JEDAI.Ed: An Interactive Explainable AI Platform for Outreach with Robotics Programming

by

Daksh Dobhal

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2024 by the
Graduate Supervisory Committee:

Siddharth Srivastava, Chair
Nakul Gopalan
Hasti Seifi

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

While the growing prevalence of robots in industry and daily life necessitates knowing how to operate them safely and effectively, the steep learning curve of programming languages and formal AI education is a barrier for most beginner users. This thesis presents an interactive platform which leverages a block based programming interface with natural language instructions to teach robotics programming to novice users. An integrated robot simulator allows users to view the execution of their high-level plan, with the hierarchical low level planning abstracted away from them. Users are provided human-understandable explanations of their planning failures and hints using LLMs to enhance the learning process. The results obtained from a user study conducted with students having minimal programming experience show that JEDAI-Ed is successful in teaching robotic planning to users, as well as increasing their curiosity about AI in general.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

As robots become more common in households, public areas and workplaces, they are being used to perform an increasingly wide range of tasks - from vacuuming floors to delivering food to manufacturing automobile parts. Such task-completing or programmable robots, however, need human input in the form of well-defined tasks that they have to accomplish. Additionally, programmable robots also require instructions on how to perform sub-tasks in a sequential manner to achieve the larger goal. In order for such robots to become widely used and adopted, programming them must not become a barrier to entry for the general population. It therefore become infeasible for robotics programming to be carried out in formal programming languages and to be conditional on the users having education in Computer Science (CS) and AI. An interface between the user and the low-level implementation instructions to a programmable robot is needed. Considerable effort has gone into developing such interfaces as CoStar (Paxton *et al.* (2016)) which bridge the gap between the novice user and the robotic system. The interface should accept instructions in a format which makes it possible for a novice user to provide instructions while abstracting the low-level implementation away from them. Users also cannot be assumed to be familiar with the capabilities of the robot they are working with, nor can they be expected to understand the limitations under which it performs tasks. As an instructional aide, the interface must estimate the users understanding of the robot and its environments, and provide feedback to clear their misconceptions.

While CoStar (Paxton *et al.* (2016)) does provide a user-understandable interface, there is no feedback from the system from the system to the user. In particular, there is a need for the system to be able to detect errors in the users instructions and explain the errors in order to correct their understanding. Towards this end, Shah *et al.* (2022) developed JEDAI, a block-based interface which aims to teach novice users robotic programming. JEDAI provides users with a set of planning tasks to solve, and provides explanations in case of errors in the users plans. Although it is a novice-friendly interface, JEDAI has certain drawbacks that limit its use as an educational platform. For an educational tool to be engaging and effective, it must adapt to individual learner needs. JEDAI, however, provides all users with the same set of tasks to solve. There is no functionality of adapting to a learners level of understanding as they progress through the tasks and provide them with easier or harder problems as needed. Secondly, the user-interface consists of multiple screens and not a single integrated display. Requiring the users to navigate between multiple displays reduces engagement and induces screen fatigue. Finally, the feedback that JEDAI gives to users in case of errors in their plans is a hand coded text template which simply lists out the reasons a particular instruction could not be executed. A more readable and user friendly user feedback mechanism is required to make the explanations understandable to the users.

This thesis aims to further the goals of JEDAI by developing JEDAI.Ed, a new platform that improves upon and addresses key limitations faced by its predecessor. JEDAI.Ed contains a wide array of improvements to the JEDAI platform, and introduces many novel features as well, which are briefly mentioned below.

§1 Adaptive Curriculum Generation

An effective learning experience requires the curriculum to adapt to the individual users level of understanding so as to not overwhelm them with problems too difficult for them to solve, while also being challenging enough to keep them engaged. In order to achieve such an effect with JEDAI, an instructor would need to repeatedly hand-code problems for each user, making it impossible for the platform to be used at scale. JEDAI.Ed eliminates this problem by adapting to the users understanding and providing them problems of increasing difficulty. It also is able to automatically detect if a user is performing poorly, and reduces difficulty as needed, all without needing any human intervention.

§2 Natural Language Explanations and Hints

As users solve tasks, they are given messages explaining why their plan failed and hints about how to solve the task. These explanations must be intuitive and easy to understand for them to be effective in enhancing the learner's understanding of the robots capabilities and limitations under which is operates. JEDAI provides explanations as an instruction that could not be executed and the list of conditions that must be true for the action to execute. While logically sound, these explanations are not intuitive or readable for a beginner having little or no exposure to AI. By using Large Language Models (LLMs) prompted with additional contextual information about the environment, JEDAI.Ed produces more informative and readable translation of the explanation messages. To create a smoother self-learning journey, users can also avail hints about which action they should take next to solve the task.

§3 Improved User Interface

The UI for the JEDAI platform consisted of multiple static webpages. All feedback messages were displayed simultaneously on the screen which could overwhelm the user with information. A separate browser window would contain the robot simulator, with users needing to go back and forth between the two. JEDAI.Ed, in keeping with our goal of developing a tool which learners can utilize with minimal supervision, has an intuitive and integrated single window display. Users can access summary of messages on the screen with more detailed information available as drowpdown lists. The overall UI is more compact and easier to work with.

§4 Support for Stochastic Actions

JEDAI supports environments with deterministic actions only. In order to move towards a more general purpose AI education platform, this work also introduces basic support for allowing users to plan with actions that might fail. We supplement the blocks interface with looping blocks, allowing the user to repeatedly try actions. This is supplemented by functionality to enable such loop-containing plans to run on low-level planners for the user to be able to view the plan execution in the simulator.

The rest of the thesis begins with a reproduction of a collaborative work based on JEDAI.Ed, which is currently under submission and covers the JEDAI.Ed platform and the user study in detail. This is followed by a section on supporting stochastic actions, followed by a section summarizing the thesis and possible future research directions, which concludes this thesis.

Chapter 2

USING EXPLAINABLE AI AND HIERARCHICAL PLANNING FOR OUTREACH WITH ROBOTS

This chapter introduces a submitted version of the work that was performed by me in collaboration with co-authors and pending acceptance at an academic conference. Co-author permissions to reproduce the content can be found in the Appendix D.

2.1 Abstract

Understanding how robots plan and execute tasks is crucial in today's world, where they are becoming more prevalent in our daily lives. However, teaching non-experts the complexities of robot planning can be challenging. This work presents an open-source platform that simplifies the process using a visual interface that completely abstracts the complex internals of hierarchical planning that robots use for performing task and motion planning. Using the principles developed in the field of explainable AI, this intuitive platform enables users to create plans for robots to complete tasks, and provides helpful hints and natural language explanations for errors. The platform also has a built-in simulator to demonstrate how robots execute submitted plans. This platform's efficacy was tested in a user study on university students with little to no computer science background. Our results show that this platform is highly effective in teaching novice users the intuitions of robot task planning.

2.2 Introduction

Recent advances in Artificial Intelligence (AI) have enabled the deployment of *programmable* AI robots that can assist humans in a myriad of tasks. However, such advances will have limited utility and scope if users need to have advanced technical knowledge to use them safely and productively. For instance, a mechanical arm robot that can assist humans in assembling different types of components will have limited utility if the operator is unable to understand what it can do, and cannot effectively re-task it to help with new designs.

This paper aims to develop new methods that will allow educators and AI system manufacturers to introduce users to AI systems on the fly, i.e., without requiring advanced degrees in CS/AI as prerequisites. These methods allow for introducing robotics programming to novices.

We use the existing JEDAI system (Shah *et al.*, 2022) as the platform for implementing and evaluating these methods. While the core JEDAI system provides a good foundation for development, it has not been developed or evaluated with the components necessary for introductory AI education. E.g., it indirectly requires the users to have some knowledge of robot simulators to operate, does not help educators with designing curricula, and only generates boilerplate, hand-coded, domain-specific text for explanations.

Our contribution Our work addresses some of the key technical challenges in developing educational AI systems for facilitating robotics programming. We accomplish our overall objective by improving and extending JEDAI to introduce JEDAI.Ed, a web application that abstracts away the intricacies of robotics programming and exposes the user to an easy-to-use interface to the robot. JEDAI.Ed incorporates several new features compared to its predecessor that enable its use in educational settings and beyond. Firstly, JEDAI.Ed provides adaptive curriculum design tools that can automatically generate problems. Our system identifies multiple causes of failure (JEDAI is limited to 1) and explains all of them and can also provide hints to users. Finally, JEDAI.Ed utilizes large language models (LLMs) not to discover information but to express factual information and justifications computed using well-defined reasoning processes thereby ensuring the reliability of information being provided.

We showcase the usefulness of JEDAI.Ed through a user study designed to assess and evaluate its utility and compare it to JEDAI. Our results show that JEDAI.Ed makes robots easy to use and piques curiosity about AI systems. Furthermore, JEDAI.Ed results in a 20% improvement in solution times and significantly greater positive sentiment compared to JEDAI.

2.3 Robotics Programming

In this section, we give a background of key concepts that allow users to program robots for accomplishing tasks.

Running example Consider a robot that is deployed at a coffee shop to help with its day-to-day operations. Depending upon the day's priorities, the owner may want to program the robot to assist with different tasks such as delivering coffee to customers or washing the cups, etc. To effectively assist the owner, the robot (a) must be equipped with a gripper, wheels, and other hardware that enable it to accomplish the required tasks, and (b) must be able to be “given" tasks (or instructions) by the owner and autonomously perform them, requiring the owner's intervention only when it cannot complete the task or execute a specific instruction.

Planning Robots (and humans) often accomplish tasks by computing a fixed sequence of instructions and then executing them sequentially. These sequences are known as *plans*, *planning* is the process of computing such plans, and algorithms that do planning are called *planners*. Planners take an input task and instruction set and output a plan (consisting of instructions from the instruction set) that solves the task. A *valid* plan for a task is a sequence of instructions that are consistent w.r.t. the initial state of the task and the semantics of the instructions.

Robot instructions and motion planning Robots cannot execute plans with arbitrary sets of instructions. They require a specific type of *low-level* plan known as a *motion plan*. This plan specifies a sequence of movements for each joint of the robot and is obtained using *motion planning*. E.g., the Fetch robot in Fig. 1e has an arm with 8 joints: $\theta_1, \ldots, \theta_8$. A motion plan, $\langle [\theta_1^1, \ldots, \theta_8^1], \ldots, [\theta_1^n, \ldots, \theta_8^n] \rangle$, that uses the robot to accomplish an example task of picking up a coffee cup from the counter would contain a sequence of *low-level* instructions $[\theta_1^i, \ldots, \theta_k^i]$ that contain numeric values, $\theta_k \in \mathbb{R}$, for all of its joints. Computing such low-level instructions needs robot-specific knowledge and requires complex algebraic arithmetic to compute a motion plan that (a) does not cause collisions, (b) does not damage the robot, (c) provides smooth (and safe) motion. These constraints make motion planning quite difficult for humans.

Human instructions and plans Contrary to robots, humans typically accomplish tasks by following instructions at a higher level of abstraction than robots. E.g., to accomplish the same task described in the preceding paragraph, a human often computes a *high-level* plan, $\langle$`Go to the counter`, `Pick up the coffee cup`$\rangle$, consisting of *high-level* instructions. Humans can find (and execute) high-level plans for complex tasks fairly easily, however, robots can not use such plans directly to accomplish tasks.

Hierarchical planning Given the difficulty of motion planning, it is easy to see that programmable robots must accept high-level instructions in order to be usable by humans. These instructions need to be compiled into motion plans for robots to execute and this is accomplished using *hierarchical planning*. Hierarchical planners work by using input (or computing) high-level plans to construct a motion plan for each high-level instruction by transcribing them to motion planning problems for use by a motion planner. In this work, we focus on human-in-the-loop (HITL) robot programming where high-level plans are provided by a human and a hierarchical planner converts such plans into a sequence of motion plans that the robot can execute. For clarity, we refer to motion plans as low-level plans in the rest of the paper.

This tiered approach to HITL robotics programming introduces some new hurdles that need to be addressed. One key challenge is that high-level plans might not be successfully compiled into low-level plans. E.g., a high-level plan ⟨`Pick up the coffee cup`⟩ cannot be compiled into a low-level plan for a single-arm robot if it is already holding something else. When such failures occur, it is imperative that the robot appropriately informs the user of the failure in high-level terms that the user can easily understand. Explaining why a failure occurred can allow a user to correct (or modify) the high-level instructions so that the desired behavior can be achieved. E.g., an explanation of the form "*I (the robot) cannot pick up the coffee cup because I am currently holding a water bottle*" allows the user to (a) identify why the robot could not accomplish the task, and (b) modify their instructions so that the robot can accomplish it.

Desiderata	JEDAI.Ed	JEDAI
Open source	✔	✔
Minimal system requirements	✔	✔
Highly customizable	✔	✔
Integrated simulation	✔	✔
Intuitive user interface	✔	✗
Curriculum generation	✔	✗
Intuitive Explanations	✔	✗
Hints	✔	✗

Table 1. A comparison of some of the features of JEDAI.Ed compared to JEDAI. A description of the desiderata and an extensive comparison is available in the supplementary material.

2.4 The JEDAI.Ed Platform

We aim to develop a platform that makes robotics programming accessible to a wide spectrum of users and use cases. As a result, we have taken several design considerations (detailed in the supplement) to develop JEDAI.Ed, an open source[1] pedagogical tool that brings robotics programming into the hands of novice users. JEDAI.Ed is usable by educators seeking to teach classes on AI, by hobbyists who are interested in robotics, by professionals for on-boarding processes in industrial settings, and many others. We compare JEDAI.Ed with JEDAI w.r.t. some of the desiderata in Table 1.

The next section discusses JEDAI.Ed's features that make it an ideal pedagogical platform for robotics programming.

[1]https://github.com/AAIR-lab/AAIR-JEDAI.

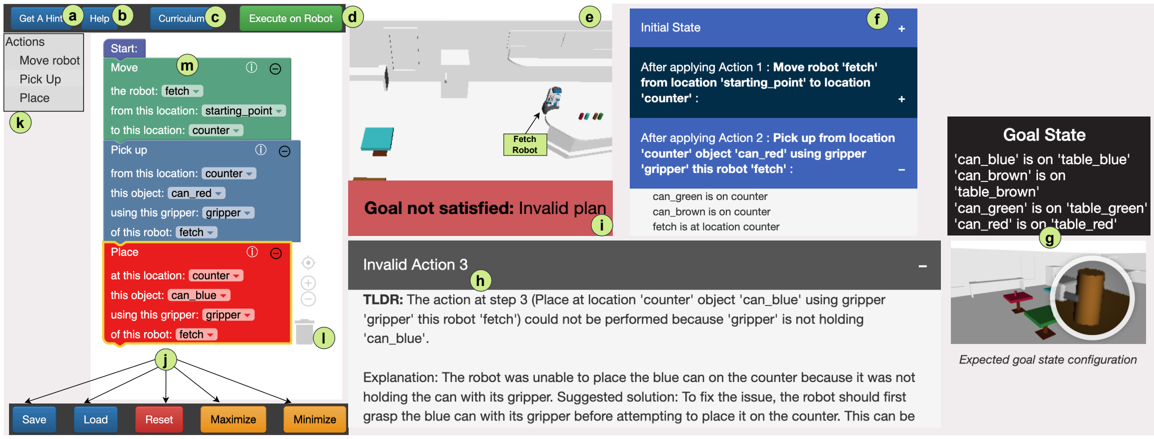


Figure 1. A screenshot (best viewed in color) of the JEDAI.Ed user interface (UI) (zoomed in and enhanced for clarity). The annotated circles describe the different sections of the UI (described in Sec. 2.4.1). Unmodified screenshots are available in the supplementary material.

2.4.1 System Overview

JEDAI.Ed is modular in design allowing for easy customization (discussed in supplement). We now provide a high-level overview of JEDAI.Ed's features followed by an example of how it can be used by an educator (student) to introduce (learn) robotics programming for the running example described earlier in this paper. We then provide a detailed technical description of the core components of JEDAI.Ed in the next section (Sec. 2.4.2).

§1 Easy deployment and installation JEDAI.Ed is a web app using the client-server design. The client, which is the interface that users interact with, can run on any JavaScript (JS) enabled web browser (most modern browsers). There are no other system requirements or installation steps for using the client making it readily usable on a large number of devices with differing hardware. The core back-end interface (described in Sec. 2.4.2) is written using Python and JS and can be deployed on the same machine or hosted on a server to handle thousands of users. Finally, delivering system updates is trivial and only requires users to reload the web page.

§2 Curriculum design tools JEDAI.Ed provides curriculum design tools that can lift the burden of creating engaging tasks from educators. First, JEDAI.Ed comes preloaded with several environments and robots that are widely used in AI coursework today. JEDAI.Ed includes an adaptive curriculum algorithm that can use domain descriptions to automatically create tasks for users. The algorithm (discussed in Sec. 2.4.2) actively tracks the performance of users on tasks and adapts the difficulty of generated tasks accordingly.

§3 Intuitive user interface (UI) Fig. 1 presents a screenshot of the JEDAI.Ed user interface. Our UI is carefully designed to enable novices to get acquainted with JEDAI.Ed using little to no supervision. JEDAI.Ed reduces scroll and navigation fatigue by presenting all pertinent information in a single window with hover-on pop-ups and collapsible text blocks. A *Help* button (Fig. 1b) on the screen allows the user to quickly familiarize themselves with the system at any point. To minimize clutter and provide a clean interface, JEDAI.Ed's UI is divided into several sections that display different kinds of information. Relevant information is conveyed to the user using several different modalities: text, visual, and audio cues allowing for rich and dynamic feedback. We include many convenience functions (Fig. 1$\{j, l\}$) to save, load plans etc.

§4 Block-based programming JEDAI.Ed uses Blockly (Google, 2018), a block-based language that allows specifying high-level instructions or plans using blocks. Such languages (e.g., Scratch (Resnick *et al.*, 2009)) have been demonstrated to be successful in allowing novice users to write programs. A plan in Blockly is intuitively represented as a sequence of linearly connected blocks (Fig. 1m). Different blocks represent different high-level instructions and automatically populated choices allow the user to easily specify the precise semantics of a high-level instruction. E.g., two high-level instructions, '`Go to the red table`', and '`Go to the counter`' are represented using the same Blockly block '`Go to the <location>`', where the *parameter* `location` is instantiated by the user with the appropriate objects. These blocks describe the high-level instruction in a user-interpretable fashion.

§5 Explanations and hints It is well-known that iteration and improvement are part of the learning process and learning from failures can be expected in an educational setting (Jackson *et al.*, 2022). JEDAI.Ed uses advances in explainable AI to take on the role of a teacher and provides feedback to the user using automatically generated explanations and hints that allow them to (a) learn why their plans are failing, (b) better understand the robot's limitations and capabilities, and (c) fix their plans so that they can successfully use the robot to accomplish their tasks.

§6 Hierarchical planner JEDAI.Ed automatically converts user-provided high-level plans into low-level plans using ATAM (Shah *et al.*, 2020), a state-of-art hierarchical planner.

§7 Integrated simulator JEDAI.Ed uses an integrated simulator, OpenRAVE (Diankov, 2010), that uses robot specifications to provide real-time executions of user-provided high-level plans in an environment. Simulated executions of low-level instructions in JEDAI.Ed are close approximations of executions in a real-world environment. This feature allows JEDAI.Ed to be used for learning how to operate a robot even if a physical robot is not available or whose cost is prohibitive.

§8 Natural language feedback JEDAI.Ed utilizes natural language templates and uses large language models (LLMs) to provide detailed feedback of explanations, hints, and other status messages in a user-interpretable format.

Example: Using JEDAI.Ed for a programmable single-arm, mobile robot We now describe a typical session of JEDAI.Ed that introduces the functionalities of a mobile manipulator like Fetch (Fig. 1*e*) that is intended to be used in a coffee shop based on the running example (Sec. 2.3).

First, the educator installs the JEDAI.Ed system on a machine (§1). Next, the educator uses the curriculum design tools to design an appropriate curriculum for the intended audience (§2). The educator can use one of the bundled environments or invokes the automatic curriculum generator. The educator then generates (or selects preset) tasks for the student to accomplish by programming the robot. The student accesses JEDAI.Ed on a web browser and begins learning (§3).

The UI presents the user with the necessary information such as the task description and goal (Fig. 1*g*), available instruction set (Fig. 1*k*), and a simulator window (Fig. 1*e*). The goal is provided both in textual as well as visual descriptions along with a magnifier to view finer details of the image. A *Help* button (Fig. 1*b*) provides useful descriptions about the interface and is available to the user at all times.

The user then uses the instruction sets along with intuitive knowledge or after reading the hoverable descriptions to create a plan of high-level instructions by dragging-and-dropping Blockly blocks (§4, Fig. 1*l*) and connecting them to the *Start* block. An audible click lets the user know that the block snapped to another block.

Every connected block is checked for validity in real-time and explanations (§5) are provided if the user's current plan contains any invalid actions (Fig. 1*h*). E.g., the explanation shown in Fig. 1*h* explains that the instruction '`Place the blue can at the counter using gripper of the Fetch`' failed because the robot was not holding the blue can. The user may also check the result of their current plan in the state display area (Fig. 1*f*). If the user has difficulty in providing high-level instructions, the user may request a hint (§5, Fig. 1a) that returns a high-level instruction as a pop-up.

Once a valid high-level plan (irrespective of whether it accomplishes the goal or not) is achieved (Fig. 1g), the "Execute on Robot" (Fig. 1d) button is activated and the user may submit their plan to be executed on the robot. The planning process (§7) and real-time execution of the low-level plan are streamed by the simulator (Fig. 1c).

2.4.2 Technical Description

JEDAI.Ed consists of five core components, (a) the user interface, (b) the curriculum design module, (c) the user assistance module, (d) the low-level module, and (e) the natural language module. Of these components, the user interface forms the front-end and is required to run on the user's web browser. All other components constitute the back-end and can be run on a different computer. The front-end communicates with the back-end via a networking interface.

Fig. 2 shows the architecture of JEDAI.Ed and the interaction between its various components. The example in the preceding section described the front-end and we expand upon the internals of the back-end in the remainder of this section.

Curriculum design module JEDAI.Ed provides several default environments like Tower of Hanoi, Dominoes, etc. that are widely used in AI coursework. New environments can be easily added by an educator without employing significant effort by providing annotated high-level instruction sets in the PDDL language (McDermott *et al.*, 1998) and compatible simulator environment files. Blockly blocks that represent these high-level instruction sets are automatically generated for them. Similarly, tasks can be automatically generated as well using the techniques described below.

Adaptive curriculum generation: JEDAI.Ed can automatically generate a curriculum tailored to a user's expertise. We do this by keeping track of the actions that a user has had difficulty with. Difficult actions are those where the user is likely to make a mistake and require an explanation or hint. We monitor these statistics as the user is interacting with JEDAI.Ed. The algorithm then generates simpler tasks that require applications of difficult actions to reach a goal. The key intuition is that the user can learn these difficult actions faster if tasks do not contain other actions thereby reducing the overall cognitive load (Moos and Pitton, 2014). We use breadth-first search (BFS) (Russell and Norvig, 2020) to generate the tasks using the high-level instruction set. Similarly, if a user is proficient at certain actions, their cost is set to be very high thus dissuading BFS from generating tasks that involve them. The generated task is then verified to be executable by the robot using the hierarchical planner. More details about the algorithm are available in the supplement.

Problem generation: Similar to JEDAI, JEDAI.Ed can also generate random tasks requiring a minimum number k of high-level instructions to accomplish. Unlike the adaptive curriculum algorithm, we use vanilla BFS to generate such tasks. These tools make adding new environments easier for educators since they need not design new problems and only need to provide the domain description and environment files.

User assistance module JEDAI.Ed assists the user by using SOTA tools for generating explanations and hints. This module also communicates with the curriculum design module by providing statistics on user-submitted plans.

Explanations: Our system uses HELM (Sreedharan *et al.*, 2018) and VAL (Howey *et al.*, 2004) for generating explanations whenever failures occur in using the user-submitted high-level plan. Once a user connects any block, the current plan is routed through these components to first identify whether the plan is valid. An invalid plan

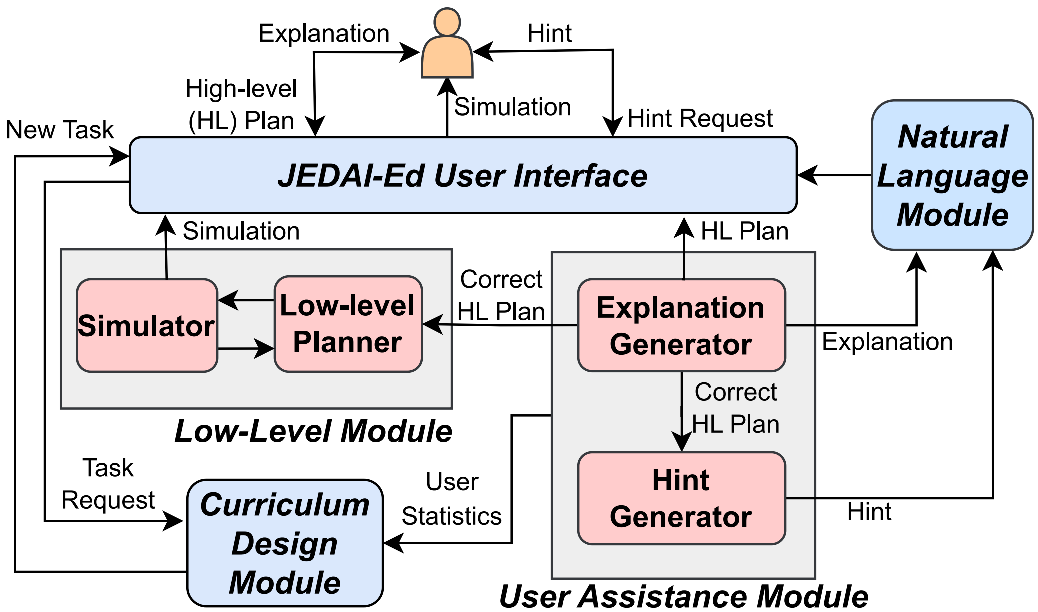


Figure 2. The JEDAI.Ed architecture (described in Sec. 2.4.2).

is passed to HELM and VAL to generate formal explanations. JEDAI.Ed informs the user of this failure by appropriately highlighting the invalid blocks as well as a status message. This status message converts the formal explanation to NL using domain-independent NL templates and LLMs. JEDAI.Ed can explain failures in two different actions simultaneously making user plan correction easier and faster. JEDAI can only explain one failure and requires users to submit plans rather than inspecting blocks in real-time like JEDAI.Ed.

Hints: JEDAI.Ed comes equipped with FF (Hoffmann, 2001), a *fast* high-level planner. FF can be used to generate the next high-level instruction needed to accomplish the task. This is presented as a user-interpretable hint to the user. Since high-level planning is a hard problem (Bylander, 1994), hinting comes preconfigured with a timeout and displays a status message if hints cannot be computed within the time limit.

Low-level module The low-level module consists of a simulator and a hierarchical planner. The hierarchical planner interacts with the simulator to compute a low-level plan (motion plan) for each instruction in a valid high-level plan. The simulator works in tandem with the planner to check for collisions as well as visually show the user the planning process. It also produces a real-time simulation of the robot executing the high-level plan once a low-level plan is computed.

Natural language (NL) module This module processes messages from all components, converting them to a human-readable message via hand-coded NL templates and/or LLMs. We use well-defined reasoning processes to avoid hallucinations in LLMs. We do this by utilizing LLMs primarily as a translator and not as a reasoner. For example, the explanation from an explainer is specified in formal syntax using the instruction set. We then use this formal syntax in a prompt that asks the LLM to convert it to natural language.

Since LLMs are few-shot learners (Brown *et al.*, 2020), we fine-tuned a GPT-3.5 Turbo LLM (OpenAI, 2022) using a few hand-coded examples to provide detailed descriptions of the generated explanations and hints. We chose GPT-3.5 since it is not cost-prohibitive and has been demonstrated to perform well in many NL translation tasks (Kalyan, 2024). A detailed description of our prompts, NL templates, and fine-tuning process is available in the supplementary material.

2.5 Empirical Evaluation

We developed JEDAI.Ed to expose novice users to AI and robotics. We conducted a user study to evaluate if JEDAI.Ed achieves the goal by evaluating the following hypotheses:

H1 (Increased curiosity): JEDAI.Ed increases the curiosity of users to learn more about robotics and AI.

H2 (Easier programming): JEDAI.Ed makes it easy for users to provide instructions to robots.

H3 (Improved understanding): JEDAI.Ed improves user understanding w.r.t. the limitations/capabilities of a robot.

H4 (Helpful explanations): JEDAI.Ed's provided explanations help users understand (and fix) errors in their plans.

H5 (Intuitive UI): JEDAI.Ed's UI is intuitive and easy to use requiring little to no study facilitator intervention.

H6 (Programming confidence): JEDAI.Ed increases users confidence in instructing robots to accomplish tasks.

H7 (Faster solving): JEDAI.Ed allows users to solve tasks faster than JEDAI.

To evaluate the validity of these hypotheses, we designed a user study for evaluating JEDAI.Ed and comparing it with JEDAI. We present the study methodology below.

2.5.1 User Study Setup

We hired 43 university students with no background in computer science as participants for an IRB-approved user study. We discarded 1 incomplete/invalid response, resulting in a sample size of 42. 23 of these were from a non-STEM background. Only 1 applicant had experience with data structures. We divided the participants into two control groups. The first (second) control group was assigned the JEDAI.Ed (JEDAI) system for use in the study. The study lasted 45 minutes, was conducted in-person and had four phases:

Pre-survey phase (8 min): Participants were presented with an introductory video about AI. Next, to acquire a detailed understanding of the particpant's background, interests in AI, and their perspective on JEDAI.Ed, we employed a pre-survey questionnaire. This survey is aimed at assessing the participant's level of awareness and curiosity regarding AI and their level of engagement with AI technologies.

Training phase (12 min): This phase was intended to get users familiarized with the system and tasks. Communication with the study facilitator was allowed. Participants were presented sequentially with three tasks of the *Coffee shop* environment (Sec. 2.3) each of which involved utilizing a Fetch robot to deliver cans to tables. JEDAI.Ed used the adaptive curriculum algorithm to generate tasks. JEDAI randomly generated tasks. We ensured all generated training tasks needed 50% fewer instructions to accomplish than the test task.

Test phase (12 min): The participants solved a test task during this phase. The test task was much harder than the training tasks and required users to deliver multiple cans (optimally using 16 high-level instructions). No communication with the study facilitator was allowed during this phase. The participants were then asked to complete a post-survey questionnaire whose questions were designed to obtain the participant's opinion on the platform they interacted with and to determine if their interest and curiosity had increased post-use. We also collected system logs for analytics data.

Sentiment change phase (13 min): This phase is intended to analyze the sentiment change after interacting with both JEDAI.Ed and JEDAI. In this phase, participants that interacted with JEDAI.Ed (JEDAI) in the previous phase were asked to interact freely with JEDAI (JEDAI.Ed). There was no test task. They were once again asked to answer a post-survey questionnaire. This questionnaire was the same as that of the test phase but they could not see the previous responses.

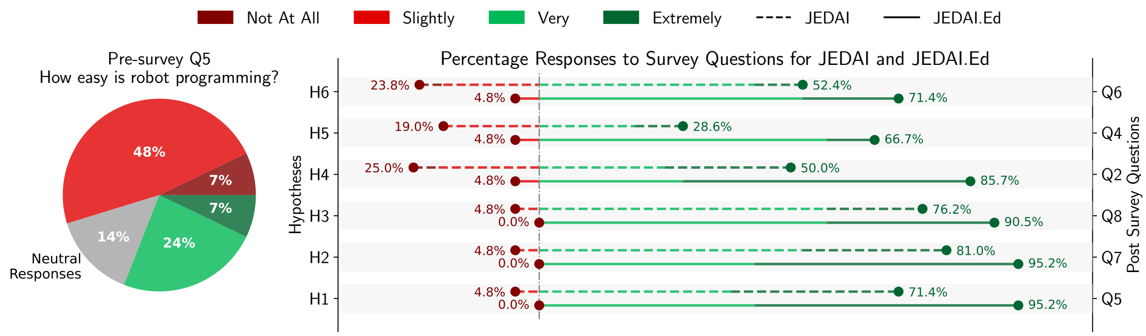


Figure 3. Results from our user study with two control groups ($n = 42$) split evenly between JEDAI.Ed and JEDAI. The x-axis plots the responses as a percentage. Green (red) bars to the right (left) indicate positive (negative) sentiment. Stem annotations indicate the total positive (negative) sentiment. The left y-axis specifies the target hypothesis. The right y-axis specifies the question that assesses its validity.

Questionnaire methodology All responses to the questions used the Likert Scale (Likert, 1932). This approach offers the benefit of providing a more intricate depiction compared to straightforward binary responses when seeking qualitative opinions about their interaction experience with a platform.

Hypothesis testing Our Likert scale data was converted to numbers from 0 to 4 with 0 (4) corresponding to the most negative (positive) response and 2 being neutral. We used the p-value obtained by using the one-sample t-test (Ross and Willson, 2017) to test the statistical significance of the data. Within a control group, we assumed the data to be two-tailed for the questions used to validate the hypotheses and used the hypothesis of no difference, i.e., $\mu_0^1 = 2$ (balanced Likert scales), as the null hypothesis. Across control groups, we used the single-tailed two-sample t-test using the null hypothesis $\mu_0^2 = 0$ to check for statistical significance between user responses to JEDAI.Ed and JEDAI.

Hypothesis	Post Survey Question used for Testing Hypothesis	Statistical Significance			
		μ	σ	$t(n)$	p
H1: Increased curiosity	**Q5:** As compared to before participating, how much has your curiosity increased to learn more about AI systems and robots?	3.48	0.60	11.24	< 0.001
H2: Easier programming	**Q7:** Do you agree that the JEDAI.Ed system made it easier for you to provide instructions to a robot for performing tasks?	3.48	0.60	11.24	< 0.001
H3: Improved understanding	**Q8:** Do you agree that JEDAI.Ed helps improve the understanding of the robot's limitations and capabilities?	3.24	0.62	9.08	< 0.001
H4: Helpful explanations	**Q2:** How helpful were the explanations that were given for the cause of an error?	3.38	0.86	7.32	< 0.001
H5: Intuitive UI	**Q4:** How intuitive was the interface?	2.71	0.72	4.56	< 0.001
H6: Programming confidence	**Q6:** How well do you think you now understand how one can use an AI system to make a plan for a robot to perform a task?	2.86	0.79	4.95	< 0.001

Table 2. JEDAI.Ed user study results ($n = 42$) used to validate our hypotheses. The table provides a short description of the target hypothesis, the corresponding questions used to validate it, and the one-sample t-test results. Comprehensive statistical data is available in the supplement.

2.5.2 Study Results

Fig. 3 and Table. 2 show our results, the survey questions used to analyze the hypotheses and data from the statistical tests. For the one (two)-sample t-test, all data was statistically significant except for JEDAI Q2, Q4, and Q6 (Q8). Moreover, JEDAI.Ed's $\mu^1 \geq 2.7$ for all questions showcases an improved, positive experience. We analyze our results below.

H1 (Increasing curiosity): Fig. 3 shows that after interacting with JEDAI.Ed, user curiosity is 95% positive. This is far greater than JEDAI, whose positive user sentiment is 71%.

H2, H3, and H6: Our pre-survey results (Fig. 3, pie) show that before using JEDAI.Ed, 55% of users believed that robot programming was not easy.

H2 (Easier Programming): 95% of users thought that JEDAI.Ed made it easier to program robots. In contrast, JEDAI only managed to increase positive sentiment to 71%.

H3 (Improved understanding), H6 (Programming confidence): After interacting with JEDAI.Ed, 90% of users think that they better understand the robot's capabilities and limitations and 71% of users were confident that they could instruct robots to accomplish tasks. We did not reject the null hypothesis in the two-sample t-test for H6 (Q8), implying that JEDAI also improves user understanding similarly to JEDAI.Ed. However, JEDAI's responses are not statistically significant $\mu = 2.38$ for Q3 (Q6) implying that JEDAI fails to significantly improve user confidence in robot programming.

H4 (Helpful explanations): Users were extremely positive in their feedback w.r.t. JEDAI.Ed provided explanations. As compared to JEDAI, JEDAI.Ed provides both brief and LLM-based descriptive explanations that better help describe why an instruction is failing. Thus, JEDAI explanations were rated significantly lower and also had a 25% negative sentiment.

H5 (Intuitive interface): Most users using JEDAI.Ed were able to navigate the interface without any help. JEDAI.Ed is modern and includes many quality-of-life features such as the ability to minimize blocks, etc. which are lacking in JEDAI. Fig. 3 shows that 66% of users found JEDAI.Ed's UI intuitive as compared to JEDAI which had only 28% positive sentiment and had a 19% negative sentiment.

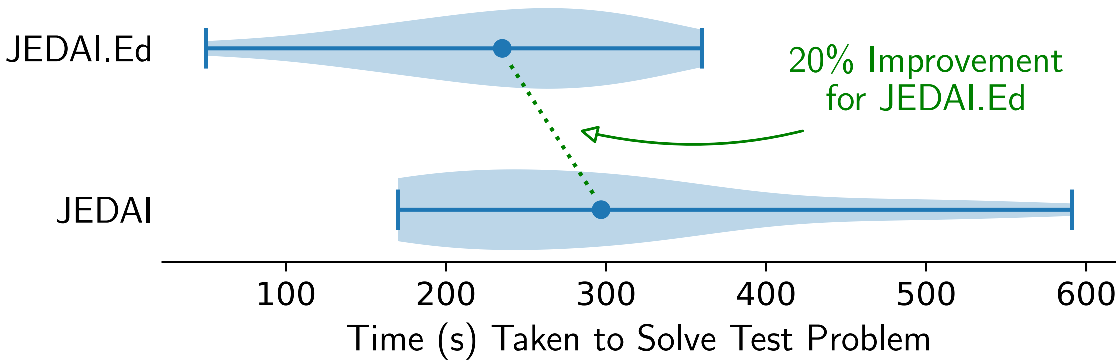


Figure 4. Violin plots that indicate the time taken by participants to solve the test task. ● represents the mean.

H7 (Faster solving): Fig. 4 shows the distribution of times required to solve the tesk task. JEDAI.Ed users were able to solve the test task in 235 seconds which is 20% faster than JEDAI. There were 4 (3) users for JEDAI.Ed (JEDAI) that were not able to solve the test task. One additional JEDAI.Ed user encountered an internal error requiring a system restart thus resulting in them not being counted.

One of the key advantages of JEDAI.Ed is the adaptive curriculum module that appropriately adjusts the difficulty of tasks so that users can learn faster. JEDAI.Ed also informs users of multiple invalid actions (and explaining two of them) in real-time as compared to JEDAI where users need to submit plans to get any feedback and are only informed and explained of a single invalid action.

Improved Sentiment over JEDAI Fig. 5 shows that users have a positive (negative) sentiment change across all metrics when interacting with JEDAI (JEDAI.Ed) first and then experiencing JEDAI.Ed (JEDAI). These observations, along with the rest of our analysis, shows that JEDAI.Ed offers several significant improvements over JEDAI resulting in an overall enhanced user-experience when using JEDAI.Ed as a platform for robotics programming.

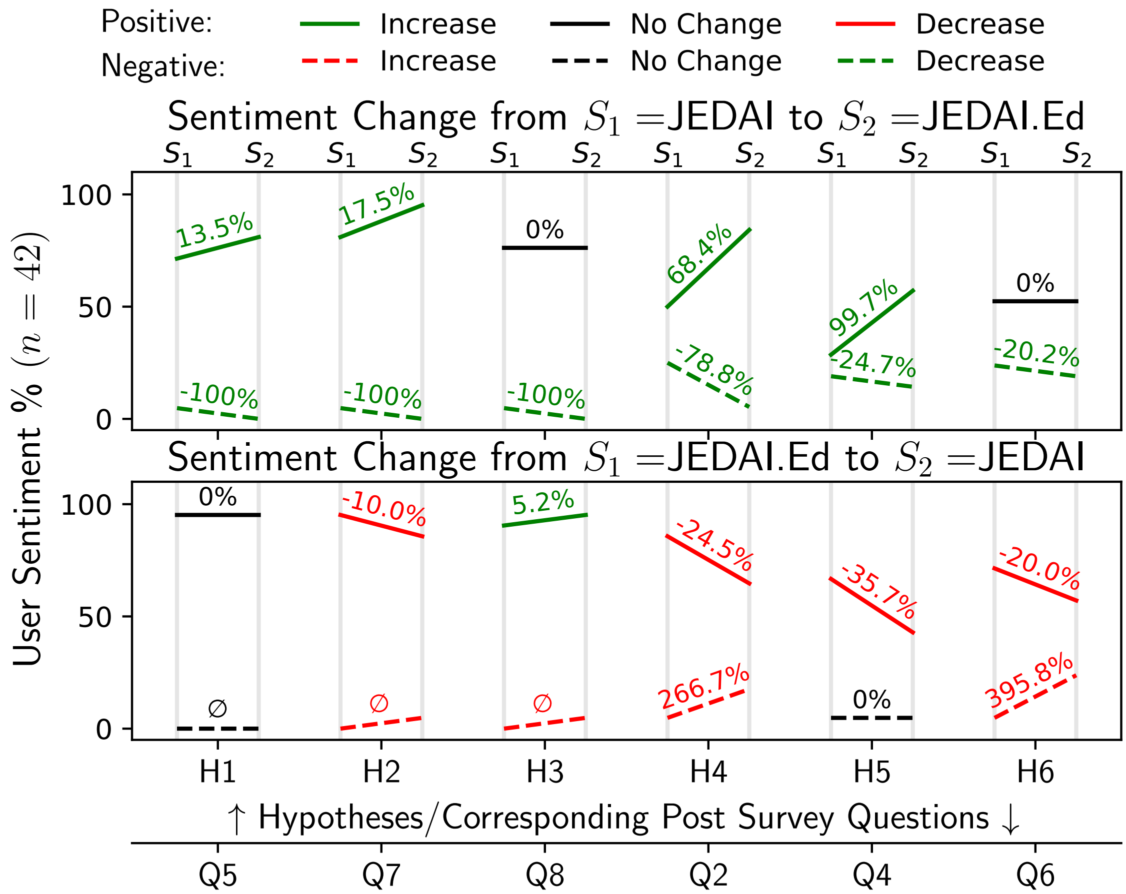


Figure 5. Slope charts for results from our sentiment change phase. Users interacted with system S_1 first and S_2 next. The y-axis shows the absolute user sentiment while annotations on the line plots show the % improvement ($\frac{S_2 - S_1}{S_1} \times 100$). We use $\varnothing$ when $S_1 = 0$.

2.6 Related Work

This work brings together several independent research directions in a single platform. We discuss them here.

Visualizations in planning Robot planning involves searching for a plan to reach some target environment configuration. There are tools which help visualize this search to make it is easy to understand for the users. Such tools include Web Planner (Magnaguagno *et al.*, 2017), Planimation (Chen *et al.*, 2019), PDSim (De Pellegrin and Petrick, 2021), vPlanSim (Roberts *et al.*, 2021), etc. Cantareira *et al.* (2022) introduced a method that summarizes plan and interaction to help understand the plan better. These methods focus on visualizing the planning process for expert users, whereas JEDAI.Ed, in addition, also helps novice users in planning on their own, giving them hints, and explaining their mistakes to them.

Easy to use robot programming interfaces Programming for robots is not an easy task, especially for novice users. Previous approaches have aimed at enabling non-experts to program robots in terms of discretized low-level instructions. CoBlox (Weintrop *et al.*, 2018) used a similar interface for creating low-level plans for robots but also required the users to provide waypoints to generate motion plans. Winterer *et al.* (2020) analyzed the use of Blockly for programming industrial robots aimed at expert users. (Huang *et al.*, 2020) developed a visual spatial programming interface for bots that includes drawing on a map, functions, conditionals, loops, etc. for domain experts to provide high-level actions to robots. These approaches are aimed at making it easy to program robots but still expect the user to be an expert.

AI concepts for students There are many approaches for teaching complex AI-related concepts to non-experts (e.g., students). Robot-VPE (Krishnamoorthy and Kapila, 2016) and Code3 (Huang and Cakmak, 2017) used a Blockly-like interface for K12 students to write programs for robots. Broll and Grover (2023) created a tool to teach complex ML concepts to students using block-based programming and pre-programmed games. Maestro (Geleta *et al.*, 2023) used goal-based scenarios to teach students about robust AI.

Generating explanations with easy-to-understand interfaces There is a large body of work on generating explanations for user-provided plans. Few such approaches (Grover *et al.*, 2020; Valmeekam *et al.*, 2022; Brandao *et al.*, 2021; Kumar *et al.*, 2022) use an easy-to-understand user interface and natural language to make the explanations easily accessible to novice users. These approaches work well for high-level symbolic domains but do not integrate low-level planning systems with them, hence the user is expected to reason and create plans in terms of complex low-level instructions.

Chapter 3

SUPPORT FOR STOCHASTIC DOMAINS

In this chapter, I describe the extension made to JEDAI.Ed for incorporating stochasticity in domains by providing users with the ability to specify blocks using procedural programming blocks such as `while` loops.

While JEDAI.Ed provides a host of features to facilitate users to teach themselves high level planning, it is limited in scope to deterministic domains, in which action effects apply with complete certainty. An important part of planning is operating in uncertain conditions, where actions may fail with some finite non zero probability. Towards this end, we have introduced support for stochastic domains. In the user-interface, users can create looping blocks within which an action block can be placed, creating a plan that iterates until the given action is successful. When the simulation is run for a stochastic plan, each action succeeds or fails with the respective configured probability.

Fig. 6 shows a screenshot of the Blockly interface with a sample plan containing loops. Such a plan containing loops needs to be unrolled into an acyclic graph in order for it to be compatible with low-level planners and produce a simulation. A cyclic plan may be considered as a graph in which each node is a tuple of a state and an action, with edges going form the node to the next state action node, or looping to self, indicating action failure. The edges themselves are weighted according to the probability of transitioning from one node to the other.

In Fig. 7, we depict a plan with two actions, each of which may succeed or fail with equal probability. On reaching the final node, the plan execution stops and the

Figure 6. Screenshot of the Blockly interface depicting action blocks placed within loops .

execution is considered successful. For a low-level planner to run this plan, it requires the plan to be in the form of an acyclic directed graph, and along with the state-action tuple, the probability of reaching that node. To achieve this, a simple breadth-first unrolling of the graph can be done, with the probability of a node defined as the probability of the parent multiplied by the weight of the respective incoming edge. The

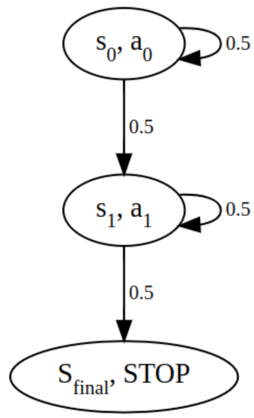


Figure 7. A sample plan with stochastic actions.

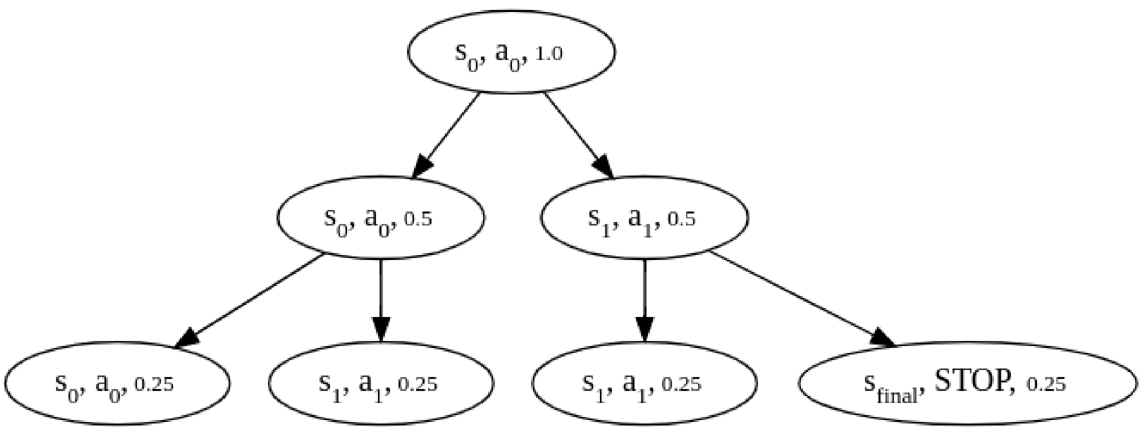


Figure 8. The directed acyclic graph obtained from unrolling the plan depicted in Fig. 7 up to a maximum depth of three nodes.

stopping condition for the unrolling is either the node being the final node of the plan, or if the height of the tree reaches a predefined limit. Unrolling the graph depicted in Fig. 7 to a maximum height of three, we obtain a low-level planner compatible tree which is shown in Fig. 8. Note that in each level of the tree, the probabilities of all nodes sum to one.

Chapter 4

CONCLUSION AND FUTURE WORK

This work introduces JEDAI.Ed, an educational tool for teaching robotics programming to beginner learners, which builds upon and improves over its predecessor JEDAI. JEDAI.Ed adapts to individual learner requirements, provides rich and easily understandable explanations and hints, and has an intuitive and easy to use UI. Further, in order to better approximate real world tasks, JEDAI.Ed adds support for stochastic tasks and looping. A user study conducted with students validate the efficacy of JEDAI.Ed in teaching robotics planning to beginner learners, its ease of use and intuitiveness as well as the usefulness of its various components.

4.1 Future Work

Despite largely positive feedback for JEDAI.Ed from the user study, a few points for improvement were identified. These minor issues which can be easily fixed included certain changes to the UI and the syntactic wording in action blocks. Additionally, fine-tuned LLMs can replace the vanilla pre-trained LLMs that are currently being used to generate better explanations and hints. Although JEDAI.Ed does support execution of plans with stochastic actions, it cannot generate explanations for incorrect actions. Underlying the current explainer is an algorithm that estimates the users current understanding of the environment assuming a deterministic setting. Another implicit assumption is that plan failure occurs only in the case that the user-provided plan contains inapplicable actions, which may not be the only cause of failure in a stochastic domain containing loops. Generation explanations for failures in stochastic settings, therefor, cannot be achieved via simple modifications to the existing explainer. Explaining failures effectively to novices will require new research.

Brandao, M., G. Canal, S. Krivić and D. Magazzeni, “Towards Providing Explanations for Robot Motion Planning”, in “Proc. ICRA”, (2021).

Broll, B. and S. Grover, “Beyond black-boxes: Teaching complex machine learning ideas through scaffolded interactive activities”, in “Proc. EAAI Symposium”, (2023).

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal *et al.*, “Language models are few-shot learners”, in “Proc. NeurIPS”, (2020).

Bylander, T., “The computational complexity of propositional STRIPS planning”, AIJ **69**, 1-2, 165–204, URL https://doi.org/10.1016/0004-3702(94)90081-7 (1994).

Cantareira, G. D., G. Canal and R. Borgo, “Actor-focused interactive visualization for AI planning”, in “Proc. ICAPS”, (2022).

Chen, G., Y. Ding, H. Edwards, C. H. Chau, S. Hou, G. Johnson, M. Sharukh Syed, H. Tang, Y. Wu, Y. Yan, T. Gil and L. Nir, “Planimation”, in “ICAPS 2019 System Demonstrations”, (2019).

De Pellegrin, E. and R. P. A. Petrick, “PDSim: Simulating Classical Planning Domains with the Unity Game Engine”, in “ICAPS 2021 System Demonstrations”, (2021).

Diankov, R., *Automated Construction of Robotic Manipulation Programs*, Ph.D. thesis, Carnegie Mellon University (2010).

Geleta, M., J. Xu, M. Loya, J. Wang, S. Singh, Z. Li and S. Gago-Masague, “Maestro: A gamified platform for teaching ai robustness”, in “Proc. EAAI Symposium”, (2023).

Google, “Blockly”, https://github.com/google/blockly (2018).

Grover, S., S. Sengupta, T. Chakraborti, A. P. Mishra and S. Kambhampati, “RADAR: Automated Task Planning for Proactive Decision Support”, Human–Computer Interaction **35**, 5-6, 387–412 (2020).

Hoffmann, J., “Ff: The fast-forward planning system”, AI Magazine **22**, 3, 57, URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1572 (2001).

Howey, R., D. Long and M. Fox, “VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL”, in “Proc. ICTAI”, (2004).

Huang, G., P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani and A. J. Quinn, “Vipo: Spatial-Visual Programming with Functions for Robot-IoT Workflows”, in “Proc. CHI”, (2020).

Huang, J. and M. Cakmak, “Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts”, in “Proc. HRI”, (2017).

Jackson, A., A. Godwin, S. Bartholomew and N. Mentzer, “Learning from failure: A systematized review”, International Journal of Technology and Design Education **32**, 3, 1853–1873, URL https://doi.org/10.1007/s10798-021-09661-x (2022).

Kalyan, K. S., “A survey of gpt-3 family large language models including chatgpt and gpt-4”, Natural Language Processing Journal **6**, 100048 (2024).

Krishnamoorthy, S. P. and V. Kapila, “Using A Visual Programming Environment and Custom Robots to Learn C Programming and K-12 STEM Concepts”, in “Proceedings of the 6th Conf. on Creativity and Fabrication in Education”, (2016).

Kumar, A., S. L. Vasileiou, M. Bancilhon, A. Ottley and W. Yeoh, “VizXP: A Visualization Framework for Conveying Explanations to Users in Model Reconciliation Problems”, in “Proc. ICAPS”, (2022).

Likert, R., “A technique for the measurement of attitudes”, Archives of Psychology **22**, 140, 5–55 (1932).

Magnaguagno, M. C., R. Fraga Pereira, M. D. Móre and F. R. Meneguzzi, “WEB PLANNER: A Tool to Develop Classical Planning Domains and Visualize Heuristic State-Space Search”, in “ICAPS 2017 Workshop on User Interfaces and Scheduling and Planning”, (2017).

McDermott, D., M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. S. Weld and D. Wilkins, “Pddl – the planning domain definition language”, Tech. Rep. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998).

Moos, D. C. and D. Pitton, “Student teacher challenges: using the cognitive load theory as an explanatory lens”, Teaching Education **25**, 2, 127–141 (2014).

noVNC, “noVNC”, https://novnc.com/info.html, [Online; accessed 2-Feb-2024] (2024).

OpenAI, “GPT-3.5”, https://platform.openai.com/docs/model-index-for-researchers, [Online; accessed 5-Sept-2023] (2022).

Paxton, C., A. Hundt, F. Jonathan, K. Guerin and G. D. Hager, “Costar: Instructing collaborative robots with behavior trees and vision”, (2016).

Pearson, E. S., R. B. D’Agostino and K. O. Bowman, “Tests for departure from normality: Comparison of powers”, Biometrika **64**, 231–246 (1977).

Resnick, M., J. H. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman and Y. B. Kafai, "Scratch: programming for all", Commun. ACM **52**, 11, 60–67, URL https://doi.org/10.1145/1592761.1592779 (2009).

Roberts, J. O., G. Mastorakis, B. Lazaruk, S. Franco, A. A. Stokes and S. Bernardini, "vPlanSim: An Open Source Graphical Interface for the Visualisation and Simulation of AI Systems", in "Proc. ICAPS", (2021).

Ross, A. and V. L. Willson, *One-Sample T-Test*, pp. 9–12 (SensePublishers, Rotterdam, 2017), URL https://doi.org/10.1007/978-94-6351-086-8_2.

Russell, S. and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)* (Pearson, 2020), URL http://aima.cs.berkeley.edu/.

Shah, N., D. Kala Vasudevan, K. Kumar, P. Kamojjhala and S. Srivastava, "Anytime Integrated Task and Motion Policies for Stochastic Environments", in "Proc. ICRA", (2020).

Shah, N., P. Verma, T. Angle and S. Srivastava, "JEDAI: A System for Skill-Aligned Explainable Robot Planning", in "Proc. AAMAS", (2022).

Sreedharan, S., S. Srivastava and S. Kambhampati, "Hierarchical Expertise Level Modeling for User-Specific Contrastive Explanations", in "Proc. IJCAI", (2018).

Valmeekam, K., S. Sreedharan, S. Sengupta and S. Kambhampati, "RADAR-X: An Interactive Mixed Initiative Planning Interface Pairing Contrastive Explanations and Revised Plan Suggestions", in "Proc. ICAPS", (2022).

Weintrop, D., A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd and D. Franklin, "Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices", in "Proc. CHI", (2018).

Wilcoxon, F., "Individual comparisons by ranking methods", Biometrics Bulletin **1**, 6, 80–83, URL http://www.jstor.org/stable/3001968 (1945).

Winterer, M., C. Salomon, J. Köberle, R. Ramler and M. Schittengruber, "An Expert Review on the Applicability of Blockly for Industrial Robot Programming", in "Proc. ETFA", (2020).

APPENDIX A

COMPARISON BETWEEN JEDAI.ED AND JEDAI

Desiderata	JEDAI.Ed	JEDAI
Open source	✔	✔
Minimal system requirements	✔	✔
Highly customizable	✔	✔
Integrated simulation	✔	✔
User-adaptive curriculum problem generation	✔	✗
LLM translated explanations	✔	✗
Hint messages	✔	✗
LLM translated hint messages	✔	✗
State space annotation	✔	✗
Continuous plan checking and explanation generation	✔	✗
Explanation of multiple failing actions	✔	✗
Collapsible information fields	✔	✗
Minimize/Maximize blocks	✔	✗
Clicking on error message highlights corresponding action block	✔	✗
Robot simulator in the same window	✔	✗

Table 3. A comparison of the features of JEDAI.Ed compared to JEDAI.

The supplementary material begins with an in-depth treatment of the desiderata mentioned in the main paper. This is followed by a exposition of the design considerations and extensibility of JEDAI.Ed. We then briefly introduce environment domains that are included in JEDAI.Ed and conclude with details about the user study and analysis of the results.

Table 3 provides a contrast between the features of JEDAI.Ed and JEDAI.

A.1 Intuitive user interface

The JEDAI user interface was a static browser window which would display all the user assistance information on to the screen simultaneously. The result was a text dense webpage. Furthermore, JEDAI required the user to manually click a button in order to check the validity of their plans and receive explanations about their errors. To test even incremental changes in their plans, users would have to move the mouse pointer back and forth between the Blockly workspace and the submit button. Finally, JEDAI presents the robot simulation in a separate browser window, further burdening the user with managing multiple browser tabs or windows while solving tasks.

The unmodified screenshots for JEDAI.Ed and JEDAI are presented in Fig. 12 and Fig. 14 respectively.

A.2 UI Improvements

Extensive work has gone into remodelling the interface of JEDAI to be more user-friendly and intuitive. An important motivation behind this was to not overload the user with too much information on the screen and bring all the components, including the simulation window into a single browser window. Optimizing on the screen space available, JEDAI.Ed presents explanations, hints etc. as collapsible text fields. This allows for a high level summary to be visible immediately on the screen along with the option for the user to expand and consume more detailed information if needed. In order to make the connection between an error explanation and the corresponding block clear to the user, JEDAI.Ed also highlights the action block when its corresponding explanation dropdown is selected. Similarly the robot simulation was integrated within the same webpage containing the Blockly workspace and explanation dropdown lists, thus providing a single, compact view port for the user.

A.3 Blockly

Another accessibility feature introduced in JEDAI.Ed is the ability to minimize and maximize blocks. Tasks that require longer plans, the connected blocks quickly overflow beyond the workspace requiring the user to scroll. Minimizing previously added blocks allows the user to be able to add and edit the arguments of the new block they are adding in the same workspace without needing to scroll down to find the last block each time.

A.4 State Space Display

To facilitate the users' understanding on the environments that they are solving tasks in, JEDAI.Ed introduces a State Space Display. Starting with the initial state of the environment and listing the change after the application of each action n the users submitted plan, the list of all true predicates is displayed, as seen in Fig. $1f$. Again, in keeping with the objective of not filling up the screen with text dense messages, the list of predicates is hidden by default and the user can expand and collapse the list as they need. In the event of an invalid submitted plan, the display is truncated at the last valid action and the error explanation dropdown is displayed.

A.5 Curriculum Generation

JEDAI and JEDAI.Ed both contain a diverse set of domains and problem tasks for users to solve (App. B.2), however JEDAI does not offer an in-built method which provides a systematic learning journey to the user. In JEDAI, an educator would have to manually design a lesson plan with students sequentially exposed to problems of increasing difficulty, but that would still be a one-size-fits-all approach, without possibility of the curriculum being tailor-made for individual learners' needs. This problem is alleviated by JEDAI.Ed's curriculum design module which automaticallyadapts to each individual learner and provides them a customized learning path for learning about the capabilities of a robot in an environment. Our curriculum module progressively generates challenging problems of increasing difficulty, while automatically adapting to users performance.

User created plans are continuously evaluated to assess their level of understanding of the various actions. Each plan is broken down into its constituent actions which are sequentially applied starting from the initial state. An action being applicable in the current state implies the user having a correct understanding of said action. Here, we increment the cost associated with that action and update the current state to be the state resulting in the application of the action on the current state. Conversely, if an action is inapplicable, the entire plan is rendered invalid and updating the current state is impossible. The associated cost of the action is decremented and the action-cost mapping is returned. This above process is illustrated in Alg. 1.

Algorithm 1: Action-Cost Mapping Update

Input: Submitted plan P, actions-costs mapping A, initial state S
Output: Update actions-costs mapping A

foreach *action a in P* **do**
    if *isApplicable*(s, a) **then**
        $A[a] \leftarrow A[a] + 1$
        $s \leftarrow applyAction(s, a)$
    else
        $A[a] \leftarrow \max(0, A[a] - 1)$
        return A
return A

Alg. 2 showcases our overall process for adaptive curriculum generation. To generate a problem task using this action-cost mapping, we use a simple greedy search over the state space, where nodes are states of the environment and weighted edges are actions with the weight of the edge being the cost associated with that action. The fringe is initialized as a minimum priority queue, with the priority of a node set as the cumulative action cost needed to reach that node and the order of addition to the fringe used to break ties. Nodes are then popped from the fringe and for each popped node, the states reachable from that node in one action are added to the fringe with the incremented cumulative action-cost. This procedure is carried out until an action currently unknown to the user is seen, or if a preset maximum depth is reached. For our experiments, the maximum depth was set to four.

Algorithm 2: Adaptive Curriculum Task Generator

Input: adaptive action-cost mapping A, initial state s_0, set of grounded actions $\mathcal{A}$, maximum tree depth d_{max}

Output: Problem Goal State g

```
f ← Min-Priority-Queue
v ← Empty visited set
add(f, (s_0, 0, 0))
add(v, s_0)
while True do
    s, c, d ← pop(f)
    foreach a ∈ 𝒜 do
        if isApplicable(s, a) then
            s′ ← applyAction(s, a)
            c′ ← c + A[a]
            d′ ← d + 1
            if A[a] = 0 or d′ ≥ d_max then
                g ← s
                return g
            if s′ ∉ v then
                add(v,s′)
                add(f, (s’,d’,c’))
```

All action costs are initialized uniformly to zero, representing that the user is unfamiliar with all the actions. As the user solves curriculum generated problems correctly, they are presented with problems that each introduce an additional unknown action. This ensures that while solving a problem, the user has only one new action to learn thus easing the cognitive burden placed upon them. The very first curriculum task generated, with all action costs set to zero, will select any action applicable in the initial state and return the state resulting from its application as the goal state. Future work may investigate how this cold start problem impacts user engagement and learning, and how it may be resolved.

A.6 Intuitive Explanations and Hints

In explaining action failures, JEDAI relied upon hand coded text templates, which print out the failing action and the unmet preconditions of said action. While logically sound and providing complete information, theses explanations are not very human friendly to read. JEDAI.Ed uses GPT-3.5 to convert these explanations to be more human readable and user friendly. JEDAI is also limited to explaining failure for only one action, which is a handicap JEDAI.Ed does not suffer from.

As mentioned in the main paper, JEDAI.Ed also allows users to get hints to help them in high-level planning. The hint is displayed in the form of a grounded high-level action, with the action name visible, but some of the grounded objects that are arguments to the action obscured. This achieves the twin objectives of nudging the user in the right direction while also not spoon feeding them the answer directly.

A.7 LLM Prompts

We use LLMs to translate explanations and hints to be more readable and user friendly. The explanation or hint generated from the user assistance module is augmented with domain and problem pddl files of the problem being solved. This information, along with a simple prompt is fed into the LLM which is tasked with a translating the explanation given the context to a readable natural language description. The LLM itself does not generate any explanation or hint, but merely acts as a translation interface. GPT-3.5 was prompted with the following text to generate user-friendly explanations and hints, which is domain agnostic and through repeated experimentation, was found to generalize across multiple problems and domains. Text within angular braces acts as a placeholder for the actual text that is input to the prompt, but omitted here for brevity. Note that the LLM is not being tasked with generating any explanation, but only to convert them to natural language:

Explanation Generation Prompt

The following lines describe the ⟨domain⟩ domain file :
⟨domain pddl⟩

The problem to be solved is described in pddl format as:
⟨problem pddl⟩

While running a plan for a problem, an action failed and an explanation generator was used to generate the following explanation:
Explanation: ⟨explanation⟩

The state of the problem - which means the set of predicates that are true in the plan upto the first invalid action are as follows
State: ⟨state⟩

Can you please convert the explanation into a brief, more non-expert friendly message that a novice user can understand? Also, can you suggested briefly what could be done to fix the issue, taking into account the state reached by the plan so far?

The error explanation for a failing action, generated using fixed text templates, as seen in Fig. 1h (marked as the TLDR version) is presented below:

Text Template Based Explanation

The action at step 3 (Place at location 'counter' object 'can_blue' using gripper 'gripper' this robot 'fetch') could not be performed because 'gripper' is not holding 'can_blue'.

Using the explanation generation prompt we mentioned earlier, and filling in the above text template generated explanation, GPT-3.5 generated the following user friendly explanation, which is also depicted in Fig. 1h (marked Explanation) :

LLM Generated Explanation

Explanation: The action of placing the blue can on the counter failed because the gripper is not holding the blue can. To fix this issue, you can have the robot fetch the blue can and place it on the counter using the gripper properly before trying to place it on the table. Make sure the gripper is holding the correct object before attempting to place it at the desired location.

A.8 Prompt Engineering

Educators can customize LLM descriptions for audiences of different expertise by simply designing appropriate prompts and using them as input to JEDAI.Ed's fine-tuned LLM without any additional training. This allows educators to take into account the needs of different users without having to expend resources for training or fine-tuning an LLM. An example prompt that can be used for generating explanations for novice users is stated below.

Hint Generation Prompt

This is the pddl domain file for the ⟨domain⟩ domain :

⟨domain pddl⟩

A user has to solve a this problem task described in pddl
⟨problem pddl⟩

The plan was run till the problem reached this state - that is the set of predicates that are true :
⟨state⟩

And the hint generated, which suggests which next action to take with certain arguments to actions replaced with ? is given below:
⟨hint⟩

Can you please convert the explanation into a brief, more non-expert friendly message that a novice user can understand? Also, can you suggested briefly what could be done to fix the issue, taking into account the state reached by the plan so far?

A sample hint generated using hand-coded text templates is illustrated below:

Sample Hint

You might want to try the action: Move To Counter from this location ? the robot ? to this location counter

The LLM generated counterpart is shown below. In a fashion similar to the explanation translation, the LLM here is used to translate and paraphrase the hints and solutions using the hint, domain and problem pddl and the state of the environment passed as context:

LLM Generated Hint

The problem is to solve a task in the Cafe World domain. The current state of the problem is that the gripper is empty, there are orders for different cans at the counter location, and the robot is at the starting point. The goal is to place the red can on the red table. To solve the problem, the next action that should be taken is to move the robot to the counter location. This action requires specifying the current location of the robot and the counter location. To fix the issue, you need to execute the move-to-counter action with the appropriate arguments. By doing this, the robot will move to the counter location and be ready to perform further actions to fulfil the goal.

APPENDIX B

DESIGN CONSIDERATIONS

This section provides a detailed description of some of the design considerations made for JEDAI.Ed.

B.1 Modular and Customizable Design

JEDAI.Ed was consciously designed to be modular and customizable, allowing for easy upgrading or replacement of a number of components to suit particular educator needs. We aim to release JEDAI.Ed as an open source software like its predecessor, and hope that the community will build upon and improve our work in order to further the goal of making AI education more accessible to the general public. Despite the many improvements and feature additions in JEDAI.Ed, it is still a lightweight web-based interface which can run on any device with a modern browser. We have also containerized the software to allow the platform to be run on any machine without the user having to worry about installing dependencies. Containerization also assists in deployment over cloud based services allowing for convenient scaling as required, especially suited to schools and other educational settings.

B.1.1 New Domains and Problems

Adding new domains simply requires the additional environment description dae files, the action configuration specifications of the domain, a semantic mapping of predicates and actions to natural language, and the domain and problem pddl files. JEDAI.Ed given all these inputs handles the creation of the blockly interface, web frontend, setting up the simulator as well as explanation generation on its own. Incorporating new problems within existing domains is even simpler and requires only the addition of new pddl problem files, or can be generated automatically using the curriculum generation module.

B.1.2 LLMs

There is flexibility in choosing the LLM used in translating explanations and hints to a user-friendly format. JEDAI.Ed can be used with any LLM, since the interface of passing the prompt abstracts away the internals of the explanation generation. The LLM being used may be stored locally in the system or accessed via an API. Educators, therefore, can use LLMs fine-tuned specifically for the task of converting predicates into human readable language, for instance. Further, the prompt being used to generate responses from the LLM can also be modified as required. different prompts can be used to generate responses in a specific format, or to be less or more verbose as needed.

B.1.3 Hinting

Educators can make hints more, or less transparent to the students as they need. A single tunable real number parameter between 0 and 1 represents the independent probability of each grounded parameter in the action hint being displayed to the user. By increasing this value, hints are more likely to reveal the grounded parameters input to the action in the hint, and vice-versa.

B.1.4 Simulator

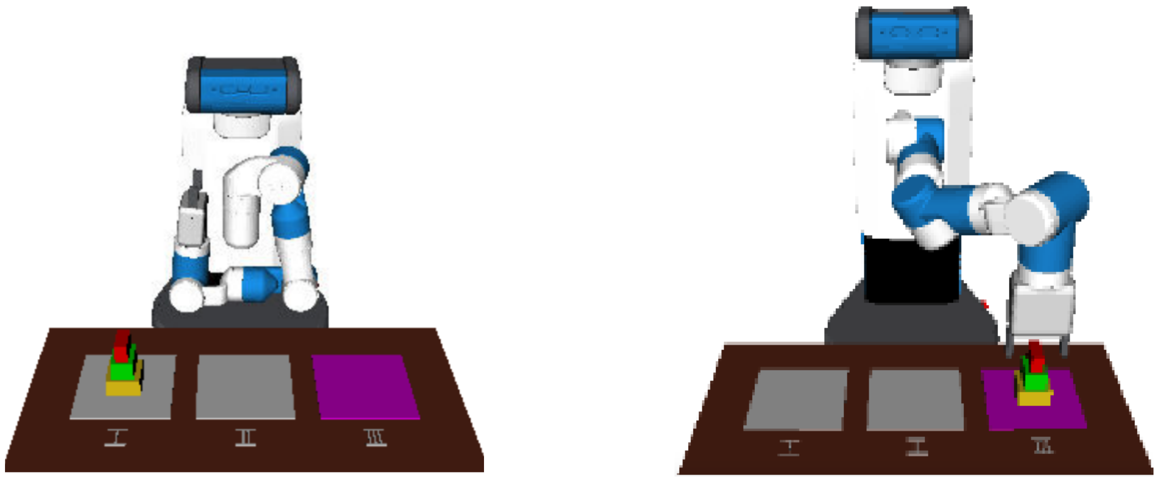
In order to be simulator-agnostic, JEDAI.Ed separates the simulator from the rest of the backend software, and streams the output to the webpage. Our work uses OpenRave streamed using noVNC (noVNC, 2024), but any robot simulator package can be used in its place.

B.2 JEDAI.Ed Bundled Environment Details

Even though the educators can add custom environments to JEDAI.Ed, it comes preconfigured with a few environments to help educators. We have seen one such environment in Fig. 1 in the main paper: Fetch robot in the cafe environment task with delivering orders to various tables. We briefly introduce three more environments here:

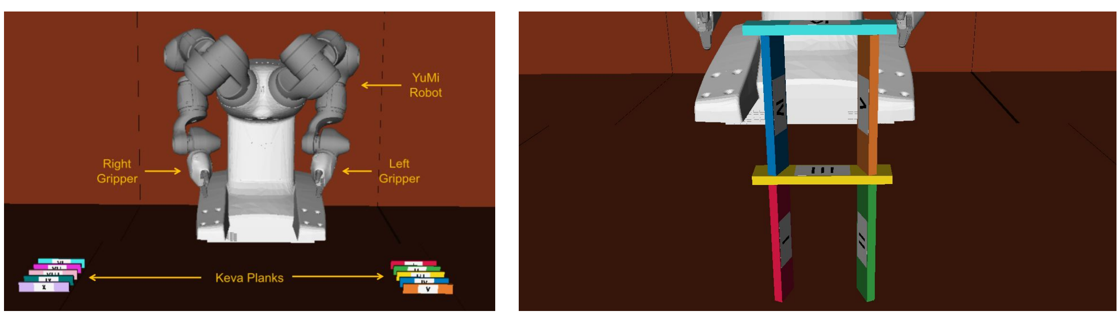
Tower of Hanoi In this environment shown in Fig. 9, Fetch robot has to solve the classical tower of Hanoi problem. It consists of three blocks on top of each other kept at a location. There are three such locations. All three blocks have to be transported to another fixed location, but the robot cannot place a larger block on top of a smaller block. The blocks are uniquely identified using their size.

Keva Planks In this environment shown in Fig. 10, YuMi, a dual-armed robot has to create structures using numbered planks kept on the table. Different structures can be created by placing planks vertically, horizontally or along their edges. Planks can be placed on the table, or on top of other planks.



(i) (ii)

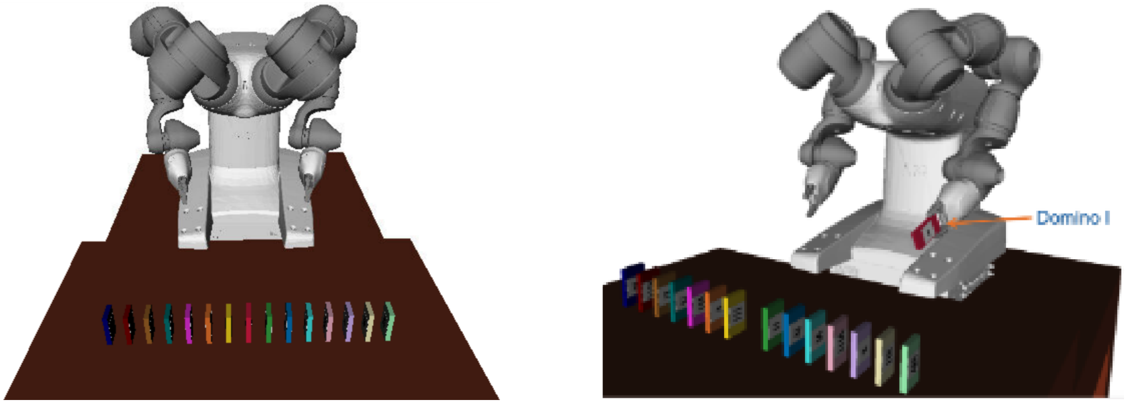
Figure 9. Screenshot of the simulator from the Tower of Hanoi environment. (i) depicts the initial state whereas (ii) a sample goal image for this environment.



(i) (ii)

Figure 10. Screenshot of the simulator from the keva environment. (i) depicts the initial state whereas (ii) a sample goal image for this environment.

Dominoes In this environment shown in Fig. 11, similar to the Keva planks setup, YuMi robot has to create structures using dominoes. Each domino can be uniquely identified using an ID that is printed on each domino. The users can move around the camera in the simulator to view these IDs.



(i) (ii)

Figure 11. Screenshot of the simulator from the dominoes environment. (i) depicts the initial state whereas (ii) a sample goal image for this environment.

APPENDIX C

USER STUDY DETAILS AND RESULTS

The following sections provide a comprehensive view of the results obtained from our user study.

C.1 Study Details

As mentioned in the main paper, the study was divided into four phases. This section provides complete data for the results (Fig. 3, Table 2) presented in the main paper.

C.1.1 Pre-survey Phase

Table 4 shows the questions and collected responses from the pre-survey questionnaire that was administered as a part of this phase. Note that the participants were yet to interact with JEDAI.Ed or JEDAI in this phase. We provide the statistical significance test results from both the overall participant pool and also segregate the responses of the participants who were assigned JEDAI.Ed or JEDAI in the next phase. Our results (one-sample t-test) show that participant responses are statistically significant across these dimensions ($\mu_0^1 = 2$) except for Pre Q5. For Pre Q5, both the combined and per-system statistical results are insignificant. Thus, participants found it neither easy nor difficult to program robots. Similarly, we cannot reject the null hypothesis from the results of the two-sample t-test (that compares responses between JEDAI.Ed and JEDAI for the questions) thus showing that participants assigned to either JEDAI.Ed or JEDAI had similar demographics in our study.

Our pre-study results also show that participants had little to no experience with robotics programming (Pre Q6) and $\geq 50\%$ think about AI (Pre Q2) or use AI-based tools (Pre Q3) daily. This indicates the greater need to impart AI education to people who will be interacting with AI systems frequently.

C.1.2 Training Phase

There was no questionnaire administered to the participants at the end of this phase.

C.1.3 Test Phase

Table 6 and Table 8 present the results obtained after administering the post-survey questionnaire at the end of this phase. Depending upon the control group to which they were assigned, participants had interacted with JEDAI.Ed or JEDAI (but not both) at this point. We report these results pictorially in Fig. 3 of the main paper. Post Q1, Q3, Q9, and Q10 are not discussed in the main paper. The results for Post Q1 are not statistically significant. This is not surprising since both JEDAI.Ed and JEDAI manage to increase user understanding of the robot's limitations and capabilities (Post Q8, H6).

Table 8 contains JEDAI.Ed specific questions (Q3, Q9, Q10) meant to evaluate the results of the hinting feature of JEDAI.Ed. We did not administer these questions to participants interacting with JEDAI since IRB protocol forbids the exposure of questions that can allow users to infer the presence of a different control group. Our results show that users the test problem moderately challenging and utilized hints only a few times. We attribute this to the training phase where the adaptive curriculum generation algorithm taught the users appropriately such that their need for hints was reduced. Since there was no study administered during the training phase we do not have any data pertaining to the usage of hints during training.

In Table 6, Post Q2 and in Table 8 Post Q10 provided students with a sixth option for the users to choose - "I did not encounter any error" and "I did not receive any "Hints"" respectively. These options were provided to handle the edge cases that the users might experience where for Post Q2, the user do not make a mistake while completing the given problem and for Post Q10, the user did not use the "Get A Hint" button of the interface to get a hint for next step. In Table 6, there is one student who chose the sixth option in Post Q2 after interaction with JEDAI as the first system. In Table 8, seven users opted for sixth option in Post Q10 after interacting with JEDAI.Ed as their first system. These data points for the respective questions were not considered for the statistical analysis of the aforementioned questions because this options do not adhere to the Likert scale that we used for the analysis.

As Table 12 shows, all the questions in the Post Survey used to evaluate hypotheses H1 - H7 show medium (>0.3) or large (>0.5) absolute effect size between JEDAI.Ed and JEDAI.

C.1.4 Sentiment Change Phase

Table 10 and Table 11 presents the overall Sentiment Change that we observed from the data collected from the user study. These results were obtained after the users were asked to interact with second system(S_2). Table 10 contains the overall sentiment change (both positive and negative) for users who were given JEDAI.Ed as their first system(S_1) and JEDAI as second system(S_2). Table 11 contains the results for sentiment change (both positive and negative) for users who interacted with JEDAI first and then with JEDAI.Ed.

We don't report the paired t-test values for these groups because for most of the questions, the paired distribution does not satisfy the normality assumption of paired t-test. We tested the normality of the data by performing d'Agostino-Pearson test (Pearson *et al.*, 1977). We were unable to run Wilcoxon Signed-Rank (Wilcoxon, 1945) test because the paired data has too many ties and the sample size was insufficient to run the test.

As discussed in section C.1.3, Post Q2 in Table 10 and Table 11 have sixth option provided to the users that does not adhere to Likert scale. Also, as per IRB protocol, users can choose not to respond to any of the questions in the survey. For both some questions in both the mentioned tables, there are users who have either opted for option sixth in Post Q2 or have chosen not to respond to some questions. In both the cases, we did not consider the data points where either the user has opted for the sixth option or has chosen not to respond as both scenarios differ from Likert scale options that we have chosen for the analysis of the questions. In Table 10, for JEDAI - Post Q1, 1 user has no response and for JEDAI Post Q2, 4 users have opted for sixth option. In Table 11, for JEDAI.Ed Post Q2, 2 users have opted for sixth option.

Question	Question and Responses				
Pre Q1	How familiar are you with Computer Science and Artificial Intelligence (A.I.)?				
	Not at all (0)	Slightly well (1)	Moderately well (2)	Very well (3)	Extremely well (4)
Total	3	14	18	6	1
JEDAI.Ed	3	5	8	4	1
JEDAI	0	9	10	2	0
Pre Q2	How often do you think about A.I. in your day-to-day life??				
	Never (0)	Once a week (1)	2-3 times a week (2)	4-6 times a week (3)	Daily (4)
Total	1	3	8	11	19
JEDAI.Ed	1	0	3	4	13
JEDAI	0	3	5	7	6
Pre Q3	How often do you interact with tools that use A.I.?				
	Never (0)	Once a week (1)	2-3 times a week (2)	4-6 times a week (3)	Daily (4)
Total	2	4	5	10	21
JEDAI.Ed	0	2	2	4	13
JEDAI	2	2	3	6	8
Pre Q4	How curious are you to learn about the extent to which A.I. systems and robots can be used today?				
	Not curious (0)	Slightly curious (1)	Moderately curious (2)	Very curious (3)	Extremely curious (4)
Total	1	4	3	21	13
JEDAI.Ed	0	2	1	12	6
JEDAI	1	2	2	9	7
Pre Q5	Assume you want a household robot to get you water from the refrigerator How difficult do you think it is to give it instructions to perform this task?				
	Very difficult (0)	Slightly difficult (1)	Neither difficult nor easy (2)	Slightly easy (3)	Very easy (4)
Total	3	20	6	10	3
JEDAI.Ed	2	7	4	5	3
JEDAI	1	13	2	5	0
Pre Q6	How familiar are you with robotics programming?				
	Not at all (0)	Slightly familiar (1)	Moderately familiar (2)	Very familiar (3)	Extremely familiar (4)
Total	26	13	3	0	0
JEDAI.Ed	11	9	1	0	0
JEDAI	15	4	2	0	0

Table 4. Pre-survey questionnaire administered during the *pre-survey phase* of our user study. For each Question ID, the first row is the question as it was presented to the participants. The second row lists the possible answers for the question (and the corresponding Likert scale values in parentheses). The third row presents the responses by all participants. The breakdown of the total responses by control group are presented in the fourth and fifth rows.

Questions		One-sample			Two-sample
		μ_0^1	μ	p	p
Pre Q1	Total	0	1.71 ± 0.89	1.56e-15	0.36e-00
	JEDAI.Ed		1.76 ± 1.09	3.80e-07	
	JEDAI		1.66 ± 0.65	2.46e-10	
Pre Q2	Total	2	3.04 ± 1.08	8.61e-08	0.58e-01
	JEDAI.Ed		3.33 ± 1.06	6.44e-06	
	JEDAI		2.76 ± 1.044	0.16e-02	
Pre Q3	Total	1	3.04 ± 1.20	4.41e-14	0.81e-01
	JEDAI.Ed		3.33 ± 1.01	6.71e-10	
	JEDAI		2.76 ± 1.33	3.36e-06	
Pre Q4	Total	2	2.97 ± 0.99	1.47e-07	0.35e-00
	JEDAI.Ed		3.04 ± 0.86	1.95e-05	
	JEDAI		2.90 ± 1.13	0.15e-02	
Pre Q5	Total	2	2.23 ± 1.12	0.17e-00	0.93e-01
	JEDAI.Ed		2.00 ± 1.26	1.00e-00	
	JEDAI		2.47 ± 0.92	0.29e-01	
Pre Q6	Total	0	0.45 ± 0.63	3.59e-05	0.26e-00
	JEDAI.Ed		0.52 ± 0.60	0.71e-03	
	JEDAI		0.38 ± 0.66	0.16e-01	

Table 5. t-test values for the Pre-survey questionnaire (μ_0^1 represents the null hypothesis mean used to conduct the one-sample t-test). We used $\alpha = 0.05$ for determining statistical significance for the t-tests.

Question	Question and Responses				
Post Q1	After interacting with the JEDAI.Ed system, how inclined are you to learn how daily problems are being solved with A.I.?				
	Not inclined (0)	Slightly inclined (1)	Moderately inclined (2)	Very inclined (3)	Extremely inclined (4)
JEDAI.Ed	0	2	6	6	7
JEDAI	2	3	4	11	1
Post Q2	How helpful were the explanations that were given for the cause of an error?				
	Not helpful (0)	Slightly helpful (1)	Moderately helpful (2)	Very helpful (3)	Extremely helpful (4)
JEDAI.Ed	0	1	2	6	12
JEDAI	1	4	5	5	5
Post Q4	How intuitive was the interface?				
	Not intuitive (0)	Slightly intuitive (1)	Moderately intuitive (2)	Very intuitive (3)	Extremely intuitive (4)
JEDAI.Ed	0	1	6	12	2
JEDAI	0	4	11	4	2
Post Q5	As compared to before participating in this user study, how much has your curiosity increased to learn more about AI systems and robots?				
	Highly decreased (0)	Slightly decreased (1)	Neither increased nor decreased (2)	Slightly increased (3)	Highly increased (4)
JEDAI.Ed	0	0	1	9	11
JEDAI	0	1	5	8	7
Post Q6	How well do you think you now understand how one can use an AI system to make a plan for a robot to perform a task?				
	Not well (0)	Slightly well (1)	Moderately well (2)	Very well (3)	Extremely well (4)
JEDAI.Ed	0	1	5	11	4
JEDAI	1	4	5	9	2
Post Q7	Do you agree that the JEDAI.Ed system made it easier for you to provide instructions to a robot for performing tasks?				
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)
JEDAI.Ed	0	0	1	9	11
JEDAI	0	1	3	11	6
Post Q8	Do you agree that JEDAI.Ed helps improve the understanding of the limitations and capabilities?				
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)
JEDAI.Ed	0	0	2	12	7
JEDAI	0	1	4	12	4

Table 6. Post-survey questionnaire administered during the *test phase* of our user study. For each Question ID, the first row is the question as it was presented to the participants. The second row lists the possible answers for the question (and the corresponding Likert scale values in parentheses). Third and fourth row represent the number of participants who chose that answer for the question after JEDAI.Ed and JEDAI respectively. We include here only those questions that were presented to both control groups.

Question		One-Sample		Two-Sample
		μ	p	p
Post Q1	JEDAI.Ed	2.85 ± 1.01	5.52e-08	0.14e-00
	JEDAI	2.28 ± 1.10	3.11e-05	
Post Q2	JEDAI.Ed	3.38 ± 0.86	2.23e-07	0.67e-02
	JEDAI	2.42 ± 1.20	0.59e-01	
Post Q4	JEDAI.Ed	2.27 ± 0.71	9.41e-05	0.22e-01
	JEDAI	2.19 ± 0.87	0.16e-00	
Post Q5	JEDAI.Ed	3.47 ± 0.60	4.24e-10	0.28e-01
	JEDAI	3.00 ± 0.89	5.17e-05	
Post Q6	JEDAI.Ed	2.85 ± 0.79	3.81e-05	0.46e-01
	JEDAI	2.33 ± 1.06	0.83e-01	
Post Q7	JEDAI.Ed	3.47 ± 0.60	4.24e-10	0.23e-01
	JEDAI	3.04 ± 0.80	7.81e-06	
Post Q8	JEDAI.Ed	3.23 ± 0.62	1.56e-08	0.64e-01
	JEDAI	2.90 ± 0.76	2.78e-05	

Table 7. t-test values for the Post-survey questionnaire ($\mu_0^1 = 2$ for the one-sample t-test). We used $\alpha = 0.05$ for determining statistical significance for the t-tests

Question	Question and Responses				
Post Q3	How challenging were the problems in the JEDAI.Ed session?				
	Not challenging (0)	Slightly challenging (1)	Moderately challenging (2)	Very challenging (3)	Extremely challenging (4)
JEDAI.Ed	3	10	6	1	1
Post Q9	How often did you use the "Hint" button during the hands-on session?				
	Once per problem (0)	2-4 times per problem (1)	Once during the session (2)	Never (3)	Didn't notice any hint button (4)
JEDAI.Ed	4	3	9	5	0
Post Q10	How well do you think "Hint" functionality helped you while solving the problems?				
	Not well (0)	Slightly well (1)	Moderately well (2)	Very well (3)	Extremely well (4)
JEDAI.Ed	0	4	5	3	2

Table 8. Post-survey questionnaire administered during the *test phase* of our user study. For each Question ID, the first row is the question as it was presented to the participants. The second row lists the possible answers for the question (and the corresponding Likert scale values in parentheses). The third row represents the number of participants who chose that answer for the question after interacting with JEDAI.Ed. Since JEDAI does not include hints, per IRB protocol, these questions were excluded from users who interacted with JEDAI immediately before being administered the survey (elaborated in App. C.1.3).

Question		One Sample Test	
		μ	p
Post Q3	JEDAI.Ed	1.38 ± 0.97	0.88e-01
Post Q9	JEDAI.Ed	2.23 ± 0.99	0.28e-00
Post Q10	JEDAI.Ed	2.21 ± 1.05	0.45e-00

Table 9. t-test values corresponding to questions in Table 8 ($\mu_0^1 = 2$ for the one-sample t-test). We used $\alpha = 0.05$ for determining statistical significance for the t-tests

Question	Question and Responses					Total Sentiment	
						Positive	Negative
Post Q1	After interacting with the JEDAI.Ed system, how inclined are you to learn how daily problems are being solved with A.I.?						
	Not inclined (0)	Slightly inclined (1)	Moderately inclined (2)	Very inclined (3)	Extremely inclined (4)		
S_1=JEDAI.Ed	0	2	6	6	7	13	2
S_2=JEDAI	0	5	5	3	7	10	5
Post Q2	How helpful were the explanations that were given for the cause of an error?						
	Not helpful (0)	Slightly helpful (1)	Moderately helpful (2)	Very helpful (3)	Extremely helpful (4)		
S_1=JEDAI.Ed	0	1	2	6	12	18	1
S_2=JEDAI	0	3	3	5	6	11	3
Post Q4	How intuitive was the interface?						
	Not intuitive (0)	Slightly intuitive (1)	Moderately intuitive (2)	Very intuitive (3)	Extremely intuitive (4)		
S_1=JEDAI.Ed	0	1	6	12	2	14	1
S_2=JEDAI	0	1	11	7	2	9	1
Post Q5	As compared to before participating in this user study, how much has your curiosity increased to learn more about AI systems and robots?						
	Highly decreased (0)	Slightly decreased (1)	Neither increased nor decreased (2)	Slightly increased (3)	Highly increased (4)		
S_1=JEDAI.Ed	0	0	1	9	11	20	0
S_2=JEDAI	0	0	1	11	9	20	0
Post Q6	How well do you think you now understand how one can use an AI system to make a plan for a robot to perform a task?						
	Not well (0)	Slightly well (1)	Moderately well (2)	Very well (3)	Extremely well (4)		
S_1=JEDAI.Ed	0	1	5	11	4	15	1
S_2=JEDAI	1	4	4	8	4	12	5
Post Q7	Do you agree that the JEDAI.Ed system made it easier for you to provide instructions to a robot for performing tasks?						
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)		
S_1=JEDAI.Ed	0	0	1	9	11	20	0
S_2=JEDAI	1	0	2	11	7	18	1
Post Q8	Do you agree that JEDAI.Ed helps improve the understanding of the robot's limitations and capabilities?						
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)		
S_1=JEDAI.Ed	0	0	2	12	7	19	0
S_2=JEDAI	1	0	0	13	7	20	1

Table 10. Post-survey questionnaire administered during the *sentiment change phase* of our user study. For each Question ID, the first (second) row is the administered question (possible answers). The third and fourth row represent the total participant answers for the question after interacting with JEDAI.Ed and JEDAI respectively.

Question	Question and Responses					Total Sentiment Positive	Total Sentiment Negative
Post Q1	After interacting with the JEDAI.Ed system, how inclined are you to learn how daily problems are being solved with A.I.?						
	Not inclined (0)	Slightly inclined (1)	Moderately inclined (2)	Very inclined (3)	Extremely inclined (4)		
S_1=JEDAI	2	3	4	11	1	12	5
S_2=JEDAI.Ed	0	3	6	10	2	12	3
Post Q2	How helpful were the explanations that were given for the cause of an error?						
	Not helpful (0)	Slightly helpful (1)	Moderately helpful (2)	Very helpful (3)	Extremely helpful (4)		
S_1=JEDAI	1	4	5	5	5	10	5
S_2=JEDAI.Ed	0	1	2	8	8	16	1
Post Q4	How intuitive was the interface?						
	Not intuitive (0)	Slightly intuitive (1)	Moderately intuitive (2)	Very intuitive (3)	Extremely intuitive (4)		
S_1=JEDAI	0	4	11	4	2	6	4
S_2=JEDAI.Ed	0	3	6	8	4	12	3
Post Q5	As compared to before participating in this user study, how much has your curiosity increased to learn more about AI systems and robots?						
	Highly decreased (0)	Slightly decreased (1)	Neither increased nor decreased (2)	Slightly increased (3)	Highly increased (4)		
S_1=JEDAI	0	1	5	8	7	12	1
S_2=JEDAI.Ed	0	0	4	11	6	17	0
Post Q6	How well do you think you now understand how one can use an AI system to make a plan for a robot to perform a task?						
	Not well (0)	Slightly well (1)	Moderately well (2)	Very well (3)	Extremely well (4)		
S_1=JEDAI	1	4	5	9	2	11	5
S_2=JEDAI.Ed	0	4	6	9	2	11	4
Post Q7	Do you agree that the JEDAI.Ed system made it easier for you to provide instructions to a robot for performing tasks?						
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)		
S_1=JEDAI	0	1	3	11	6	17	1
S_2=JEDAI.Ed	0	0	1	14	6	20	0
Post Q8	Do you agree that JEDAI.Ed helps improve the understanding of the robot's limitations and capabilities?						
	Strongly Disagree (0)	Disagree (1)	Neither Agree nor Disagree (2)	Agree (3)	Strongly Agree (4)		
S_1=JEDAI	0	1	4	12	4	16	1
S_2=JEDAI.Ed	0	0	5	11	5	16	0

Table 11. Post-survey questionnaire administered during the *sentiment change phase* of our user study. For each Question ID, the first (second) row is the administered question (possible answers). The third and fourth row represent the total participant answers for the question after interacting with JEDAI and JEDAI.Ed respectively.

Hypothesis	Effect Size
H1 - Increased Curiosity	0.60
H2 - Easier Programming	0.58
H3 - Improved Understanding	0.47
H4 - Helpful Explanations	0.81
H5 - Intuitive UI	0.63
H6 - Programming Confidence	0.54
H7 - Faster Solving	-0.53

Table 12. Effect Size for hypotheses evaluated in the Test Phase between users interacting with JEDAI.Ed and JEDAI. Effect size for H1-H6 was calculated using the responses on the Likert scale, while H7 was evaluated on the time taken in seconds to solve the test problem. The negative value of effect size for H7 indicates lesser time taken by JEDAI.Ed users to solve the problem.

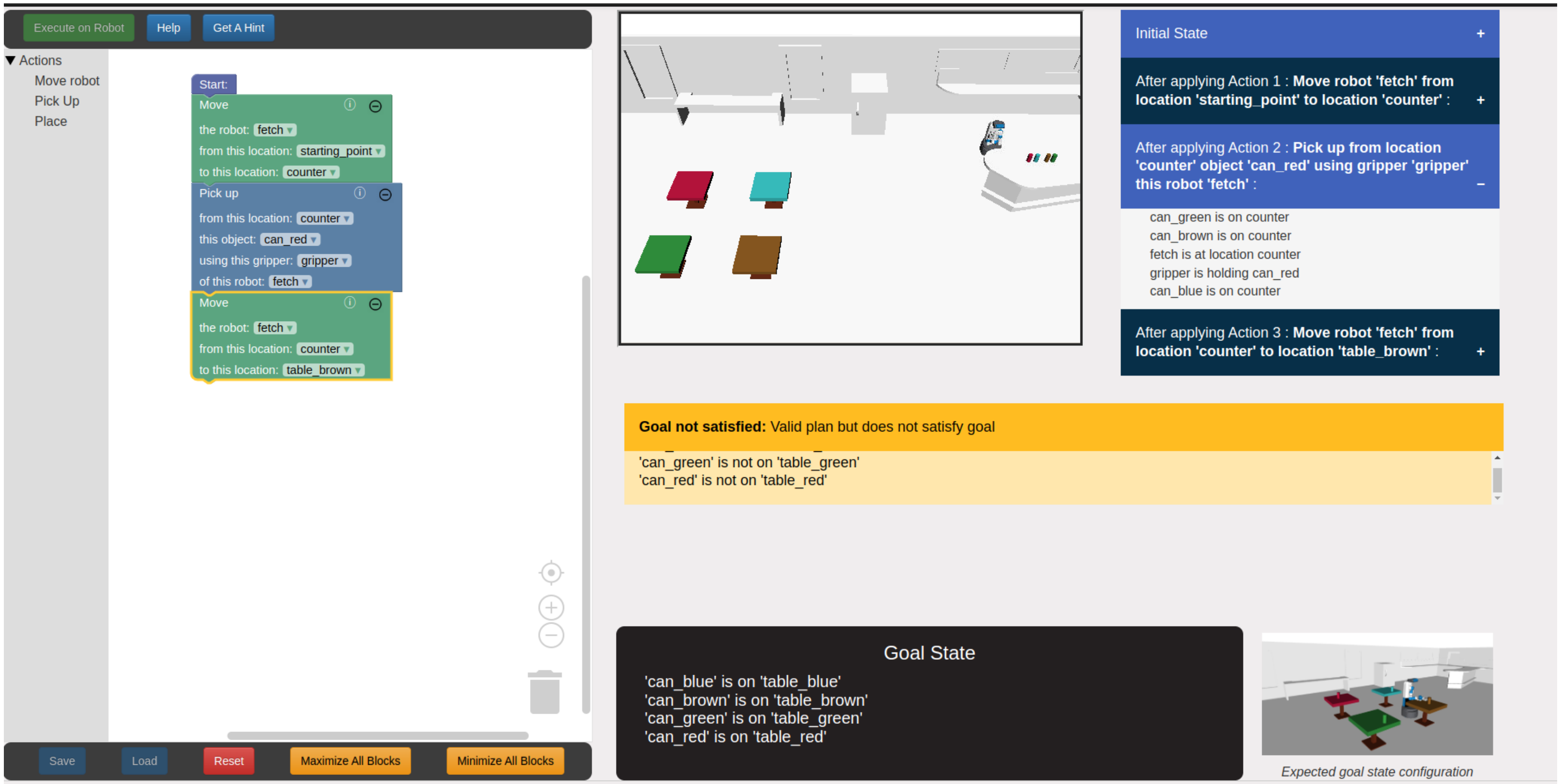


Figure 12. Screenshot of the JEDAI.Ed user interface.

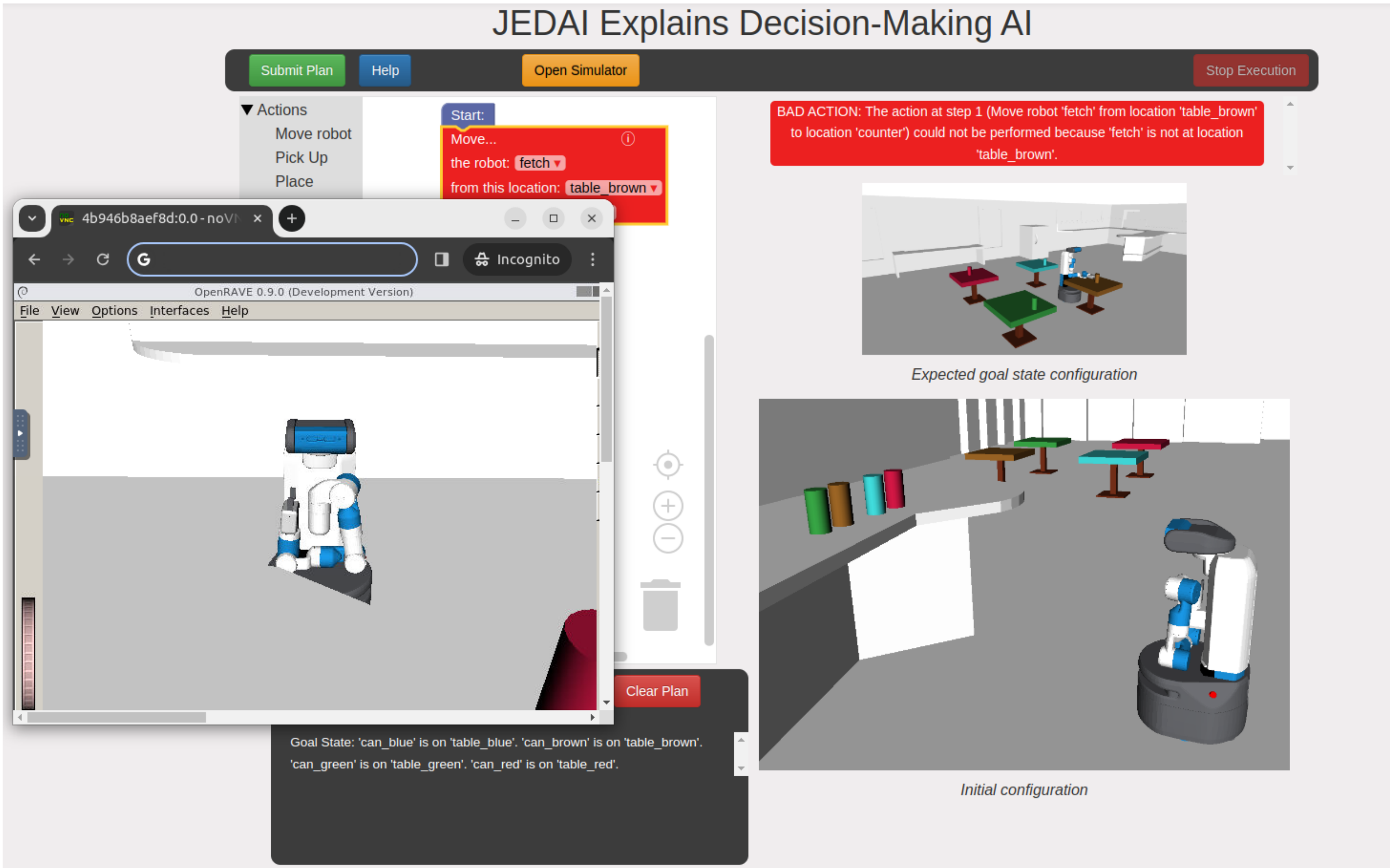


Figure 13. Screenshot of the JEDAI user interface.

APPENDIX D

PERMISSION FROM CO-AUTHORS

This work was part of a collaborative project and has been submitted for acceptance in an academic conference. Per ASU policy[2], the material under submission appears in Chapter 2 (with slight modifications). I am deeply grateful to my collaborators: Rushang Karia, Pulkit Verma, Jayesh Nagpal, Rashmeet Kaur Nayyar, Naman Shah, and Dr. Siddharth Srivastava, for their assistance with this thesis[3]. All the aforementioned co-authors have reviewed and provided their permission for the collaborative work to be used in this document.

[2] https://graduate.asu.edu/current-students/completing-your-degree/formatting-your-thesis-or-dissertation/asu-graduate-college

[3] `{rushang.karia,verma.pulkit,jnagpal1,rmnayyar,npshah4,siddharths}@asu.edu`

APPENDIX E

IRB APPROVAL FOR THE USE OF HUMAN SUBJECTS

ASU Knowledge Enterprise Development

APPROVAL: MODIFICATION

Siddharth Srivastava
IAFSE-SCAI: Computer Science and Engineering
480/727-7451
[redacted]

Dear Siddharth Srivastava:

On 2/22/2024 the ASU IRB reviewed the following protocol:

Type of Review:	Modification / Update
Title:	Adaptive Training and Education for Adaptive AI Systems
Investigator:	Siddharth Srivastava
IRB ID:	STUDY00017584
Funding:	Name: DOD-NAVY: Office of Naval Research (ONR), Grant Office ID: FP00036287
Grant Title:	None
Grant ID:	None
Documents Reviewed:	• Consent Form, Category: Consent Form; • IRB Social Behavior, Category: IRB Protocol; • Recruitment Flyer, Category: Recruitment Materials; • Recruitment Script, Category: Recruitment Materials; • User Study Questions, Category: Measures (Survey questions/Interview questions /interview guides/focus group questions);

The IRB approved the modification.

When consent is appropriate, you must use final, watermarked versions available under the "Documents" tab in ERA-IRB.

In conducting this protocol you are required to follow the requirements listed in the INVESTIGATOR MANUAL (HRP-103).

Sincerely,

IRB Administrator

Figure 14. IRB Approval For The Use Of Human Subjects.