Ah, so given all of that, how does... I can come up to like... 840. How does that sound for you?	This sounds like a good bike. How often have you used it?
seller: 840 works for me.	I've only used it 5 times. Once on a road tour across Oregon, and a few times on dirt trails in my hometown. How much are you looking to pay for this?
	736
	I can't go that low. Since I want 800 and you want 736, let's meet in the middle at 768. That way we both get what we want. I'll even include both sets of tires at that price. How does that sound?
	If you throw in a warranty, I will meet you at 770
	Ok. I can throw in a 2 year warranty for 770.

Figure 3.6: Negotiation 3: DARE vs. CraigslistBargain

Item Description: Gently used, pre-LOVED iPad 2 for sale. I purchased a new iPad & my prior device needs a new home.

Seller Goal: $175
Buyer Goal: $140
Final Price: $150

Negotiation #4

Buyer: Hey there, I saw your listing for the iPad. Does it come with the charger and the box and everything? Also, is that a case in the background?

Seller: Hi, thanks for the interest! And yes, everything that you see there in the picture is included. So that includes that little pamphlet that you see on the right, the charger, the cable, and yes, you are correct, that is the case.

Buyer: Oh, that's awesome. Okay, so it comes with everything. Let's see, the iPad 2 is a pretty old model. So I think... how does a hundred sound for the whole set?

Seller: Yeah, no, that's completely fine, especially with the fact that, you know, it's not even updating to some of the newest iOS versions, so I completely get that. But yeah, no, I'm sorry, but 100's a little bit too low for me, in the sense that I completely understand it's one of the earliest models, but when I say gently used, I mean it's in the sense that I only used it as pretty much as a Kindle, because some of the book reading and PDF reading features on it were just phenomenal. But either way, so because of that, it might as well just be brand new. But yeah, that including the charger and the cable and throwing in a case, can you meet me potentially at like 150?

Buyer: Okay, so it sounds like it wasn't used very much at all, mainly just used at home. Does it come with any books? You said you mainly used it as a Kindle, does it still have some PDFs or books on it?

Seller: Yes, it most definitely does. I swear it has half my library on there. So expect to see some of the classics, but over a catalog of, I want to say like, 250 books, which you're more than welcome to.

Buyer: In that case, consider me sold.

Figure 3.7: Transcript of example negotiation 4.

DARE Transcript	CB Transcript
buyer: Hey there, I saw your listing for the iPad. Does it come with the charger and the box and everything? Also, is that a case in the background?	Hey there! Does the iPad have any dents or scratches?
seller: Hi, thanks for the interest! And yes, everything that you see there in the picture is included. So that includes that little pamphlet that you see on the right, the charger, the cable, and yes, you are correct, that is the case.	No it is in like new condition.
buyer: Oh, that's awesome. Okay, so it comes with everything. Let's see, the iPad 2 is a pretty old model. So I think... how does a hundred sound for the whole set?	Do the headphone and charging ports work properly?
seller: Yeah, no, that's completely fine, especially with the fact that, you know, it's not even updating to some of the newest iOS versions, so I completely get that. But yeah, no, I'm sorry, but 100's a little bit too low for me, in the sense that I completely understand it's one of the earliest models, but when I say gently used, I mean it's in the sense that I only used it as pretty much as a Kindle, because some of the book reading and PDF reading features on it were just phenomenal. But either way, so because of that, it might as well just be brand new. But yeah, that including the charger and the cable and throwing in a case, can you meet me potentially at like 150?	Yes everything works properly.
buyer: Okay, so it sounds like it wasn't used very much at all, mainly just used at home. Does it come with any books? You said you mainly used it as a Kindle, does it still have some PDFs or books on it?	Great. If I pick it up, would you be willing to let it go for $133?
seller: Yes, it most definitely does. I swear it has half my library on there. So expect to see some of the classics, but over a catalog of, I want to say like, 250 books, which you're more than welcome to.	That is a little lower than I would like. Can you come up a bit?
buyer: In that case, consider me sold.	The highest I can afford is $150, and I'll throw in a new case for your new iPad.

Figure 3.8: Negotiation 4: DARE vs. CraigslistBargain

Bibliography

[1] Chris Callison-Burch, Lyle Ungar, and Ellie Pavlick. Crowdsourcing for nlp. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 2–3, 2015.

[2] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[3] What is css? - learn web development: Mdn. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS.

[4] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

[5] Sam Dutton. Record audio and video with mediarecorder. URL: https://developer.chrome.com/blog/mediarecorder/.

[6] Justin Ellingwood. Postgresql advantages: Benefits of using postgresql. URL: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql.

[7] Meta Fundamental AI Research Diplomacy Team (FAIR)†, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried,

Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

[8] Ffmpeg. URL: https://ffmpeg.org/.

[9] Alexander Gruenstein, Ian McGraw, and Ibrahim Badr. The wami toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces. In *Proceedings of the 10th International Conference on Multimodal Interfaces*, ICMI '08, page 141–148, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1452392.1452420.

[10] He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. *arXiv preprint arXiv:1808.09637*, 2018.

[11] Html: hypertext markup language — mdn. URL: https://developer.mozilla.org/en-US/docs/web/html/element/audio.

[12] Https - mdn web docs glossary: Definitions of web-related terms: Mdn. URL: https://developer.mozilla.org/en-US/docs/Glossary/HTTPS.

[13] What is https? — cloudflare. URL: https://www.cloudflare.com/learning/ssl/what-is-https/.

[14] Jessica Huynh, Jeffrey Bigham, and Maxine Eskenazi. A survey of nlp-related crowdsourcing hits: what works and what does not. *arXiv preprint arXiv:2111.05241*, 2021.

[15] Tonino Jankov. Nginx vs apache: Web server showdown, Oct 2022. URL: https://kinsta.com/blog/nginx-vs-apache/.

[16] What is javascript? - learn web development: Mdn. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[17] Jinja homepage. URL: https://jinja.palletsprojects.com/en/2.10.x/.

[18] Michael Klein. What is bootstrap: A beginner's guide to bootstrap, Sep 2021. URL: https://www.codecademy.com/resources/blog/what-is-bootstrap/.

[19] Malhar Lathkar. Introduction to fastapi. In *High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python*, pages 1–28. Springer, 2023.

[20] Daniel A McFarland, Dan Jurafsky, and Craig Rawlings. Making the connection: Social bonding in courtship situations. *American journal of sociology*, 118(6):1596–1649, 2013.

[21] Mediadevices - web apis: Mdn. URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia.

[22] Mediarecorder - web apis: Mdn. URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder.

[23] Amazon mechanical turk api reference. URL: https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/Welcome.html.

[24] What is nginx?, Apr 2023. URL: https://www.nginx.com/resources/glossary/nginx/.

[25] What is a reverse proxy? — proxy servers explained — cloudflare. URL: https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/.

[26] Patricia Saylor. *Spoke: A framework for building speech-enabled websites.* PhD thesis, Massachusetts Institute of Technology, 2015.

[27] Jack Urbanek and Pratik Ringshia. Mephisto: A framework for portable, reproducible, and iterative crowdsourcing, 2023. arXiv:2301.05154.

[28] Websockets - web apis: Mdn. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[29] Introducing whisper. URL: https://openai.com/research/whisper.

Vita

Rishikumar Salem was born in Cleveland, Ohio on 21 December 1998, the son of Hemanthkumar Salem and Priya Salem. He received the Bachelor of Science degree in both Computer Science and Psychology from the University of Texas at Austin. He was accepted into the Computer Science Integrated BS/MS program in May 2021.

Permanent address: 13020 Bismark Drive
Austin, Texas 78748

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.