Build-A-Being Yourself: A practical system and method for embodied conversational agents in web-based research

Anonymous Author(s)

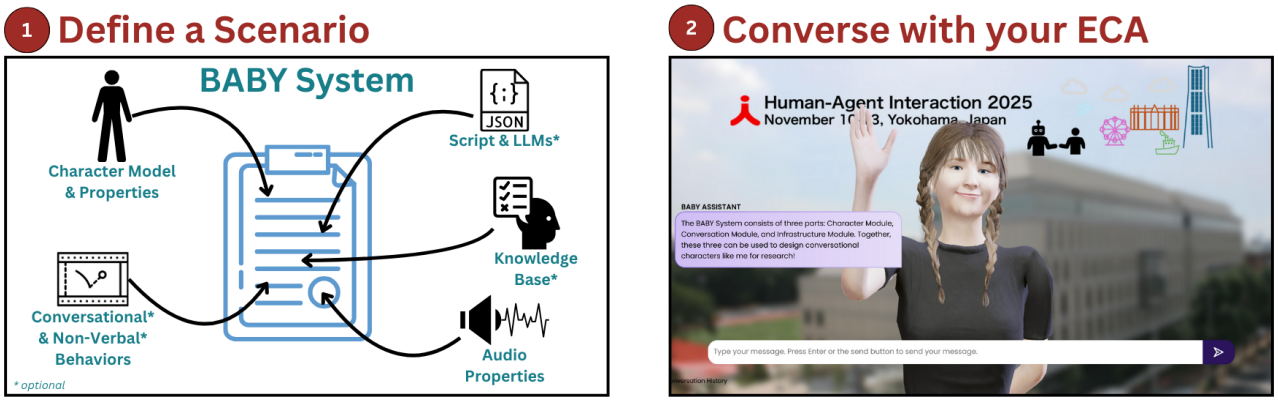

Figure 1: Creating an ECA conversation using BABY can be as simple as defining scenario properties. See Section 4 for details.

Abstract

Embodied conversational agents (ECAs) can extend the reach and impact of real people when time and attention are limited. The benefits of ECAs often lie in their accessibility and scalability, enabling them to meet population demands while significantly reducing costs associated with human counterparts. However, these benefits can only be realized when we can effectively build and deploy ECAs. While existing ECA frameworks offer robust features, many require advanced implementations or allow for little modification. Therefore, we introduce the Build-a-Being Yourself (BABY) system in efforts to help researchers create ECAs practically in *design* and *usage*. BABY achieves this through a fully JavaScript, non-game engine design that is apt for customization and affords the usage of popular AI models and tools. By defining the ECA, a JSON script for conversation, and any desired models, BABY can allow researchers to create a variety of conversations within a single scenario file. In this article, we introduce the three-module architecture for Character, Conversation, and Infrastructure, and the methods to employ BABY for generating ECAs. To conclude, we highlight sample user studies and example customizations through BABY, with a discussion of an ongoing endeavor to streamline the creation of ECA conversations through a platform.

ACM Reference Format:
Anonymous Author(s). 2018. Build-A-Being Yourself: A practical system and method for embodied conversational agents in web-based research. In *Proceedings of XX (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, XX

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Embodied conversational agents (ECAs) have the power to transform individual lives—but only if we can actually build them. When effectively built, ECAs give us the power to extend our reach and impact when human time and attention are limited. As researchers, we have seen the impact ECAs can have across healthcare, training simulations, and education. Seemingly simple, brief conversations with these agents can lead to improved mental well-being [10, 24, 25, 27, 31]. Doctors and nurses use ECAs to reinforce learning and practice applying studied techniques [8, 20, 22, 29]. Perhaps now more than ever, educators are incorporating ECAs into the learning process to support more students and lower barriers to education [17]. Coupling the benefits with advances in large language models (LLMs), ECAs are becoming increasingly capable, conversational, and pertinent. But these benefits and capabilities are meaningless without tools to actually build and deploy them.

This is not the first article to address the development of ECAs in research. Similar systems and frameworks to create ECAs have been successfully developed, refined, and published [1, 4, 15, 26, 28, 32]. To support robust conversations, prior systems often maintain complex, interconnected modules to support speech generation, natural language processing, content generation, or ECA rendering. The design and engineering of these intricate systems have pushed and enhanced the capabilities and qualities of ECAs. Yet, effectively adopting and employing such systems can be challenging for researchers who are less familiar with ECA development. While commercial plug-and-play solutions exist (e.g., SoulMachine, DigitalHumans), they often afford little customizability.

Therefore, we have designed the **Build-A-Being Yourself** system (BABY) to support reproducibility and practical development of ECA research. At a high level, the BABY system can be used

to generate and author open-ended and rule-based conversations with 3D ECAs supported by AI audio and large-language models (LLMs). The primary motivation behind BABY is **practicality** in *design* and *usage*, aiming to empower researchers to start developing ECAs quickly. BABY aims to be practical in design: (1) a condensed architecture consisting of three total modules for character, conversation, and infrastructure (see Section 3), (2) independence from game engines by utilizing only JavaScript (i.e., Node.js, Three.js) in efforts to reduce technical barriers (see Sections 3.1 & 3.3), and (3) interchangeability of audio models and LLMs (i.e., no strict requirements for any singular model) (see Section 3.2). BABY aims to be practical in usage: (1) creating entirely new conversations by replacing desired parameters (e.g., ECA model, audio model) in just *one* file (see Section 4.1) and efforts to support a streamlined approach via web platform (see Section 4.2), (2) a simple, modifiable JSON script to facilitate conversations and LLMs (see Section 3.2), and (3) opportunities for more technically advanced researchers to customize and extend BABY beyond provided in this article (see Section 5). To afford these advantages, we have developed and adopted individual components across a variety of established libraries, tools, and open-source repositories. Our contribution to the Human-Agent Interaction community does not stem from any singular tool's development, but rather from our design and operationalization of the components to enable researchers to begin developing ECAs.

In this article, we detail prior systems in relation to BABY, weighing the advantages of each system and BABY. We describe the core modules of BABY's architecture and opportunities for researchers to adopt and/or further enhance the system as needed. Highlighting its practicality, we provide guidance and resources on self-employing BABY to quickly create virtual conversations through a single file, with mention of ongoing endeavors for practicality through a streamlined platform version. Finally, to demonstrate how BABY might be utilized by researchers, applications of BABY in current user studies and other example customizations are illustrated to conclude the article.

2 Related Work

This section highlights key systems and frameworks that have laid the groundwork for ECA development with overall comparisons to BABY. For clarity, the prior systems are described in the context of *modular* (i.e., designed as interchangeable components) and *integrated* (i.e., used as a unified, all-in-one system) ECA systems; however, the individual ECA systems may contain properties or advantages across both. Within each section, a brief discussion of the unique advantages of prior systems and BABY is provided to help indicate when BABY or another system may be preferable.

2.1 Modular ECA Systems

Three comparable ECA systems are presented in terms of their *design* of modular structures with often standardized or interchangeable modules. Perhaps one of the most well-established systems is the **Virtual Human Toolkit** (VHTK) [11, 15]. VHTK consists of a modular structure for components such as speech, natural language understanding and generation, and agent behavior. A novelty of this system at creation was its *MultiSense* capabilities, allowing agents to interpret non-verbal information from users. To support ECAs, *SmartBody*, *NonVerbal Behavior Generator*, and mark-up languages are used to render the character and animation. A re-released was also published in 2022 to support features like commercial AI services and operability [14]. Another modular ECA system comes from Bickmore et al.'s work with **Relational Agents** (RAs). The architecture's original design is modular in its components like DTask (dialogue) and LiteBody (ECA) [7]. Mark-up languages like XML and BML are used to support the modules' interoperability, while maintaining a lightweight nature [4, 6]. These allow RAs to integrate tailoring, adaptive strategies, and longitudinal conversations to promote user goals and wellbeing [3, 5]. Finally, the **Agents United Platform** (AUP) enables diverse use cases, including personalized coaching, interactive education, virtual customer service, therapeutic support, and immersive entertainment, affording multi-agent interactions. AUP is built upon several modules to perform a series of *sense*, *remember*, *think*, and *act* steps to generate multi-agent conversations [1]. Dialogues use a WOOL Web Service, allowing for conversations to be informed by knowledge bases and memory, with GRETA and ASAP for embodiment.

A competitive advantage of the prior modular systems is in their employment of standardized tools for behavior, conversation, and embodiment (e.g., SAIBA, FML, BML, GRETA). These standardized tools may allow versed ECA researchers to adopt similar systems with greater ease and create robust conversations afforded by systems with complex NLP components and granular control over non-verbal behaviors. Another such advantage is in longitudinal memory of conversations (e.g., RA, AUP) that can allow for repeated interactions with returning users. The tradeoff in standardization has advantages for BABY as it lowers the required knowledge needed for such frameworks, NLP systems, mark-up languages, or typical ECA integrations altogether. In efforts for practicality, BABY also maintains a condensed, modular architecture and streamlines language processing through LLMs and audio models. BABY natively supports *some* granularity for non-verbal behavior through its JSON script conversations and open-sourced resources for popular tools like Mixamo. Though BABY is apt for single-session conversations, the lack of dependence on game engines or individual models enables the opportunity to build upon these capabilities.

2.2 Integrated ECA Systems

Three comparable ECA systems are presented in terms of their *usage* as an integrated system with a standalone, specialized purpose. **Virtual People Factory** (VPF) [28] is designed to facilitate conversations specialized for virtual interviewees. The primary use case is in helping nursing and medical students practice interactions with virtual patients to build self-efficacy [23, 30]. These conversations are designed through crowdsourced dialogues from novices and experts through VPF, which allow for open-ended conversations with ECAs. A suggestion system is used to prompt experts to potential user inputs creating a corpus of responses for ECA conversations. The framework itself is built on the Unity game engine, using physics-based animation and inverse kinematics to allow for preset animations. Conversely, **Virtual Interviewer Platform** (VIP) [31, 32] aims to build conversations for virtual interviewers to

promote healthy behaviors through persuasive models. Using VIP, conversations are driven by the ECA, through branching scripted conversations. VIP is built on a similar framework to VPF, with a WebGL of Unity to render the interaction and preset characters for users. However, the conversations are purely rule-based, rather than in the NLP systems of VPF and VHTK. Both VPF and VIP are utilized as platforms, where researchers can create ECA conversations on web interfaces with little-to-no coding expertise. The **Virtual Agent Interaction Framework** (VAIF) also aims for practicality [12]. VAIF allows users to create ECA conversations and scripted events fully in a singular Unity application, often for virtual and augmented reality. Its usage of third-party tools and libraries allows developers to make ECA interventions with relative ease, like with OVR Lip-Sync (used in BABY). However, the authors note customization limitations, given the reliance on Unity and C#. While not a platform like VPF/VIP, VAIF aims to unify the creation of ECA conversations within one primary editor window.

Competitive advantages of the prior integrated systems arise in their specialized nature, designed to successfully accomplish a *specific* task (e.g., interviews, immersive conversations). Prior integrated systems are also often informed by domain experts or theory-driven models. Therefore, systems like VPF are expected to have greater performance in regard to their intended purposes. In comparison, BABY is aimed to act as a more open template for researchers to adopt, create, or adapt conversations through a single scenario file, akin to AUP, or alternatively, through an early version of a web platform, akin to VIP and VPF (see Section 4). As a result, BABY can support interviewee conversations like in VPF and interviewer conversations like in VIP. BABY's operationalization of the JSON script with LLMs innovates on these systems' lack of support for such tools, potentially reducing development time in systems reliant on crowdsourcing. Finally, BABY's fully JavaScript system allows more advanced researchers to customize to their use cases while limiting the required familiarity with game engines used in these systems.

3 BABY System Architecture

The system architecture is described with respect to its employment as in the *DIY Method* described in Section 4 and its prospective use cases in Section 5. Collaborations used to inform the design choices arise from prior work using ECAs in education, health communications, and clinical practice/training. These collaborations shaped the platform's included features and allowed us to informally refine the system based on real-world use cases[1]. Thus, this section will systematically describe the overarching modules of the collaboratively-informed BABY system: *character module*, *conversation module*, and *infrastructure module*. For any open-source or third-party modules, a brief description of relevant components will be provided. For further details on such material, additional resources containing details for implementation will be provided.

3.1 Character Module

In this section, we will cover the third-party components that are used to *generate*, *lip-sync*, and *animate non-verbal behaviors* of the 3D character model. Though the primary *TalkingHead* component is not designed by the authors, the description of this module should help inform interested researchers who wish to modify aspects of the system and provide context to the other modules in the system.

The BABY system uses the *TalkingHead* open-source repository to lip-sync and animate ECAs for web-based applications. Therefore, the generation of ECAs is designed to be compatible with *TalkingHead*'s lip-sync and animation capabilities. The open-source repository (*TalkingHead*) is built on Three.js and created by a third-party developer named Mike Suominen (**GitHub**: *met4citizen*)[2].

Generation. BABY supports characters in the `.glb` file format, which is a 3D model format that includes properties such as geometry and textures in one compact file. Compatible .glb models can be generated using tools such as ReadyPlayerMe[3], a free-to-use 3D character creation tool for developers. The generated .glb files are relatively small (typically < 5 MB), and ReadyPlayerMe supports compatibility with Mixamo rigs for animation. It's worth noting that ReadyPlayerMe is not the only viable option for this system. However, the designed character will need to include blendshapes to support lip-syncing and animation. As such, Avaturn[4] is another natively supported tool that can be used to create blendshape-enabled .glb models. If one wishes to use other character creators, a summary guide in designing avatars is provided in *TalkingHead*'s appendix items.

Lip-Sync. BABY supports a variety of text-to-speech options to be lip-synced to the ECA by building on TalkingHead's lip-syncing capabilities. *TalkingHead* provides an integrated approach to lip-syncing through word-level timestamps. Given the audio and timestamps of individual spoken words, the *TalkingHead* class can synchronize the ECA's lips to the audio. This process makes use of Oculus's Lipsync visemes and codes (e.g., "sil", "PP", "FF", "TH")[5]. To do this, BABY and *TalkingHead* pre-process the spoken text appropriately (e.g., convert symbols and numbers to words). With the pre-processed text and timestamps, *TalkingHead* generates new visemes, timestamps, and word durations for the lip-synced animations. Naturally, the visemes are the visual representation of the audio phonemes that are used to properly shape the facial blendshapes based on the timestamps and durations. Given the desired audio to output with a timestamp-based transcription, one's 3D character model can be lipsynced to the audio in real time through a singular function call. If one is interested in altering or creating a new lip-sync module (e.g., languages), *TalkingHead*'s appendix items provide recommendations for doing so. One may also be able to directly use the viseme or blendshape data to support other languages, such as with the Microsoft Azure Speech SDK[6].

Non-Verbal Behaviors. BABY allows characters to be animated and posed through *TalkingHead*, which maps well to the ReadyPlayerMe and Avaturn models described in **Generation**. The *TalkingHead* animations include a set of non-verbal templates in the forms of poses (e.g., leaning to side, straight, wide stance), gestures (e.g., waving, thumbs up, shrug), animations (e.g., idle, breathing,

[1] We have anonymized our relation to any prior ECA systems developed by our group and omitted citations to collaborated projects using our *prior* ECA systems.

[2] https://github.com/met4citizen/TalkingHead
[3] https://readyplayer.me/
[4] https://avaturn.me/
[5] https://developers.meta.com/horizon/documentation/unity/audio-ovrlipsync-viseme-reference/
[6] https://github.com/microsoft/cognitive-services-speech-sdk-js

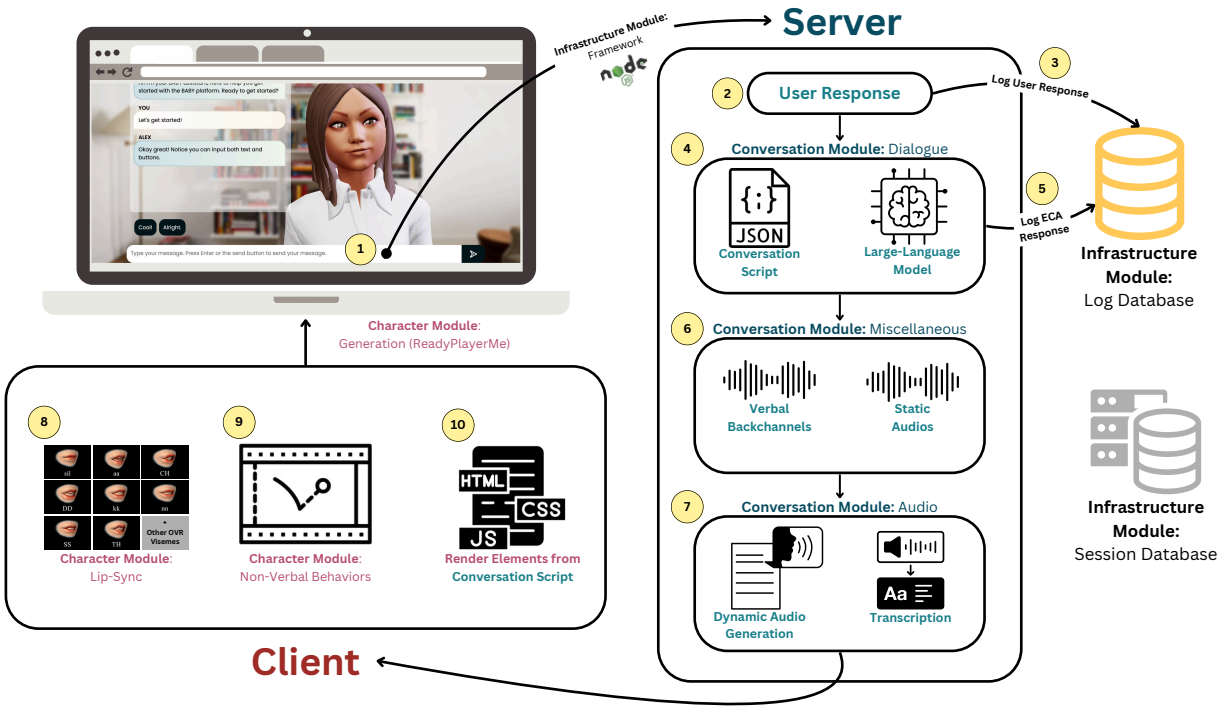

Figure 2: A high-level diagram of the modules described in Section 3 and the order in which the steps occur. The Character Module occurs in the `Client` side. The Conversation Module and Infrastructure Module can be seen in the `Server` side.

speaking), and mood templates (i.e., one can set a mood for the ECA that correspond to specific pose and animation templates). However, the *TalkingHead* class also natively supports Mixamo[7] (.fbx) animations. Interested researchers are encouraged to utilize their appendix items to generate other custom poses, gestures, animations, and moods. With the proper requirements, one can simply attach the *TalkingHead* class and use the natively built-in non-verbal behaviors automatically. If granular control over individual behaviors or customization is desired, the JSON script can be employed or adapted to signal such behaviors from the templates or Mixamo.

3.2 Conversation Module

This section will discuss the *Conversation Module* used to design dialogue for conversational ECAs. BABY can natively support forms of rule-based and open-ended conversations with LLM support. Thus, we will share our method of generating such conversations – performed exclusively through a simple JSON script. While other systems often use mark-up languages or dedicated scripting tools [3, 14, 15], using a JSON file allows less familiar researchers to quickly get started on designing virtual conversations. The explanation and examples will follow our schematics for the conversation script; however, researchers will find once more that they can modify the properties of this script to fit their needs with relative ease. The following section will cover the *conversation script* and *audio* generation of the BABY system, building on the Character Module in Section 3.1.

Conversation Script. The BABY system uses a simple JSON conversation script structure to build conversations for ECAs. The conversation script is designed to emulate rule-based conversations through nodes. Our structure defines nodes as a string `nodeName` in a key-value pair that can be used to reference the contents of that node. Each node contains the following information: (1) **response**: the dialogue of the character with any usage of LLMs, (2) **input**: input options afforded to the user, and (3) **additional media**: any media that one wishes to render to the user.

The **response** object of a node contains a `dialogue` to be spoken by the ECA when the node is triggered. Additionally, any `behavior`'s file path or directive can be included to trigger an animation for the `dialogue`, if not encapsulated in the default behaviors described in the Character Module. The `response` object can optionally include a `useAi` object that specifies parameters for how an LLM can be used to build upon the `dialogue`. We have denoted three main usages for LLMs in our research contexts: `modifyDialogue`, `prependAiDialogue`, and `appendAiDialogue`. Given the base dialogue, researchers can utilize LLMs to deliver variations of the dialogue, prepend an AI response, or append an AI response (see Listing 1 for an example node named `"SampleNode"` with the **response**, **input**, and **additionalMedia** properties). Each of the three usages only requires a single prompt with instructions on generating a modified, prepended, or appended dialogue. One may

[7] https://www.mixamo.com/

```
{
    "SampleNode": { ⟵ This is the nodeName
        "response": { ⟵ Main properties highlighted blue for clarity.
            "dialogue": "Greetings, user.",
            "behavior": "/public/behaviors/Twirl.fbx",
            "useAi": {
                "prependAiDialogue": {
                    "prompt": "Introduce yourself."
                },
                "modifyDialogue": {
                    "prompt": "Vary the greeting."
                },
                "appendAiDialogue": {
                    "prompt": "Ask the user their name."
                },
                "length": 50
            }
        },
        "input": {
            "text": {
                "nextNode": "LoopNode"
            },
            "audio": {
                "nextNode": "LoopNode"
            },
            "button": {
                "options": [
                    {
                        "label": "I would like to remain anonymous.",
                        "nextNode": "LoopNode"
                    },
                    {
                        "label": "I would like to end this conversation.",
                        "nextNode": "END_NODE" ⟵ Some node to end the
                        ↪ conversation
                    }
                ]
            }
        },
        "additionalMedia": {
            "mediaType": "image",
            "link": "www.repository.com/file.png"
    },
    "LoopNode": {
        .
        LoopNode's nextNode's are each set to LoopNode to replicate
      ↪ chatbot functionality.
        .
    },
    .
    .
}
```

Listing 1: Sample conversation script illustrating the JSON structure of a node described in Section 3.2.

also specify a `length` that serves as a general maximum number of words allowed in an LLM-generated response. See Listing 1 for a sample conversation script illustrating each property.

The **input** object contains the options that users have in responding to the given ECA response. Currently, the system allows `text`, `audio`, and `button` options for user input. When used as is, the conversation script's design requires pre-defined knowledge on node paths, but conversations can be designed to be "open-ended", like an FAQ chatbot, by having nodes recursively point to themselves (see "LoopNode" in Listing 1). Each input must have at least one `nextNode` string that corresponds to an existing `nodeName` in the JSON script. When the respective user input is received by the system, the associated `nextNode` is delivered to the user with the appropriate `response` items. Given the rule-based nature, `text` and greencodeaudio options must provide a specification to the `nextNode`. Similarly, if button inputs are allowed, the system expects an array of all desired options, each containing a `label` (i.e., the button text) and the `nextNode`.

Finally, the **additionalMedia** object of a node is an optional object that supports a media supplement for the response. This object expects a preset option of `mediaType` (i.e., image, pdf, web link) and a `link` or path to the media itself. Such media may be useful in contexts where additional resources may be necessary beyond the content spoken by the ECA, such as in educational and health conversations.

Audio. Many audio generation tools can be added, but BABY is integrated with support for ElevenLabs[8], LemonFox[9], and Google Cloud [10] already. These models are apt for BABY as BABY allows for real-time lip-syncing of audio if provided the following four requirements: audio, spoken words in the audio, spoken word timestamps, and spoken word durations. The audio is generated by the audio model, spoken words are derived from the response, and timestamps and durations can often be produced with the audio model, such as in ElevenLabs, LemonFox, and Google Cloud. We suggest using an audio model that supports transcriptions in parallel to improve latency. For applications that require lengthier generation times (e.g., wordier responses or slower audio models), we suggest making use of our integrated *verbal backchannels* or including audio *streaming*. However, BABY can support *all* text-to-speech methods and audio models by transcribing the audio post-hoc. An example of this is integrated using OpenAI's Transcriptions API, which provides the required words, timestamps, and durations properties.

3.3 Infrastructure Module

The Infrastructure Module is designed to structure the Character and Conversation Modules together for web-based research. In this section, we will discuss the supporting *framework* and *database* components of the BABY system that allow it to be usable for research.

Framework. The framework of BABY is built on an `express` framework in Node.js[11] which allows for a simple division between client-side and server-side files and processes. The Character Module's components are built largely on the client side. This takes the burden of rendering and animation from the server to the client instead. Conversely, the server side allows one to securely store and log sensitive information, handle routing, and operate the Conversation Module through the Node.js Express framework. A server-side component is also needed if one wishes to store and log data into secure databases. Game engines, such as Unity or Unreal Engine, are not utilized in BABY since the framework is exclusively built in JavaScript and rendered through Three.js. This allows BABY to be used and customized without prior knowledge of such game engines and potential to run on a wider array of devices that are JavaScript-enabled.

A brief breakdown of the infrastructure, conversation, and character modules to render, animate, and lip-sync the ECA with dialogue and audio is shown in Figure 2. In short, each user response

[8] https://elevenlabs.io/
[9] https://www.lemonfox.ai/
[10] https://cloud.google.com/text-to-speech
[11] https://nodejs.org/

is sent to the Node.js server (Steps 1-2). The user responses are logged (Step 3), and then the next dialogue is generated according to the Conversation Module (Step 4). The generated response is then logged (Step 5) before associated audios, such as verbal backchannels or pre-generated audios, are immediately sent back to the client (Step 6). As these audios are lip-synced and played to the user, the remaining audio is generated and transcribed on the server (Step 7). After each audio is completed, it is streamed back to the client for the rendering process (Steps 8-9). Finally, the client side renders the visual elements associated with the respective node (e.g., buttons from JSON Conversation script, dialogue, images) (Step 10). When the user responds to the dialogue with the given input options, the same process is cycled through. The native BABY system is optimal for semi-structured conversations that have either a user-specified or fixed end to a conversation.

Database. The database is the last component to address for using BABY in research, by handling data collection. There are two primary databases that we recommend including in the platform: *logging* and *session*. Our implementation includes SQL database tables, but other approved forms of storage (e.g., MongoDB) can be used, as in prior systems [32]. The *logging* database is where research-approved data from the user can be stored. When a response is received from the user, the response itself and a timestamp are stored in the developer-defined logging database. Similarly, the ECA's response and timestamp are stored upon generation. Routes or methods for other data to be stored can easily be included in the system's database as needed. The *session* database's purpose is to prevent memory leaks and provide a more seamless user experience by saving in-simulation variables. Similar to how the framework is built with *express*, we use an *express-session* component to store any variables associated with individual user interactions. A cookie is associated with each user through express-session, allowing data to be associated with the user's cookie. However, it can be risky to store this data in the server's session storage (e.g., memory leaks from overflow), which is why a session database is implemented. A second connection to a developer-defined SQL-style database table is opened, where session data is stored. Once the session database is connected, all session variables will be routed to and from the database. The primary benefit of these safeguards is towards having longitudinal interventions or in resolving errors, such that ongoing conversations can be re-rendered with ease.

Customizability Notes. We would like to reiterate that the described modules are our attempt at a practical implementation for ECA conversations. ECAs are primarily created through free-to-use tools that require no prior 3D modeling skills, but other ECA models can be integrated[2]. Given that conversations are merely defined through a JSON file, we highly encourage researchers to customize as needed given the system's capabilities. Alternative audio models and forms of response, input, and media can be defined according to one's specifications. For instance, researchers may want to attach a "knowledge base" and retrieval-augmented generation to improve the accuracy of responses, which we have provided a method to do so in the DIY Method using existing APIs. The infrastructure is natively built using SQL databases, but if one can implement an alternative, BABY does not depend on any specific database. Ultimately, the goal is to help researchers build and deploy ECAs effectively with *some* reduced learning curve - like for developers who have not deployed ECAs prior. It is fully possible and encouraged to swap tools for one's desired in-house or commercial replacement.

4 Methods to Employ BABY

This Section 4 will now explain how one can practically make use of the BABY system. The primary way to use BABY is in our described *"Do It Yourself" (DIY) Method*. This method is built on the system architecture described in Section 3 and is intended to be a simple way of replicating our setup. The DIY Method can be adapted by those with *some* technical expertise or specific needs that we have not provided. Briefly mentioned is the BABY Platform, which is an ongoing endeavor to allow users to create ECA conversations without any additional coding (i.e., interface to build a conversation script and intervention properties). This is to support those who may prefer a less technical out-of-the-box option.

4.1 DIY Method

To get started with the platform for local development, one can follow the detailed guide in the *BABY Steps* repository [12]. The repository also contains a document with detailed steps[13] on creating a character, setting up the files, and instantiating a project with BABY. Below is a brief step-by-step on utilizing the *template* folder for quickly launching a new project. The Teaser Figure 1 illustrates a high-level visual of creating an ECA conversation through BABY.

Defining the ECA and Models. The first step is to create an ECA and choose models. As mentioned in the *Character Module* in Section 3.1, BABY uses blendshape-enabled ECAs for lip-syncing and non-verbal behaviors. These can be created[13] and exported quite easily at no cost through tools like ReadyPlayerMe or Avaturn. For models, BABY is natively compatible with commercial models like OpenAI, ElevenLabs, LemonFox, and Google TTS. Simply 'copy' the desired model (e.g., "gpt-4o" or "en-US-Neural2-F") and API key to utilize the desired model.

Create a Conversation Script. Second, one must create their conversation script for their ECA. The documentation [13] and Section 3.2 provide steps to do so, but a complete conversation script must include a *start* and *end* node that specify how the conversation begins and concludes (e.g., link to survey).

Defining the Scenario. With ECA, models, and a script, one can define their scenario accordingly. The *template* includes a *scenario.js* file, where the prior steps and additional dimensions of the scenario can be designated. This can include properties such as the ECA's general non-verbal behaviors, camera view, verbal backchannels, or choice between two provided template interface layouts.

Running the BABY Conversation. By replacing the desired fields in *scenario.js* with one's specifications, the interaction can be fully launched with a simple *run* command. The fields have already been defined such that the *template* folder will run a functional visual demo upon running.

Additional Notes. For those with more specific use cases or other

[12] Full template and repository will be made available upon acceptance. Instead, a condensed, anonymized version of core files for Server, Client, JSON Conversation Scripts, and Character model are attached for validation in this hyperlinked Google Drive (click here).

[13] See hyperlinked Google Doc for condensed, anonymized version of steps (click here).

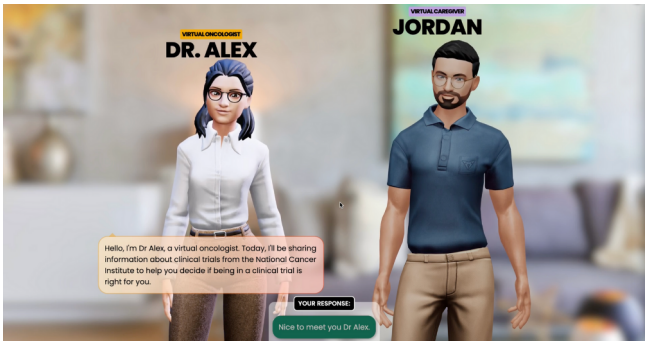

Figure 3: A user study conducted with multi-agents using the BABY system by Redacted et al. ECAs were employed to educate patients about participation in clinical trials.

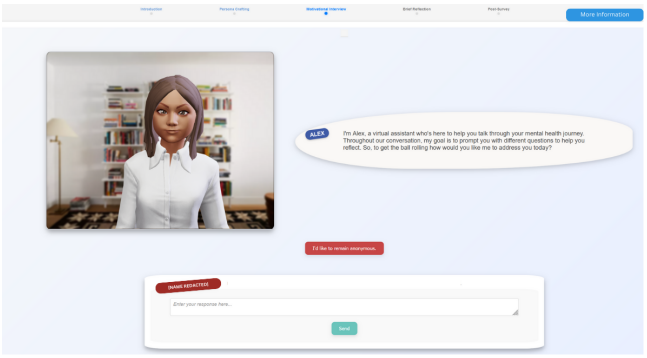

Figure 4: A user study conducted with a virtual interviewer using the BABY system. The ECA was employed to address wellbeing and readiness.

desired functionality, we have also provided a folder of the *core files* that one can fork and build atop. The core files contain only the main client-side and server-side files for functionality utilized in BABY. The aforementioned template interface layouts are two collaboratively defined HTML/CSS pages for BABY, but new HTML and CSS can be defined with a general understanding of web development. The authors note that many applications using BABY in our prior work have begun by treating the *template* folder as a baseline and building the desired components on top.

4.2 BABY Platform

An ongoing development is towards a platform version of BABY, which aims to allow researchers to build ECA conversations without any coding expertise. The platform is built on the same BABY system modules described in Section 3 but is streamlined for ease of use. The platform allows researchers to create entire ECA conversations by defining scenarios, akin to the DIY Method, which are deployed on the platform with plans to allow export for self-hosting. Researchers can upload their models, design open-ended and rule-based conversations, include supplementary files for retrieval augmented generation [21], choose interface layouts, or simply use the platform to build a conversation script JSON that can be exported for the DIY Method. The platform, in beta, can be accessed at the following link: [link redacted for anonymization].

5 Applications of BABY and Novelty

Towards the goals of practicality in design and usage, the section demonstrates applications and novelties of BABY in the form of: (1) user studies deployed to real participants, using the DIY Method, (2) ability to customize and extend BABY evidenced in internal developments, and (3) limitations of BABY's capabilities.

5.1 User Studies Afforded by BABY

Multi-Agent Conversations. Redacted et al. utilized BABY to provide clinical trial education with multiple ECAs [citation redacted].[14] The native BABY system is designed with a singular ECA by default, but the Character Module can be extended to incorporate multiple ECAs that can converse with the user and each other. This capability builds upon prior work that supports conversations with multiple ECAs [1, 4]. Figure 3 illustrates a simulated conversation where a user interacts with two ECAs: a virtual oncologist and a caregiver. The use of more than one agent allows the intervention to offer different types of support to the user (Dr. Alex: information support; Jordan: social support). Having multiple agents can improve an intervention's persuasion [18, 19], especially in health domains [2, 9, 16]. The added engagement and perspective from multi-agents can allow one to better tailor interventions to promote healthy living [2].

Virtual Interview Conversations. The authors have also utilized BABY to conduct virtual interviews, a primary use case from prior work. Prior systems, such as VIP [32] or VPF [28], were designed to be used as either a 'virtual interviewer' (e.g., virtual health counselor) or 'virtual interviewee' (e.g., virtual patients), respectively. Because the Conversation Module relies on a user-defined JSON script (see Section 3.2), BABY can be adapted to handle ECAs for interviews from both perspectives. Figure 4 illustrates a user study deployed to participants with a virtual interviewer conversing through a brief wellness intervention. Conversations can also be further informed by attaching additional knowledge to inform the agent, as in this study and similar [1]. The illustrated user study is a prior conducted user study with its findings to be peer-reviewed.

5.2 Example Customizations of BABY

Beyond the example user studies, we also provide internal customizations of BABY. These capabilities are provided to showcase *opportunities* for leveraging BABY, rather than user-tested studies.

Web Customization. Because BABY is designed exclusively through JavaScript with no game engine, ECAs can be more readily integrated into existing webpages or even as a web extension. This gives the ECA the ability to not only interact with the user and their webpage but to maintain knowledge of information across webpages as well. Figure 5 illustrates a Chrome/Firefox extension version of BABY developed in-house that can read content on the page and highlight specific web elements. Familiarity with 3D environments may allow greater granular control of the character using Three.js. The illustrated character in this example is also

[14] Article to appear

Figure 5: Since BABY is rendered through only web tools, BABY may be hosted within an existing website or as a web extension.

programmed to *follow* along with the user's fixation (e.g., ECA's head follows the mouse or eye-tracked cursor).

Conversation Customization. The modules in BABY allow for characters to have custom traits (e.g., LLMs) or custom behaviors (e.g., Mixamo and conversation script). Figure 6 illustrates a website application of BABY (link redacted for anonymization) that allows users to alter the persona of their ECA in real-time, based upon prior LLM-persona interface design [13]. While prior systems are pre-designed to tailor interactions (e.g., VIP, RA), BABY acts as an 'open slate' where researchers can build applications specific to their use cases, like custom delivery of content and behaviors, by using or extending the methods provided.

5.3 Limitations

BABY's main contribution is in the motivation for practical design and usage. However, there are limitations in its current state that may better quantify our contribution. While instances of user studies with ECA conversations from BABY are illustrated, formal system-level testing (e.g., usability) is outside the scope of the present article and remains future work, like in [1, 12, 32]. Though informal interviews and longitudinal collaborations informed the design, community-driven efforts and collaborations will further shape BABY's design and usage. Towards its design, the character rendering and animating are exclusively built on JavaScript without game engines to help lower the technical and/or computational requirements. However, this may naturally limit the character options, fidelity, and other capabilities. Towards its usage, other prior ECA systems may hold unique advantages that are not feasible in BABY (e.g., other ECA tools like GRETA or mark-up languages). Other usages may be possible but would necessitate *some* customization to support in BABY, which we have aimed to make feasible. For instance, BABY currently supports one form of open-ended and rule-based dialogue. Other related options, like dialogue based on the query's intent (intent matching), could be integrated instead of the native conversation module. Finally, the reliance on existing tools like OpenAI, ElevenLabs, and Google Cloud will incur costs. Applications that must be web deployed are typically done through tools like AWS, which we have successfully used within the examples of Section 5, but will naturally require some knowledge of web

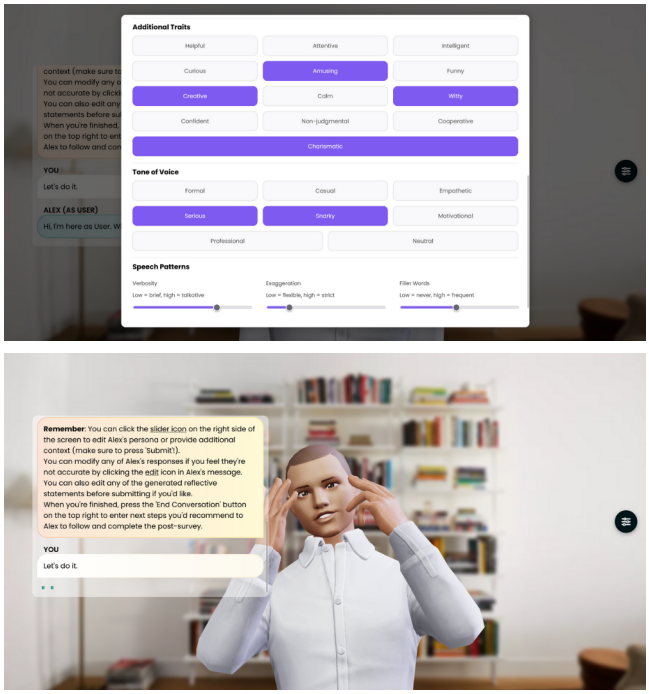

Figure 6: BABY may be extended to support specific needs, such as in adjusting (a) the ECA's persona or (b) behaviors.

integrations and also incur costs. Our design choices were made in an effort to help researchers begin developing ECAs without as many dependencies, using existing libraries and commercial tools aimed at relieving technical burdens. Hence, we have described the primary contribution of this paper as the BABY System architecture and the DIY Method, affording researchers the *opportunity* to adopt alternative solutions independently of the authors.

6 Conclusion

The more people who can build ECAs, the more we can understand, study, and improve the way we design these ECAs. The present article aims to enable these goals through the BABY system. In this article, we discussed the architecture, steps to get started, and applications of building ECA conversations with BABY. We invite interested researchers to try the DIY Method of the system or the beta platform version as described for the quickest results. From there, researchers can consider what is required by their specific application and which of the modules, if any, they would prefer to replace with their desired tools. Future work will focus on learning and extending the system's capabilities, making customizations even more accessible, and further developing the *BABY Platform* for streamlined conversation creations. In conclusion, the BABY system represents a "baby" step toward simplifying the development of ECAs, making it easier for researchers and developers to engage with this ever-growing field. As the system and platform progress, we plan to share our tools and insights that support the creation of more practical and scalable ECAs, helping to drive the next phase of research and development in this area.

References

[1] Tessa Beinema, Daniel Davison, Dennis Reidsma, Oresti Banos, Merijn Bruijnes, Brice Donval, Álvaro Fides Valero, Dirk Heylen, Dennis Hofs, Gerwin Huizing, et al. 2021. Agents united: An open platform for multi-agent conversational systems. In *Proceedings of the 21st ACM international conference on intelligent virtual agents*. 17–24.

[2] Tessa Beinema, Harm op den Akker, Lex van Velsen, and Hermie Hermens. 2021. Tailoring coaching strategies to users' motivation in a multi-agent health coaching application. *Computers in Human Behavior* 121 (2021), 106787.

[3] Timothy Bickmore and Amanda Gruber. 2010. Relational agents in clinical psychiatry. *Harvard review of psychiatry* 18, 2 (2010), 119–130.

[4] Timothy Bickmore, Daniel Schulman, and George Shaw. 2009. DTask and LiteBody: Open source, Standards-Based tools for building Web-Deployed embodied conversational agents. In *Intelligent Virtual Agents: 9th International Conference, IVA 2009 Amsterdam, The Netherlands, September 14-16, 2009 Proceedings 9*. Springer, 425–431.

[5] Timothy Bickmore, Zhe Zhang, Matthew Reichert, Clevanne Julce, and Brian Jack. 2020. Promotion of preconception care among adolescents and young adults by conversational agent. *Journal of Adolescent Health* 67, 2 (2020), S45–S51.

[6] Timothy W Bickmore. 2003. *Relational agents: Effecting change through human-computer relationships*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[7] Timothy W Bickmore, Daniel Schulman, and Candace L Sidner. 2011. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of biomedical informatics* 44, 2 (2011), 183–197.

[8] Juan Cendan and Benjamin Lok. 2012. The use of virtual patients in medical school curricula. *Advances in physiology education* 36, 1 (2012), 48–53.

[9] Harm Op den Akker, Rieks Op den Akker, Tessa Beinema, Oresti Banos, Dirk Heylen, Bjørn Bedsted, Alison Pease, Catherine Pelachaud, Vicente Traver Salcedo, Sofoklis Kyriazakos, et al. 2018. Council of Coaches-A Novel Holistic Behavior Change Coaching Approach. In *4th International Conference on Information and Communication Technologies for Ageing Well and e-Health*. SCITEPRESS-Science and Technology Publications, 219–226.

[10] Nicole R DeTore, Oyenike Balogun-Mwangi, Elizabeth S Eberlin, Katherine N Dokholyan, Albert Rizzo, and Daphne J Holt. 2024. An Artificial Intelligence-Based Virtual Human Avatar Application to Assess the Mental Health of Health Care Professionals: A Validation Study. *Journal of Medical Extended Reality* 1, 1 (2024), 215–226.

[11] Jonathan Gratch, Arno Hartholt, Morteza Dehghani, and Stacy Marsella. 2013. Virtual humans: a new toolkit for cognitive science research. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 35.

[12] Ivan Gris and David G Novick. 2018. Virtual agent interaction framework (VAIF): a tool for rapid development of social agents. (2018).

[13] Juhye Ha, Hyeon Jeon, Daeun Han, Jinwook Seo, and Changhoon Oh. 2024. CloChat: Understanding how people customize, interact, and experience personas in large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–24.

[14] Arno Hartholt, Ed Fast, Zongjian Li, Kevin Kim, Andrew Leeds, and Sharon Mozgai. 2022. Re-architecting the virtual human toolkit: towards an interoperable platform for embodied conversational agent research and development. In *Proceedings of the 22nd ACM International Conference on Intelligent Virtual Agents*. 1–8.

[15] Arno Hartholt, David Traum, Stacy C Marsella, Ari Shapiro, Giota Stratou, Anton Leuski, Louis-Philippe Morency, and Jonathan Gratch. 2013. All together now: Introducing the virtual human toolkit. In *Intelligent Virtual Agents: 13th International Conference, IVA 2013, Edinburgh, UK, August 29-31, 2013. Proceedings 13*. Springer, 368–381.

[16] Marian ZM Hurmuz, Stephanie M Jansen-Kosterink, Harm op den Akker, and Hermie J Hermens. 2020. User experience and potential health effects of a conversational agent-based electronic health intervention: Protocol for an observational cohort study. *JMIR research protocols* 9, 4 (2020), e16641.

[17] Yang Jiang, Xinghua Fu, Jing Wang, Qinling Liu, Xinyu Wang, Peijie Liu, Runchen Fu, Jiangpiao Shi, and Yibo Wu. 2024. Enhancing medical education with chatbots: a randomized controlled trial on standardized patients for colorectal cancer. *BMC Medical Education* 24, 1 (2024), 1511.

[18] Reshmashree B Kantharaju, Dominic De Franco, Alison Pease, and Catherine Pelachaud. 2018. Is two better than one? Effects of multiple agents on user persuasion. In *Proceedings of the 18th international conference on intelligent virtual agents*. 255–262.

[19] Reshmashree B Kantharaju, Alison Pease, Dennis Reidsma, Catherine Pelachaud, Mark Snaith, Merijn Bruijnes, Randy Klaassen, Tessa Beinema, Gerwin Huizing, Donatella Simonetti, et al. 2019. Integrating argumentation with social conversation between multiple virtual coaches. In *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*. 203–205.

[20] Jihyun Lee, Hyungsin Kim, Kwan Hoon Kim, Daeun Jung, Tanisha Jowsey, and Craig S Webster. 2020. Effective virtual patient simulators for medical communication training: a systematic review. *Medical education* 54, 9 (2020), 786–795.

[21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[22] Sok Ying Liaw, Sim Win Ooi, Khairul Dzakirin Bin Rusli, Tang Ching Lau, Wilson Wai San Tam, and Wei Ling Chua. 2020. Nurse-physician communication team training in virtual reality versus live simulations: randomized controlled trial on team communication and teamwork attitudes. *Journal of medical Internet research* 22, 4 (2020), e17279.

[23] Benjamin Lok and Adriana E Foster. 2019. Can virtual humans teach empathy? *Teaching Empathy in Healthcare: Building a New Core Competency* (2019), 143–163.

[24] Gale M Lucas, Albert Rizzo, Jonathan Gratch, Stefan Scherer, Giota Stratou, Jill Boberg, and Louis-Philippe Morency. 2017. Reporting mental health symptoms: breaking down barriers to care with virtual human interviewers. *Frontiers in Robotics and AI* 4 (2017), 51.

[25] Gale M. Lucas, Albert Rizzo, Jonathan Gratch, Stefan Scherer, Giota Stratou, Jill Boberg, and Louis Philippe Morency. 2017. Reporting mental health symptoms: Breaking down barriers to care with virtual human interviewers. *Frontiers Robotics AI* 4, OCT (2017), 1–9. doi:10.3389/frobt.2017.00051

[26] Samuel Mascarenhas, Manuel Guimarães, Rui Prada, Pedro A Santos, João Dias, and Ana Paiva. 2022. Fatima toolkit: Toward an accessible tool for the development of socio-emotional agents. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 12, 1 (2022), 1–30.

[27] Albert Rizzo, Belinda Lange, John G Buckwalter, Eric Forbell, Julia Kim, Kenji Sagae, Josh Williams, JoAnn Difede, Barbara O Rothbaum, Greg Reger, et al. 2011. SimCoach: an intelligent virtual human system for providing healthcare information and support. (2011).

[28] Brent Rossen and Benjamin Lok. 2012. A crowdsourcing method to develop virtual human conversational agents. *International Journal of Human-Computer Studies* 70, 4 (2012), 301–319.

[29] Antoinette Schoenthaler, Glenn Albright, Judith Hibbard, Ron Goldman, et al. 2017. Simulated conversations with virtual humans to improve patient-provider communication and reduce unnecessary prescriptions for antibiotics: a repeated measure pilot study. *JMIR medical education* 3, 1 (2017), e6305.

[30] Heng Yao, Alexandre Gomes de Siqueira, Megan L Rogers, Sarah Bloch-Elkouby, Olivia Lawrence, Giuseppe Sarli, Adriana Foster, Serge A Mitelman, Igor Galynker, and Benjamin Lok. 2024. The impact of scaffolded and non-scaffolded suicidal virtual human interaction training on clinician emotional self-awareness, empathic communication, and clinical efficacy. *BMC medical education* 24, 1 (2024), 413.

[31] Mohan Zalake. 2022. *Designing Effective Virtual Human Conversations to Promote Healthy Behaviors*. Ph. D. Dissertation. University of Florida.

[32] Mohan Zalake, Alexandre Gomes de Siqueira, Krishna Vaddiparti, Pavlo Antonenko, Felix Hamza-Lup, and Benjamin Lok. 2020. Towards rapid development of conversational virtual humans using Web3D technologies. In *Proceedings of the 25th International Conference on 3D Web Technology*. 1–2.