FlatFinder: A conversational agent for student accommodations

Matteo Mascherin
Dipartimento di Ingegneria e Scienza dell'informazione, Università di Trento
matteomascherin@studenti.unitn.it

***Abstract*—In this work, I present FlatFinder, a conversational agent designed to assist students in finding their ideal accommodation. Conversational agents can vary in complexity, from simple rule-based systems that follow predefined scripts to advanced AI models capable of adapting dynamically based on conversation history and user intent. They are widely used across multiple domains, including customer support, healthcare, education, and entertainment. Powered by the open-source Meta Llama 3-8B model, the proposed agent understands and interacts with users through natural language. The design incorporates multiple components to enhance coherence and control, emphasizing the importance of separating distinct tasks, each handled by specialized LLM-based experts. By adopting this system architecture, integrating a database to connect the agent with external information sources becomes seamless. FlatFinder supports users throughout the accommodation search process, covering key phases such as filter-based searching, property selection, comparative analysis, and inquiry handling for specific listings. All these functionalities are integrated into a structured conversation, ensuring a smooth user experience while adhering to all domain constraints of the agent.**

I. Introduction

A Conversational agent, also known as a human dialogue system, is an AI-driven system designed to engage users in natural language interactions. These systems leverage computational linguistics, machine learning, and natural language processing (NLP) techniques to understand, generate, and respond to user queries in a coherent and contextually relevant manner.

Fundamentally, these systems aim to simulate human-like dialogue by maintaining engagement, ensuring relevance, and responding appropriately based on the user's needs. Mixed-initiative dialogue mechanisms allow both the agent and the user to take turns steering the conversation, fostering a more natural and interactive experience. Additionally, modern dialogue systems incorporate strategies to manage errors, ambiguities, and conversational coherence, ensuring that interactions remain meaningful and effective.

FlatFinder is a conversational agent designed to assist students in finding suitable accommodation. This work aims to provide a valuable tool that simplifies the house-hunting process for teenagers seeking a home or shared apartment for their studies abroad. By integrating an external database of available properties - using an Indian one as an example[1] - users can access updated housing information and inquire about specific listings.

[1]see Appendix B

FlatFinder is tailored specifically for students, ensuring that the unique needs of young people in a study-oriented living environment are considered. A typical interaction with the agent begins with an initial search based on general parameters, followed by more refined inquiries focused on more specific details.

Throughout the conversation, the system ensures coherence by continuously verifying its current state. Additionally, mixed-initiative dialogues are supported, allowing the agent to ask questions and confirm user intents to facilitate interaction. While enabling these features, the conversation remains strictly focused on house searching through a fallback policy, which helps guide the user in cases of internal system errors or inappropriate requests.

The code of this project can be found at: https://github.com/MatteoMaske/ConversationalAgent/

II. Conversation design

In order to implement a good conversational agent, we need to make sure that we define well the specific domain in which the system will operate. In this work, the typical conversation is divided into two parts: first there is a house search, and this is followed by some information exchange on the proposed houses. Since the target users for the system are students, a lot of information regarding shared-apartment can be found, such as: tenant profile preferred by landlord, number of bathrooms, and also in which floor is the apartment located. This allows users to adjust the search to match their personal needs. To enhance further this concept, house comparison is also handled, so that multiple factors and accommodations can be considered at the same time. In this project, four main interactions are handled:

1) house_search: the user wants to look for a house and can provide some filters for the research
2) house_selection: among a list of houses proposed by the system, the user chooses to focus on one of them
3) house_comparison: among the list of proposed houses, the user expresses interest in comparing some of them on some features
4) ask_info: the user asks specific information about a specific house.

A. Conversation properties

The agent is designed to be **mixed-initiative**, in fact it leaves space for the user to initiate the conversation and add

any detail at any stage in the chat. The system supports **over-informative** users, as well as **under-informative** ones, by leveraging different techniques. In case of over-information, all the memorized slots will just have one single value, which could also be redefined later in the conversation. If more intents are expressed, by dividing the sentence into chunks, only one intent is identified for each chunk. Under-informative users are handled by simply asking missing information in order to fill each slot for a certain intent.

After the conversation reaches a complete intermediate phase, the state of the system is tracked and **coherence** is guaranteed throughout the whole chat. Moreover, the incoherence that the user may show, is addressed using **confirmations** between turns and making sure the state of the system is always in a valid state. By instructing the system model using custom prompts, the agent can lexicalize properly formatted messages containing conversational markers as well as error messages which serve as a **fallback policy** to provide a smooth and controlled user experience. If the conversation gets interrupted for any reason, the last *active* state is memorized and can be used later to restore the chat.

Engagement is a crucial aspect of the agent, reinforced through carefully crafted system prompts within the natural language generator component. Specifically, for each next best action that needs to be lexicalized, an appropriate system prompt is selected to ensure the agent remains fully aware of the current conversation state.

III. Conversation model

The proposed pipeline utilizes the Meta Llama 3-8B model [1] to power the core components of the system. The architecture consists of three fundamental modules:

1) *Natural Language Understanding* (NLU) – This component processes user queries by identifying and classifying relevant intents and slots to extract meaningful information.
2) *Dialogue Manager* (DM) – Based on the structured output from the NLU, the dialogue manager determines the next best action from a predefined set of possible actions.
3) *Natural Language Generation* (NLG) – Responsible for the final stage, where the selected action is formulated into a coherent response using the recent conversation history.

A schematic representation of the designed pipeline is provided in 1.

Each of these components is implemented through individual requests to an open-source LLM [1], leveraging specialized domain-specific system prompts. These tailored prompts ensure each module operates independently while maintaining a cohesive interaction flow. The system is designed to be adaptable across different LLMs, although minor prompt tuning may be required for optimal performance.

The implementation of the NLU, DM, and NLG follows a common schema:

- *Prompt preparation* – The input, whether originating from the user or system, is structured with a system prompt and relevant conversation history is attached.
- *Response generation* – The LLM processes the prepared prompt and generates a response.
- *Post-processing* – The response is parsed into JSON format to ensure integration within the system (only for NLU and DM).

In addition to the core modules, a *state tracker* maintains intermediate system states and ensures the conversation progresses logically. This component performs several crucial functions:

- *update* – Adjusts the system state based on recognized intents and extracted slots from the NLU.
- *initialize_slots* – Assigns default values to all slots, enabling tracking of subsequent modifications.
- *handle_intent* – Processes information once all required slots for a given intent are filled, ensuring accurate state updates.
- *fallback_policy* – Retains the last active state and provides users with a structured fallback response in case of system errors or inappropriate queries.

This modular design enhances control, coherence, and adaptability, allowing the system to deliver a structured and user-friendly interaction experience.

A. Natural language understanding

The Natural Language Understanding (NLU) component of the conversational agent is responsible for mapping user inputs to predefined intents, ensuring accurate interpretation and appropriate responses. The system processes only the latest user request, while optionally considering previous interactions for context.

The following intents are recognized within the system:

- **house_search**: This intent is triggered when the user expresses a desire to search for available student housing or refine search criteria. Example queries include requests for accommodations in specific locations, budget constraints, or particular housing features.
- **house_selection**: When a user shows interest in a specific house and requests additional details or wants to proceed with that property, this intent is activated. Example inputs include selecting a particular listing or asking to focus on one property.
- **compare_houses**: The system identifies this intent when the user seeks a comparative analysis between two or more houses or specific features of multiple properties. Examples include comparisons based on rent, number of rooms, or location attributes.
- **ask_info**: Users requesting details about a specific house, such as the number of floors, available amenities, rent, or owner contact information, trigger this intent. It ensures precise responses tailored to the property being inquired about.
- **out_of_domain**: Any user request that does not fall within the domain of student accommodations in India, such as queries unrelated to housing, is classified under this intent. This ensures the conversational agent remains focused on its primary function.

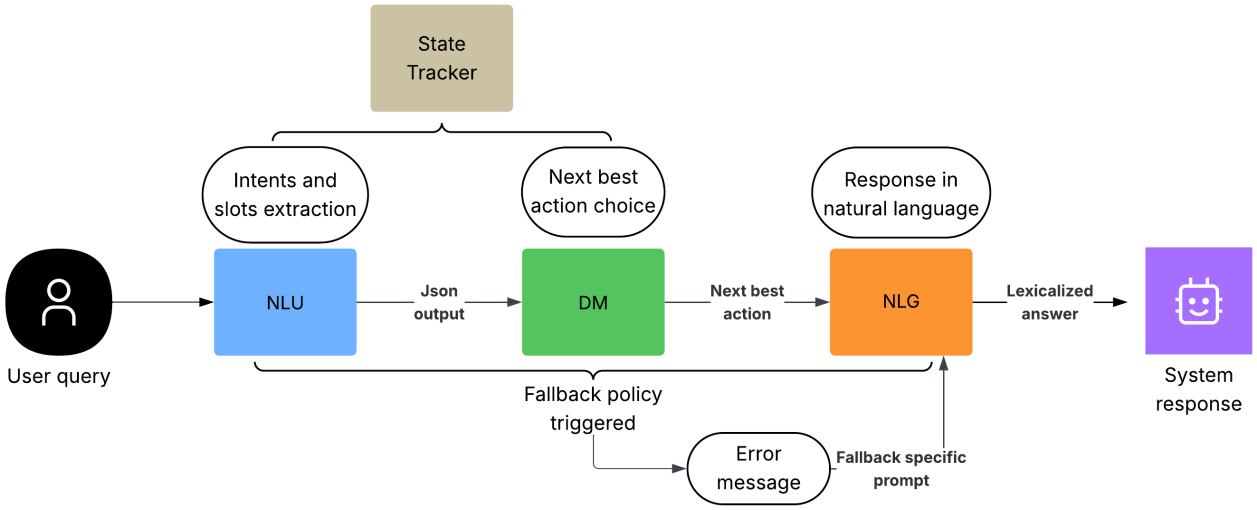

Fig. 1. A simple scheme of the proposed system

Each intent is associated with a set of slots (see Appendix A) that capture specific information extracted from user input in a structured form. Slots can represent integers (e.g., rent or number of rooms), strings (e.g., location names), lists (e.g., selected houses or requested features), or remain null when the information is missing.

By leveraging intent and slot-based information extraction, the system ensures that conversations remain structured and that intent recognition aligns seamlessly with downstream processes, such as DM and NLG.

B. Dialogue Manager

The Dialogue Manager (DM) is responsible for orchestrating the conversation flow by determining the next best action based on the user's intent and slot values. This component ensures structured interactions, maintaining coherence and logical progression throughout the dialogue.

The DM operates using predefined actions to manage user input effectively:

- **Slot Request Handling**: If a required slot value is missing, the system prompts the user to provide the missing information. This ensures that all necessary details are gathered before proceeding.
- **Intent Confirmation**: When all required slots for a given intent are filled with valid values, the system confirms the user's intent, signaling that the request is fully understood and ready for execution.
- **Information Retrieval**: For specific queries related to property details, the system needs to retrieve and provide relevant information based on explicitly requested properties.

C. Natural Language Generation

The Natural Language Generation (NLG) component is responsible for generating coherent and contextually appropriate responses based on the user's intent and extracted slot values. It ensures that the conversational agent delivers structured, informative, and engaging responses while maintaining domain constraints and user experience consistency.

In order to provide a complete and meaningful prompt for each scenario, more templates are used in order to precisely instruct the LLM to tackle every possible situation. One of the prompts is specifically reserved for lexicalize error messages raised according to the *fallback policy*. The modular approach to prompt-based response generation enables scalability across different conversational scenarios.

IV. Evaluation

Evaluating a conversational agent involves assessing both its internal mechanisms and real-world performance to ensure effectiveness, coherence, and user satisfaction. This section outlines two primary evaluation methodologies: intrinsic evaluation, which focuses on the system's internal quality, and extrinsic evaluation, which measures its impact on user interactions.

A. Intrinsic Evaluation

Intrinsic evaluation focuses on the internal performance of the conversational agent, measured using exact metrics such as F1-score. A critical component of the intrinsic evaluation is having a ground truth target, which has to be aimed by the system. The only components that can be tested using this method are NLU and DM, since their output has to adhere to a fixed schema.

In order to evaluate the NLU, string injection is employed to generate a certain amount of test samples starting from some hand-crafted templates. On top of few human crafted template, Claude 3.5 Sonnet [2], has been used to increase the number of templates and ensure diversity in the content. For each intent, there is a fixed number of template (10) to ensure all possible scenarios are equally represented. The templates can be seen in Appendix C. By using placeholders, the value for each slot is randomly generated and placed inside the template in the correct position. Exploiting this method, it is straight forward to generate also the NLU ground truth for each test user query. Both intent classification and slots extraction are evaluated in term of accuracy, F1, precision and recall, taking the macro-average for intents and micro-average for slots. The intent accuracy reached is 0.93 and the other results are shown in IV-A and II. The confusion matrix for intent classification can be seen in 2.

Intent name	Precision	Recall	F1
House search	1	1	1
House selection	0.88	0.97	0.92
Ask info	0.79	0.87	0.83
Compare houses	1	0.80	0.89
Out of domain	1	1	1

TABLE I
RESULTS IN TERMS OF PRECISION, RECALL AND F1 FOR INTENT CLASSIFICATION

From the evaluation, intent classification turns out to be almost perfect, in fact the only intents confused are *ask_info* and *compare_houses* which are the most conceptually similar one.

In evaluating slot extraction, typos and type mismatch in the extracted json by the NLU, have to be considered in order to account just for major errors. Driven by this observation, a fuzzy ratio [3] can be applied to measure the similarity between the extracted slot values and their expected values, allowing for minor differences (such as typos or slight variations in formatting) while flagging major discrepancies as errors. By setting a threshold for the fuzzy ratio score of 0.8, all irrelevant mismatches have been filtered out, ensuring a more robust evaluation of the NLU's performance.

Intent name	Precision	Recall	F1
House search	1	1	1
House selection	0.94	0.87	0.87
Ask info	0.78	0.80	0.79
Compare houses	0.72	0.71	0.70

TABLE II
RESULTS IN TERMS OF PRECISION, RECALL AND F1 FOR SLOTS EXTRACTION, GROUPED BY INTENT NAME.

Similarly, to evaluate the Dialogue Manager (DM), for each intent a list of sample NLU outputs were either hand-crafted or AI-generated, and accuracy was computed. The evaluation compared two different implementation strategies: LLM-based and deterministic rule-based logic.

The LLM-based DM achieved an accuracy of **0.93**, demonstrating strong performance in intent-to-action mapping. This approach is particularly robust in handling noisy or ill-formed input, such as user queries with typos or uncommon phrasing, thanks to the generalization capabilities of the model. However, occasional errors were observed due to *hallucinations*—instances where the model generated plausible but incorrect outputs.

On the other hand, a deterministic implementation using strict rule-based logic was also evaluated. In this case, since the decision mechanism is entirely hard-coded, a perfect accuracy of **1.00** was reached. This approach is straightforward to implement, interpretable, and completely predictable. However, it may fail to handle input variations that deviate from predefined patterns and does not scale well to complex or evolving systems with a large number of intents and actions.

B. Extrinsic Evaluation

Extrinsic evaluation measures the agent's impact on user experience and practical usability. This involves real-world testing and user-centred evaluations, including user satisfaction surveys, task success rate and interaction efficiency.

Explicit evaluation was conducted by directly involving users in the assessment process through surveys and task-based metrics. The results in Appendix E show the feedback gathered from 10 users who interacted with the assistant in a real-world scenario. To ensure a realistic evaluation, participants were selected with diverse backgrounds, ranging from IT students to adults with no prior knowledge of the domain. Each user completed a short questionnaire evaluating four key extrinsic aspects: *task success rate*, *user satisfaction*, *interaction efficiency*, and *real-world deployment*.

From the responses, we observe a generally positive perception of the system. The task success rate indicators are promising: 7 out of 10 users stated that the information provided was sufficient to make a decision, and the assistant scored an average of 4.5 out of 5 for asking the right questions. This reflects a strong alignment between the assistant's dialogue strategy and user needs.

In terms of user satisfaction, the assistant achieved the highest possible score (5) for both clarity of language and professional tone, and a high average rating (3.8) for overall satisfaction. These results suggest that the assistant delivers a fluent and user-friendly interaction experience.

Efficiency metrics also point to good performance, with 80% of users completing their tasks in six or fewer dialogue turns and an average response time score of 3.9. These values highlight the assistant's capability to guide users effectively with minimal friction.

Finally, regarding deployment readiness, 4 users said they would recommend the assistant, and 4 responded "maybe," indicating that while the system shows good potential, some improvements may still be required before wide-scale adoption.

Overall, the explicit evaluation confirms that the dialogue system performs well across key dimensions of usability and user satisfaction, with some areas for refinement based on individual preferences and edge-case behaviours.

V. CONCLUSION AND FUTURE WORK

In this work, I presented **FlatFinder**, a conversational agent designed to manage human requests by adhering to the principles of natural human dialogue. The system effectively guides users through the house-hunting process by engaging in coherent and informative conversations, supported by up-to-date data retrieved from a dedicated database.

Evaluation of individual system components, combined with user feedback, indicates that the proposed solution successfully meets user needs while delivering a pleasant and efficient user experience.

For future work, the range of supported interactions could be expanded, and additional functionalities could be introduced. For instance, providing an interface for landlords to indicate frequently asked questions, so that the system can automatically answer the user on a given knowledge base.

REFERENCES

[1] A. G. et al., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[2] Anthropic, "Claude 3.5 sonnet," 2025, accessed: 2025-05-25. [Online]. Available: https://www.anthropic.com/news/claude-3-5-sonnet

[3] A. Cohen, "Fuzzywuzzy: Fuzzy string matching in python," 2020, accessed: 31 May 2025. [Online]. Available: https://pypi.org/project/fuzzywuzzy/

APPENDIX A
INTENTS AND SLOTS

Below is a summary of the intents and associated slots used in the NLU component of FlatFinder.

- **Intent: `HOUSE_SEARCH`**
  1) `house_bhk`: Number of bedrooms, hall, and kitchen in the house.
  2) `house_size`: Size of the house in square feet (numeric string).
  3) `house_rent`: Monthly rent fee in INR (numeric string).
  4) `house_city`: City where the house is located.
  5) `house_location`: Location within the city, distinct from the city name.
  6) `house_furnished`: Desired furnishing level: *furnished*, *semi-furnished*, or *unfurnished*.
- **Intent: `ASK_INFO`**
  1) `properties`: A list of property attributes the user wants to know about (e.g., *rent*, *size*). Present only if requested.
- **Intent: `HOUSE_SELECTION`**
  1) `house_selected`: Numeric index (0-based) of the selected house from the previously shown list.
- **Intent: `COMPARE_HOUSES`**
  1) `houses`: List of 0-indexed house indices that the user wants to compare. May be `null`.
  2) `properties`: List of property attributes to compare on (e.g., *rent*, *size*). May be `null`.

APPENDIX B
HOUSE DATABASE

The housing database used in this project, is a public one available for free in Kaggle (link). It has been selected since it contains example information which include shared-apartment features for each entry. In fact, details on number of bathrooms, floors and type of tenant preferred by the landlord can be found. All the housing information provided by the agent are fetched from this source and this could be further extended with other cities or more up-to-date data.

APPENDIX C
USER QUERY TEMPLATES

This appendix reports a representative set of user query templates for each intent to assess the quality of the NLU component. Curly braces denote slot placeholders.

Intent: `HOUSE_SEARCH`

1) I'm looking for a {house_bhk} BHK {house_furnished} house in {house_location} within {house_rent} rupees
2) Can you help me find a {house_size} sq ft {house_furnished} apartment in {house_location}?
3) Show me {house_furnished} houses in {house_location} under {house_rent} rupees
4) I need a {house_bhk} bedroom house in {house_location}, {house_city}
5) Looking for {house_furnished} properties in {house_location} around {house_size} square feet
6) Find me a {house_bhk} BHK flat with rent under {house_rent}
7) Search for {house_furnished} homes in {house_location} area
8) I want to rent a {house_size} sq ft house in {house_location}
9) Show available {house_bhk} BHK options in {house_location} under {house_rent}
10) Need a {house_furnished} flat in {house_location} within my budget of {house_rent}

Intent: `HOUSE_SELECTION`

1) I would like to know more about house option {house_selected}
2) Tell me about house number {house_selected}
3) I'd like to move to option {house_selected}
4) I'm interested in house {house_selected}
5) Show me more information about property {house_selected}
6) Let's look at house {house_selected}
7) I want to explore option {house_selected}
8) Give me details about listing {house_selected}
9) Go ahead with the house {house_selected}
10) Select house number {house_selected}

Intent: `ASK_INFO`

1) What is the {info_type} of this house?
2) Could you tell me the {info_type} of the property?
3) I'd like to know the {info_type}
4) What about the {info_type} of this accommodation?
5) Tell me more about the {info_type}
6) Can you share the {info_type} and {info_type_1} information with me?
7) What's the {info_type} and {info_type_1} like in this apartment?
8) I need to know about both the {info_type} and {info_type_1}
9) Please provide me more details about the {info_type} of the house
10) Show me the {info_type} information of this house

Intent: `COMPARE_HOUSES`

1) I'd like to compare houses {house_index} and {house_index_1}
2) What's the difference between house {house_index} and house {house_index_1} in term of {property}?
3) How do houses {house_index} and {house_index_1} compare on {property}?
4) Show me a comparison of properties on the {property}
5) Could you compare the {property} of these houses?
6) I want to compare the {property} between {house_index} option and {house_index_1} option
7) What are the differences between houses {house_index} and house {house_index_1} in {property} and {property_1}?
8) Compare {property} and {property_1} for houses {house_index} and {house_index_1}
9) Show me how houses {house_index} and {house_index_1} differ in terms of {property}
10) Let me see a comparison of houses {house_index} and {house_index_1} focusing on {property}

Intent: `OUT_OF_DOMAIN`

1) I'd like to have an ice cream
2) Which places can you suggest for a good pizza?
3) What's the weather like today?
4) Can you help me book a flight to Paris?
5) Tell me a joke
6) I want to order some groceries
7) How do I make pasta carbonara?
8) What movies are playing at the cinema?
9) Tell me about the story of the majestic Carlo Magno
10) What's the capital of France?

APPENDIX D
INTENT CLASSIFICATION

In this section, the confusion matrix obtained after evaluating the NLU on intent classification, can be examined.

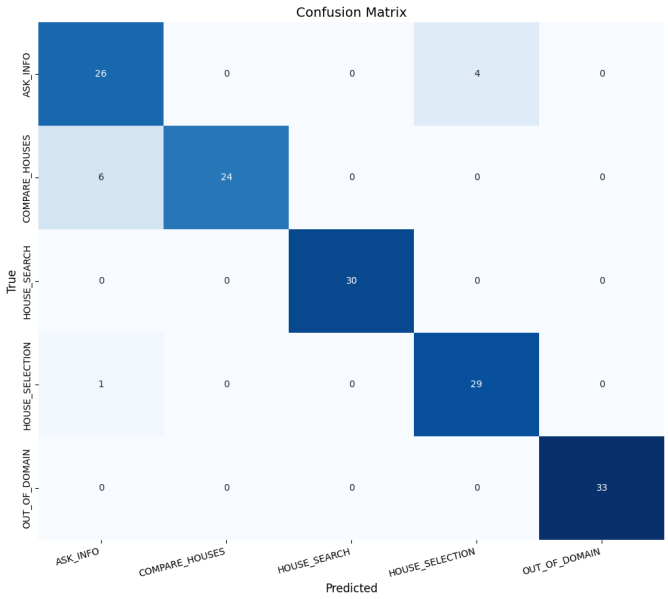

Fig. 2. Confusion matrix for intent classification

APPENDIX E
EXTRINSIC EVALUATION

In this section, the explicit evaluation questions as well as the answers from different users are shown.

Question (possible answers)	**Average Result**
Task Success Rate	
Was the information provided sufficient to make a decision? (Yes / No)	Yes: 7, No: 3
How confident are you that the system understood your needs? (Scale: 1 = Not confident at all, 5 = Very confident)	4.0
Did the system ask the right questions to help you complete your task? (Scale: 1–5)	4.5
User Satisfaction	
How satisfied are you with your experience using this assistant? (Scale: 1 = Very dissatisfied, 5 = Very satisfied)	3.8
Was the language used by the assistant clear and easy to understand? (Scale: 1–5)	5
How natural did the conversation feel? (Scale: 1–5)	4.0
Was the assistant polite and professional? (Scale: 1–5)	5
Interaction Efficiency	
Did the assistant help you complete your task quickly? (Scale: 1 = Very slowly, 5 = Very quickly)	3.9
How many back-and-forth messages did it take to reach your goal? (1–3 / 4–6 / 7+)	1–3: 4, 4–6: 4, 7+: 2
Real World Deployment	
Would you recommend this assistant to a friend looking for housing? (Yes / No / Maybe)	Yes: 4, Maybe: 4, No: 2

TABLE III
EXPLICIT EVALUATION STUDIES CONDUCTED ON 10 USERS.