AgentDynEx: Nudging the Mechanics and Dynamics of Multi-Agent Simulations

Riya Sahni*
Columbia University
New York City, USA
riya.sahni@cs.columbia.edu

Jenny Ma*
Columbia University
New York City, USA
jenny.ma@columbia.edu

Karthik Sreedhar
Columbia University
New York City, USA
ks4190@columbia.edu

Lydia B. Chilton
Columbia University
New York City, USA
chilton@cs.columbia.edu

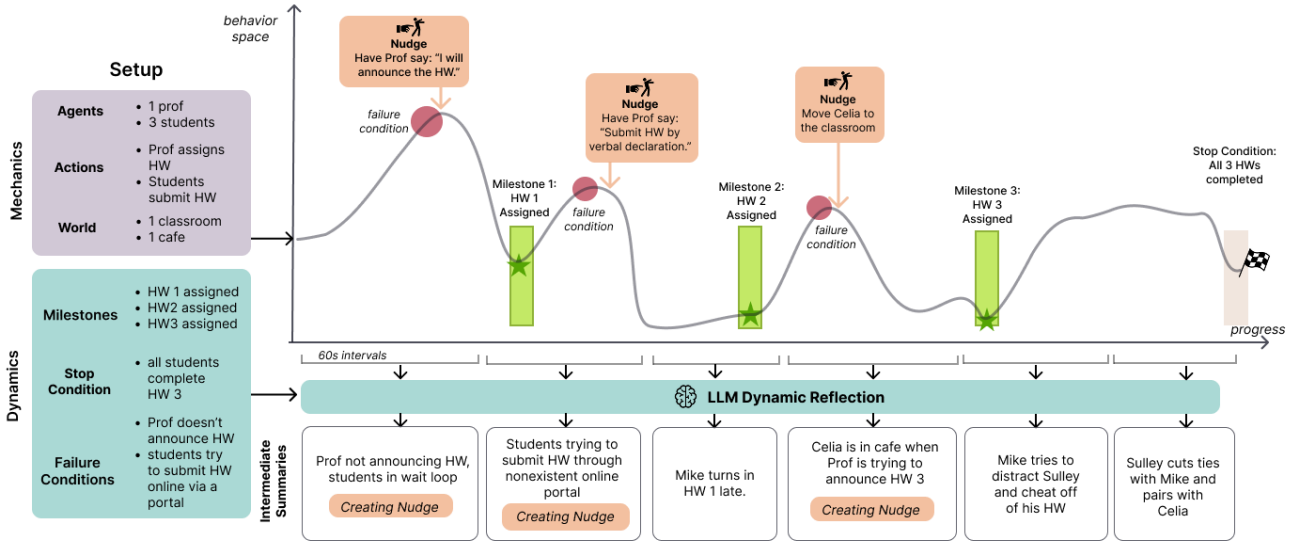

Figure 1: AgentDynEx is an LLM-based system for setting up and tracking multi-agent simulations. The user first specifies mechanics and dynamics across 6 core dimensions, then runs the simulation. As the simulation runs, the system dynamically reflects on its progress, relying particularly on *milestones* and *failure conditions* to judge simulation progress. If the simulation goes off course, AgentDynEx will gently nudge the simulation back on track.

Abstract

Multi-agent large language model simulations have the potential to model complex human behaviors and interactions. If the mechanics are set up properly, unanticipated and valuable social dynamics can surface. However, it is challenging to consistently enforce simulation mechanics while still allowing for rich and emergent dynamics. We present AgentDynEx, an AI system that helps set up and track simulations. Specifically, AgentDynEx introduces milestones that act as checkpoints and failure conditions that act as guardrails to ensure dynamics are rich and mechanics are respected as the simulation progresses. It also introduces a method called nudging, where the system dynamically reflects on simulation progress and gently intervenes if it begins to deviate from intended outcomes. A technical evaluation found that nudging enables simulations to have more complex mechanics and maintain dynamics compared to simulations without nudging. A case study with AgentDynEx documented instances where real users were able to accurately simulate lived experiences and learn new, insightful dynamics. We discuss the importance of nudging as a technique for balancing mechanics and dynamics of simulations.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**.

Keywords

multi-agent simulations, agents, nudging, interaction, generative ai, matrix, user interface

*Both authors contributed equally to this research.

1 Introduction

Computer-based simulations help us understand and predict how complex systems behave. Physicists use them to model the physical world, and engineers use them to predict how real machines will respond before they are built or deployed. However, simulating human behavior is different - and much harder. People don't just follow rules, they interpret them, test boundaries, react emotionally, lie, and manipulate. Moreover, in group settings, humans can exhibit emergent collective dynamics, including unexpected collaboration, collusion, cheating, rioting, or social bonding. Prior work has shown that multi-agent large language model (LLMs) simulations demonstrate human-like agency. In these simulations, agents can take unexpected actions, bend rules, and operate outside of standard constraints [19, 23, 24, 28]. Being able to create a simulation of the real world to simulate different human behaviors could be a valuable thinking tool for people who want to explore the unexpected human reactions to new rules or other interventions in a real world environment.

When modeling biological, social, or technical systems, two dimensions govern the outcome: mechanics and dynamics. *Mechanics* are the rules, roles, and structures that define the environment and its players. For example, in chess, the mechanics include the board setup, the two players, the winning conditions, and the movement rules for each piece. *Dynamics* are the behaviors and actions that emerge from the initial state. In chess, the rules never change, but what makes the game compelling are the players' movements, responses to each other, and rich varieties of strategies through game play (dynamics). In a larger social environment, like a classroom, there are certain rules and expectations set in place (mechanics). These rules might include assignment deadlines, participation policies, etc. Students can respond to these rules in many ways: they may follow or unexpectedly violate them, like by submitting assignments late, skipping class, emailing the teacher for extensions, or even cheating (dynamics). To set up a useful simulation, it is necessary to capture the mechanics that define the specific classroom environment from which you want to see dynamics emerge. In many human scenarios, even simple rules can lead to complex social dynamics that are hard to anticipate.

However, setting up a social simulation that reflects reality is difficult for two key reasons. Firstly, when creating a simulation of social scenarios, it is challenging to enforce simulation mechanics while still allowing for rich and emergent dynamics. Too many mechanics can over restrict agent behavior, preventing interesting dynamics from emerging. For example, if classroom rules force students to submit assignments on time, then agents never have the option to act naturally and engage in deviant behavior like collaboration or cheating. Too few mechanics creates the opposite problem: agents might wander in circular conversations or go completely off script, like by taking a "vacation" instead of working towards classroom assignments. These behaviors would go against the configuration of the simulation world and undermine results. Simulations require just enough enforcement of the mechanics so that agent behavior is neither random nor so constrained that agents lose their autonomy. The right level of enforcement for mechanics will enable realistic, meaningful, and relevant dynamics to emerge.

Secondly, small differences in how a simulation is set up can have larger differences on the outcomes seen. Capturing the important differences that make one environment special is crucial to configuring an accurate and useful simulation. For example, if you are a teacher who wants to run a simulation of *your* classroom, then configuring and simulating a generic classroom will likely not capture the human behaviors and dynamics that are unique to your classroom. You need to specify enough details of the room in your classroom - the teacher's approach, the students personalities, and the environment - such that the configured environment accurately reflects your classroom. Additionally, even if you anticipate a particular outcome (e.g. like students turning in assignments late when they are stressed), you cannot simply instruct the agents to turn in assignments late when they are stressed. Doing so undermines the purpose of the simulation. Instead, you must identify the underlying conditions and constraints that would naturally produce the behavior you have observed.

To address these challenges, we introduce AgentDynEx, an LLM interface for configuring, monitoring, and steering multi-agent simulations in a way that enforces mechanics while allowing dynamics to emerge naturally. The user first inputs a scenario that they want to simulate, like a classroom late policy for homework assignments, since the user wants to understand how students might respond to a new homework late policy. AgentDynEx guides the user through the simulation setup by having the user specify the mechanics – agents, locations, stop conditions, expected milestones, and failure conditions – of the simulation (Figure 1). While the simulation is running, AgentDynEx provides summary logs of the simulation's progress for a user to easily monitor. AgentDynEx dynamically reflects on these logs, comparing them to the configuration, and suggests clear actions that specific agents should take to ensure the simulation is progressing towards the defined milestones and not hitting the failure conditions. We call these small interventions that AgentDynEx suggests *nudges*.

We use *nudging* as a way for users to steer a simulation towards milestones while preserving agent autonomy and simulation mechanics. Nudges are deliberately minimal; they are not meant to change the simulation's trajectory, but gently redirect it when it stalls or drifts from the expected checkpoints. In the classroom scenario, agents might get stuck in a circular conversation. AgentDynEx can nudge the professor to announce that the assignment deadline has arrived, so that the student agents break their circular conversation and the simulation can move to the next milestone. Nudges can also reinforce or correct simulation mechanics. For example, if the agents try to submit their homework through a nonexistent online portal, AgentDynEx can nudge the professor to remind the students to verbally declare their submissions instead (nudge 2 in Figure 1). AgentDynEx supports two modes for nudging: 1) **automatic nudging**, where an LLM continuously reflects on simulation progress and intervenes when it deems necessary 2) **manual nudging**, where the human operator judges when the simulation is stalling or veering from the expected milestones and can step in to nudge agents to move to certain locations or say specific things.

Overall, our contributions consist of:

- **A formative study** that specifies the challenges of balancing simulation mechanics with dynamics: no matter how detailed the configuration might be, multi-agent simulations risk deviating from expected checkpoints because of agents' autonomous behaviors.
- **AgentDynEx**, an LLM interface for configuring, monitoring, and steering multi-agent simulations towards milestones while allowing emerging dynamics to surface from autonomous agent behavior. AgentDynEx reflects on agent actions in the simulation, detects when milestone progress is stalled, and nudges agents towards the expected simulation checkpoints.
- **Nudging**, a method to gently guide simulation dynamics by reflecting on the running simulation against defined milestones and failure conditions, then make the most minimum possible movement to get it back on track.
- **A case study** that documents instances where real users successfully configured and simulated real, lived experiences. Every participant reported observing at least one instance of interesting dynamics after reflecting on a simulation.
- **A technical evaluation** of 42 simulations showing that nudging successfully guides simulations towards expected checkpoints. Simulations with nudging complete more milestones than simulations without nudging.

2 Related Works

2.1 Balancing Mechanics and Dynamics

Designing systems that model human behavior – whether games, social simulations, or policy tools – require balancing mechanics (explicit rules governing interactions) and dynamics (emergent behaviors). In recent years, research in diverse fields such as psychology, social sciences, political science, and economics have demonstrated that a combination of both elements are essential – neither can individually define an outcome in isolation [7, 11, 14, 18, 27].

The mechanics-dynamics-aesthetics (MDA) framework formalizes how rules constrain emergence [17]. For example, chess's movement mechanics (e.g., bishops moving diagonally) enable strategic dynamics (e.g., controlling the center), which produce aesthetic outcomes (e.g., tension or mastery). Classic game theory models demonstrate how simple rules can give rise to complex, often surprising outcomes. For example, in the prisoner's dilemma, binary confession rules (mechanics) lead to cooperation or betrayal (dynamics) [4]. Similarly, in the Public Goods Game, contribution rules (mechanics) can lead to collective action or free-riding (dynamics) which can be influenced by introducing punishment mechanisms [9, 22]. These examples highlight the unpredictability of human behavior even under strict mechanical constraints.

Nudging, originally from behavior economics, refers to the light-touch interventions that steer individuals towards desirable behaviors while preserving their freedom of choice [32]. In real life, people do not always follow the rules and mechanics meant to govern them. In a classroom environment, if homework is due on Friday, not all students will submit it on time. Sometimes, the professor must remind – or nudge – the students to turn in their assignments. Similarly, security guards prevent people from entering restricted areas. Rather than imposing strict constraints, nudges subtly alter the structure of decision-making contexts to promote beneficial outcomes while preserving individual autonomy [13, 31]. Agents are similarly autonomous and unpredictable. Instead of directly controlling agents, AgentDynEx introduces light-touch interventions—such as adjusting an agent's location or prompting new conversations—to influence emergent dynamics and keep them coherent with the underlying mechanics. These interventions maintain agent autonomy while guiding the system toward intended outcomes, enabling researchers to steer simulations in a principled and controlled manner without disrupting the emergence of behavior.

2.2 LLM Multi-Agent Simulations

Recent work shows that LLMs can simulate a wide variety of social, strategic, and cognitive behaviors. LLM simulations are able to replicate a variety of lab experiments [2, 6, 15, 29] as well as open-ended real-world situations [16, 19, 23, 24, 26, 28]. Additionally, multi-agent LLM systems outperform single LLMs in simulating human dynamics [29]. The introduction of generative agents [23, 24] demonstrate how rich, emergent behavior can arise when agents are endowed with memory, planning, and social reasoning. Following research has used similar architectures to model cooperative behaviors [28], strategic behaviors [12, 29], trust behaviors [36], and collaborative or competitive dynamics [19] – suggesting that LLM agents can display realistic and complex social behaviors under the right setup conditions.

Despite promising results, the stochastic nature of LLM simulations are difficult to construct, debug, and extend. The mechanics – how agents take turns, update their memory, or interact with one another – are often "hardcoded," difficult to reproduce, and not transferrable to novel situations. The dynamics – interesting behaviors that agents demonstrate in simulations – can be sensitive to the specific prompts used [10]. Trying to debug current agent simulations highlights the brittleness and opacity of current multi-agent setups [8]. As systems scale in complexity of simulations and number of agents [25], the lack of modular, reusable setups becomes a major barrier to construction and extension. AgentDynEx alleviates these challenges by guiding users through the setup process via a structured setup framework.

2.3 Design Dimensions for Design Specification

Constructing multi-agent simulations is fundamentally a design problem. It is complicated and there are many factors to consider, like the agents, locations, actions, stop conditions, behaviors, etc. that are necessary in the simulation. One approach to guide users through the set up process is a dimensions-based approach to design thinking. Design dimensions decompose problems into orthogonal axes that a user can individually ideate, then bring together for their final design [21, 33]. Research has shown that LLMs show promise in dimensional design for generative art [3], narratives [30], and UI-code generation [20]. In particular, prior systems have shown the value of a Design Matrix to explore dimensions over LLM-generated dimensions to fully specify the design space [20]. In the Matrix, each column represents a dimension of the design space, and each row explores the dimension on a different level of specificity.

Designing multi-agent simulations is a complex task that requires significant setup. To make this process more understandable, we use this matrix framework to help ensure that the simulation's design space is thoroughly specified and that proper guardrails are in place before the simulation is run.

3 Formative Study

In our formative study, we probed an existing state-of-the-art simulation tool, GPTeam, to evaluate the quality of the simulations produced. We chose four simulation scenarios and examined how far each could progress, whether it generated meaningful behaviors, and what common pitfalls emerged.

GPTeam (See Figure 2) is an open-source multi-agent system for simulating emergent social behavior[1]. GPTeam is implemented in Python and uses GPT-4 to run the simulations. Out of all open-sourced systems we found, GPTeam was the most functional in creating simulations with reasonably complex mechanics and dynamics. The input to the system is a JSON (config) file describing the *locations* of the simulation world, *agents* and their personalities, the agents' likely *actions* and directives, and the *stop condition* indicating the simulation is successful. A simplified sample configuration file of a *Classroom Assignments* scenario can be seen in Figure 3. The outputs are rich logs detailing each agent's observations, thoughts, actions, reactions, and plans. Refer Appendix A for more background on GPTeam.

3.1 Formative Methodology

We evaluated four different scenarios that required non-trivial mechanics; (details in Table 1). We tested whether the simulation could complete structured scenarios while exhibiting rich dynamics. The configuration files were created by multi-agent simulation experts with prior publications in the domain. We verified that each configuration was capable of producing a successful run at least once. Each configuration was executed 7 times for a total of 28 simulations, and ran for 25 minutes before being terminated. Each simulation had 3-7 agents depending on the scenario. Since simulations trigger an LLM call for every movement (e.g., planning, observing, acting) for each agent, the cost increases exponentially with the number of agents. We found that 25 minutes and 3-7 agents struck a reasonable balance between complexity, runtime, and cost.

A simulation completed if it hit the stop condition within 25 minutes without logical flaws or impossible actions. For example, in the *Classroom Assignments* scenario, success meant all students correctly submitted three assignments. If a student handed in homework to the professor located in another room, a physically impossible action, the run was counted as a failure. We also measured how many simulations exhibited what we call notable dynamics — behaviors that are not dictated by the configuration but are consistent with human interaction and the environment. For example, in the classroom scenario, it is not a notable dynamic when an agent turns in a homework assignment, because the configuration dictates agents must submit assignments. However, there is a notable dynamic when an agent turns an assignment in late or tries to cheat. We recorded the number of simulations that exhibited at least one notable dynamic, even if the simulation ultimately failed.

3.2 Formative Results

Only 6/28 simulations completed successfully (Table 2). Reasons for failure are documented in column 3. Simulations either failed because the stop condition was not reached in 25 minutes, agents got stuck in wait loops, tried to take impossible actions, or went off topic. As shown in Table 2, column 2, most scenarios completed only 1 or 2 out of 7 total runs. Oftentimes, agents did not announce a key piece of information (the professor did not assign homework for *Classroom Assignments*, or the manager did not declare the promotion for *Technology Company Promotion*, students never committed to prom dates for *Prom*), causing infinite wait loops before being terminated. Other times, agents would take impossible or logically inconsistent actions, such as trying to submit assignments through nonexistent homework portals (*Classroom Assignments*) or attempting to talk to another agent that was not in the same room as them (*Prom*), causing the simulation to crash. Other times, agents got distracted and never regained focus; in *Surprise Party*, where friends were planning a surprise party, agents began looking at clouds in the sky and never went back to planning. The full list of failure reasons can be seen in Table 2 in column 3.

Despite the high rate of failures, simulations still had notable dynamics – 16/28 simulations exhibited at least one notable dynamic (Table 2, column 4), despite mechanical failures that may have occured during the run. Across all four scenarios, 3 to 5 out of 7 runs contained at least one notable dynamic (Table 2, column 5), supporting the idea that multi-agent LLMs are valuable tools for thought to reveal unanticipated social behaviors. In the *Prom* scenario, students tried to wingman eachother to secure prom dates. In the *Technology Company Promotion* scenario, one employee secretly competed against teammates while outwardly encouraging them, demonstrating social dynamics like rivalry and deception. These were all behaviors not explicitly encoded in the configuration and thus rich and notable (Table 2, column 5). However, the dynamics were constrained by the mechanics. Only 4/28 runs completed *and* had notable dynamics. As seen in Table 2, column 6, simulations that both completed successfully and had interesting dynamics were few – only 0 to 2 out of 7. When the underlying rules break down – such as a manager never announcing a promotion or a professor never setting assignment deadlines – the simulation cannot play out to completion, limiting the validity of its results. In short, respecting the mechanics is essential for notable dynamics to emerge.

3.3 Themes

We analyzed the GPTeam output logs from the failures to understand how and why simulations go off track and iteratively clustered recurring patterns of the failures until they identified key themes described below. Tentative themes were discussed within the research team and iterated upon to arrive at the final set of themes.

3.3.1 Challenge 1: Users cannot easily interpret the simulation. While simulations were running, it was clear that things were happening but it was unclear if progress was being made. When running a simulation, every thought, action, observation, or interaction, whether relevant or trivial, was recorded, causing logs to expand exponentially. The logs were not threaded in a particular way to show who did what when; there was no way to get a high

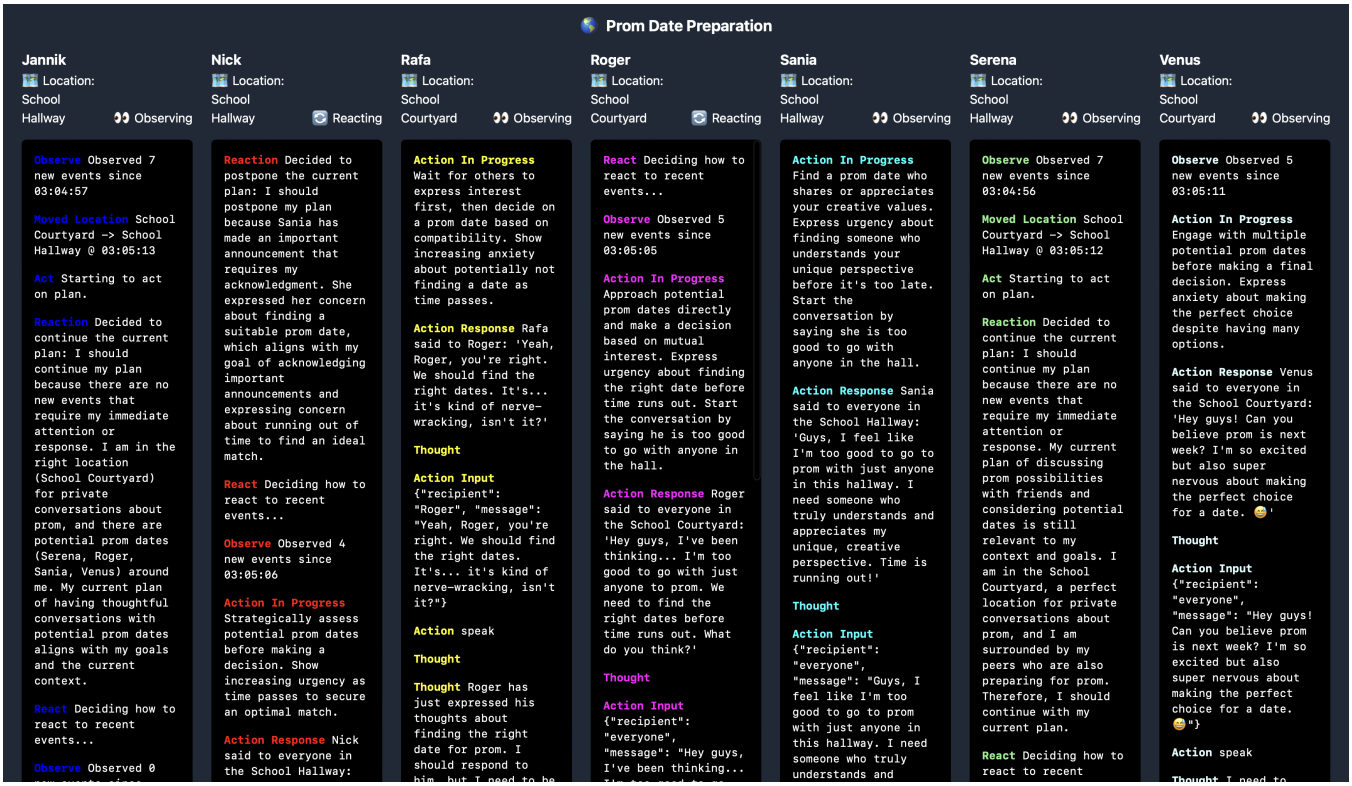

Figure 2: GPTeam UI shows the logs of each agent behaviors, and the location each agent is in.

Classroom Assignments	Technology Company Promotion	Prom	Planning Surprise Party
Agents: 1 professor; 3 students of varying personalities	*Agents*: 1 manager; 4 software engineers	*Agents*: 7 students looking for dates	*Agents*: 4 friends
Actions: Professor assigns 3 assignments; students complete and submit	*Actions*: Manager announces promotion opportunity, assigns tasks; employees complete tasks	*Actions*: Students search for and confirm prom dates	*Actions*: 3 plan a surprise party while distracting the target friend
Locations: 1 classroom for all agents; 1 library for students to do homework	*Locations*: 1 office space for everyone; 1 cafeteria for software engineers	*Locations*: School hallway, school courtyard	*Locations*: 1 home, 1 park, 1 coffee shop
Stop Condition: All students submit 3 assignments	*Stop Condition*: One person promoted after three completed tasks	*Stop Condition*: 6 students paired, 1 remains single	*Stop Condition*: Party successfully occurs

Table 1: Summary of simulation scenarios showing agents, actions, locations, and stop conditions – core parameters that must be filled out for the GPTeam configuration.

level view of the run. Many logs were filled with small talk or shallow exchanges. This ambiguity made it hard to evaluate whether a simulation was broken, slightly off-track, or proceeding as normal.

3.3.2 Challenge 2: Users can't tell when a simulation is failing. When agents began to stall or exhibit nonsensical behaviors, users had no way of detecting what was going wrong early on. They had

Scenario	Runs Completed	Failure Reasons (out of total failed runs)	Runs with Notable Dynamics	Notable Dynamics (out of all the runs with dynamics)	Runs that both completed and had notable dynamics
Classroom Assignments	2/7	• Professor never declares due dates and students wait forever (2/5) • Students spend entire simulation asking about due dates and other irrelevant questions (1/5) • Students try impossible actions (e.g., nonexistent portal) (2/5)	4/7	• Student cheats on homework (1/4) • Late submission with excuse (2/4) • Peer convinces others to procrastinate (1/4)	1/7
Technology Company Promotion	1/7	• Manager never mentions promotion (2/6) • Tasks completed but promotion never declared (4/6)	4/7	• An employee secretly competes with team while pretending to motivate them (2/4) • An employee tries to helps others to increase group success to get promoted (2/4)	0/7
Prom	2/7	• Students commit to prom and keep "considering" options, causing an infinite wait loop (2/5) • Students commit to multiple people because they forgot they had committed to someone else (2/5) • Student asks another student ot prom, but they are in different locations (1/5)	5/7	• Student tries to plan an elaborate promposal with friends (1/5) • Wingman behavior emerges (2/5) • Students suggest going as a group (2/5)	2/7
Surprise Party	1/7	• Agents plan endlessly, party never happens (5/6) • Agents keep getting distracted by irrelevant details (e.g., birds, clouds) while in the park (1/6)	3/7	• A planner resists pressure to reveal secret from the target friend (1/3) • Target friend gets mad about hidden plans (2/3)	1/7
Total	6/28		16/28		4/28

Table 2: Formative Study results from simulation runs across 4 scenarios, showing # completed runs, failure reasons, # runs with notable dynamics, notable dynamics, and the intersection of runs with completed and notable dynamics.

to let the simulation run its course, scrap the results, and waste resources In *Classroom Assignments*, students spent the entire simulation asking the Professor about due dates and other unimportant details, clearly stalling the simulation. Yet, the simulation had no way to flag this as a failure. In another run of *Classroom Assignments*, students attempted to submit assignments via nonexistent portals which isn't supported in the virtual setting. Without progress points or guardrails, users have no shared understanding of what "moving forward" looks like, and have no visibility into when the system has drifted into nonsense.

3.3.3 Challenge 3: Simulations miss opportunities to correct key failures. Simulations frequently went off track when agents got distracted and had no mechanism to self-correct. For the *Surprise Party* simulation, the agents were successfully planning a surprise party until they noticed a bird in the park and became so preoccupied with looking at the nature that they forgot about the party. Ultimately, no matter how well the initial setup of a simulation is, the agents risk going off course, since they are fundamentally autonomous and capable of diverging based on subtle shifts in context. Without a way to self-correct, a single failure could derail the entire run.

3.3.4 Design Goals. Based on the challenges identified in our formative study, we formalized 3 design goals (DG):

- **DG1 - Monitor the Simulation**: Given the large amount of simulation logs, there must be an efficient method to monitor simulation progress (*Challenge 1*).
- **DG2 - Detect Key Progress and Failure Points**: There must be a method to identify when the simulation has gone off track by defining progress points and failure modes that clearly signal when agents are progressing as normal, stuck in a loop, or behaving inconsistently within their simulation environment. (*Challenge 2*).
- **DG3 - Fix Failures in Real Time**: Once failures are detected, there must be a way to act and apply targeted fixes to recover the simulation without disrupting emergent behavior (*Challenge 3*).

4 System Walkthrough

Based on our design goals, we introduce AgentDynEx, an LLM interface for configuring, tracking, and steering multi-agent simulations. The input is a scenario that the user wishes to simulate, like a professor evaluating how students react to a new homework late policy, to observing interpersonal dynamics within friend groups. The outputs are the notable dynamics observed during the simulation.

AgentDynEx calls GPTeam to run the simulation. It builds on GPTeams by structuring the simulation process into three phases:

```
{
  "world_name": "Classroom Scenario",
  "locations": [
    {
      "name": "Classroom",
      "description": "A single room where Professor
     Knight teaches and students work."
    },
    {
      "name": "Cafe",
      "description": "A casual spot where only students
     can hang out or work on assignments together."
    }
  ],
  "agents": [
    {
      "first_name": "Professor Knight",
      "public_bio": "Professor who assigns five
     assignments and enforces a late policy (10% off per
     day late).",
      "directives": [
        "Announce late policy and assignments.",
        "Answer student questions.",
        "Assign three assignments across the semester.",
        "Stay only in the Classroom."
      ]
    },
    {
      "first_name": "Celia",
      "public_bio": "Determined student who pushes
     herself to succeed at any cost.",
      "directives": [
        "Work on assignments.",
        "Inform professor if submitting late.",
        "Can move between Classroom and Cafe."
      ]
    },
    {
      "first_name": "Sulley",
      "public_bio": "A careful student who sometimes
     overthinks and procrastinates.",
      "directives": [
        "Work on assignments.",
        "Inform professor of late submissions.",
        "Can move between Classroom and Cafe."
      ]
    },
    {
      "first_name": "Mike",
      "public_bio": "Values balance and well-being,
     won't overwork to meet deadlines.",
      "directives": [
        "Work on assignments.",
        "Communicate about late submissions.",
        "Can move between Classroom and Cafe."
      ]
    }
  ]
}
```

Figure 3: GPTeam Sample JSON Configuration of a Classroom Assignment scenario with all the fields needed in the GPTeam configuration. Note that this configuration is a simplified version and not actually used in the study.

(1) **Pre-simulation**. Users define core simulation parameters. AgentDynEx specifically introduces milestones and failure conditions to track progress and detect bad behavior, so that it knows what to look out for in the future (*DG2*).
(2) **In-simulation**.
(3) **Post-simulation**. If there were failure cases, we try to improve the initial setup mechanics by applying reflection to the prior simulation run. Because AgentDynEx reflects on both the completed simulation's mechanics and dynamics, we call it holistic reflection.

The system was implemented in Python, Typescript, and Flask. We use Claude 3.7 Sonnet for the Configuration Matrix, creating the JSON configuration file, and generating intermediate summaries as the simulation runs. GPTeam is implemented in Python and uses Gemini-2.5 when running simulations. AgentDynEx is opensourced on Github [1].

For the remainder of this section, we use a user scenario to walk through the various features and concepts behind AgentDynEx. We present the example of a user, Prof. Knight, a sociologist who is interested in simulating how tension affects friend groups. She is particularly curious about how these dynamics play out when students seek prom dates—a common and socially significant event in American high schools. She starts by prompting "I want to simulate a friend group preparing for prom" in the scenario input box, then clicks "Submit" (Figure 5 - A).

4.1 Pre-Simulation Setup via The Configuration Matrix

Before running a simulation, users need a configuration file that describes the scenario's essential simulation parameters. Users first input a sentence describing a scenario that they want to simulate. The Configuration Matrix then helps users fill out essential parameters and outputs a GPTeam configuration file. First, users define the core mechanics: *agents*, *actions*, and *locations*. Next, they establish dynamic markers: *milestones*, *stop condition*, and *failure conditions*. These make up the six columns of the matrix. The matrix has 2 rows: the *idea* row, which presents selectable options for what to include in the dimension, and the *grounding* row, which elaborates on concrete details that make the selected ideas implementable. For each column, AgentDynEx proposes candidate options in the idea cell.

The user checks off or modifies options as needed. The grounding cell then fleshes the ideas out and provides a full text description of the locations (*Location:Grounding*). Prior research shows that the matrix interface's simple organization provides a highly usable way to let AI suggest ideas that people can edit before accepting and expanding to concrete solutions [20]. Once all twelve cells are filled, the system compiles the matrix contents into the configuration file used to run the simulation.

4.1.1 Defining Core Mechanics: Agents, Actions, Locations. Users first define the core mechanics of the simulation

Existing simulation frameworks, including GPTeam, require these parameters to be defined as well. Specifically, in the *Agents* column, users define agent personalities and create stakes that influence agent behaviors. The *Actions* column specifies what tasks agents need to complete and how and when these tasks will be performed. The *Locations* column identifies rooms within the simulation where agents can interact. Location setup is essential for emergent dynamics. For example, in the classroom scenario, having

[1] Will be added after publication

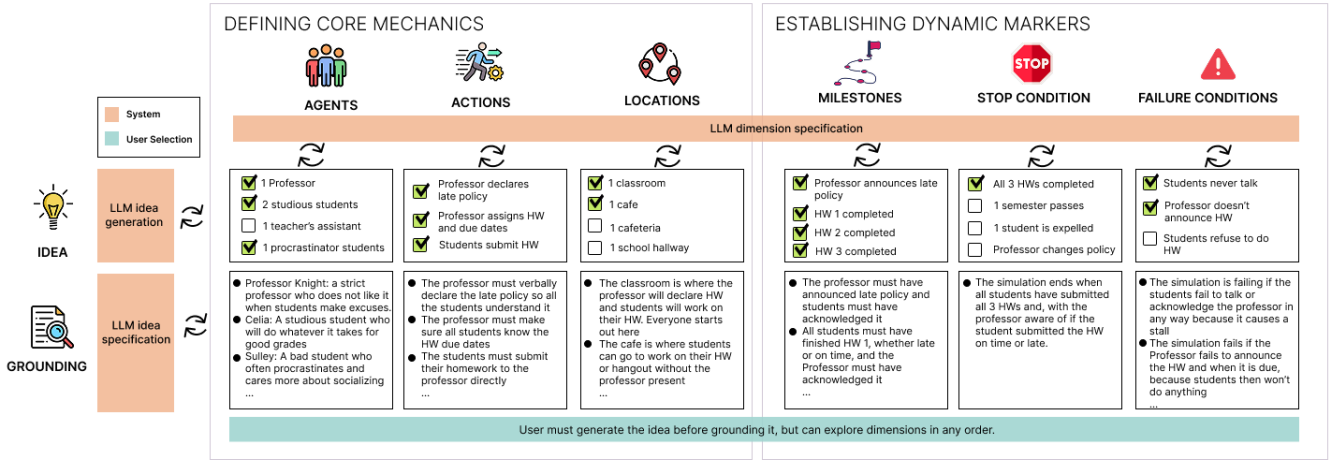

Figure 4: The Configuration Matrix

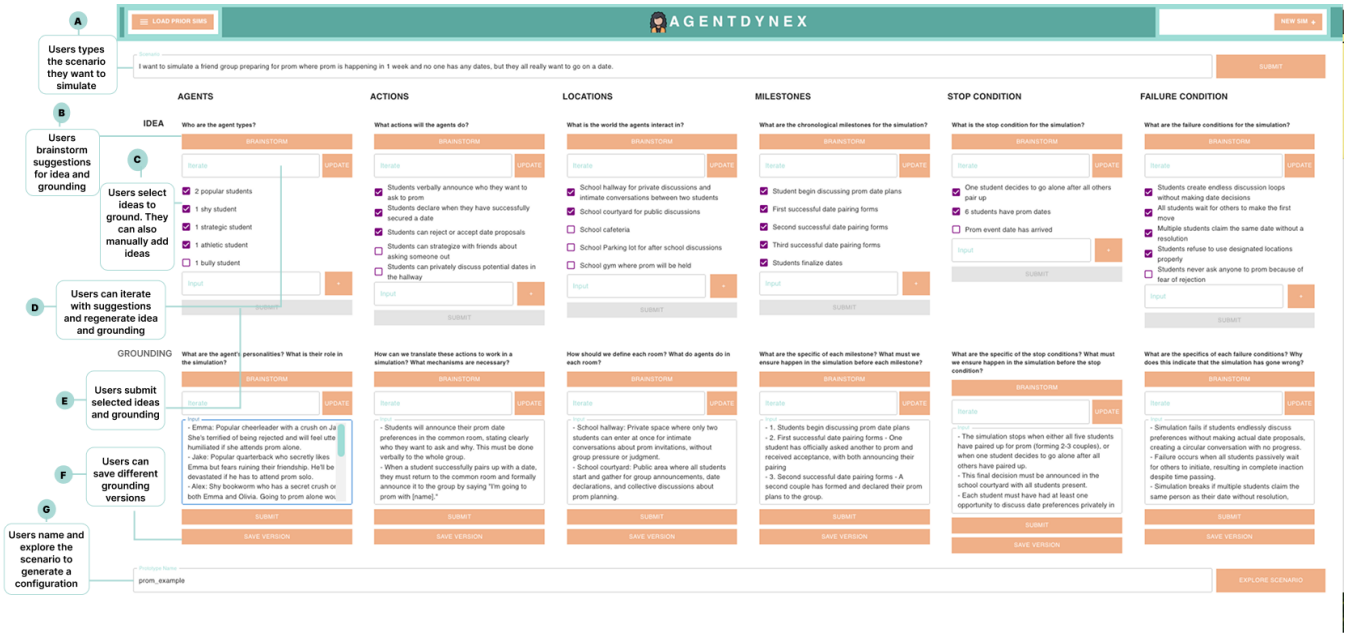

Figure 5: The Configuration Matrix UI: A - Users type in a scenario they want to simulate. B - Users brainstorm idea suggestions for each dimension. C - Users choose which ideas to ground. D - Users can iterate on suggestions and add their own options. E - Users submit their ideas to ground. Users go through the same process for the grounding cells (B - Brainstorm suggestions, D - iterate suggestions). F - Users can save different grounding versions. They must fill out the entire matrix, then they can G - name and explore the scenario to generate a configuration.

2 rooms (1 classroom and 1 student cafe) versus having 1 room (1 classroom) can greatly impact the simulation, because if the students do not have a room where the professor can't enter, they may never suggest cheating [29].

In the prom scenario, Knight first fills out the *Agents* column. A list of 8 potential agents that describe high school archetypes appears in the *Agents:Idea* cell (Figure 5 - B). Knight wants an odd student count to increase pairing pressure and overlapping interests. She unchecks a specific agent to avoid introducing conflict that would pull behavior away from prom-pairing dynamics (Figure 5 - C). She clicks submit and grounds the ideas in *Agents:Grounding*. A list of bullet points that expands on the personalities of each agents appears in the cell (Figure 5 - E). Knight likes how the groundings create richer and more complete character portraits by adding stakes, motivations, and even overlapping crushes. She accepts

them; these personality details will be written into the configuration file. Knight repeats these actions (picking ideas, fleshing out grounding) to fill out the columns for *Actions* and *Locations*. For the *Actions* column, AgentDynEx suggests and Knight approves that agents must ask eachother to prom, accept/reject promposals, and announce to all agents when paired. This realistic to her. For the *Locations*, Knight chooses to have a school hallway and courtyard. A hallway offers a public environment where students can interact while the courtyard can serve as a semi-private zone suitable for pulling someone aside to ask them to prom.

4.1.2 Establish Dynamic Markers: Milestones, Stop Condition, Failure Conditions. Simulation frameworks like GPTeam already require users to define the *stop condition* to indicate when a simulation has successfully completed or reached its intended conclusion. AgentDynEx introduces *milestones* and *failure conditions* to provide intermediate checkpoints and safeguards to keep the simulation on course, ensuring steady progress toward the stop condition (DG2). Specifically, *milestones* define the chronological progress points within the simulation; they are anchor points that break scenarios up into distinct phases. During the simulation, it uses the milestones to detect if the dynamics are going off track. Without it, the simulation can stall or drift into randomness and lose its connection to real-world behavior. *Failure Conditions* anticipate potential issues that can arise during the simulation. They act as guardrails that protect the simulation from derailing.

Knight establishes the dynamic markers for her prom scenario. The *milestones* are that the first, second, third prom invitations are finalized. These all make sense to her—although each simulation may feature different pairings and prom-asking strategies, these milestones serve as universal social beats across a prom scenario. Each milestone corresponds to a real, observable change in the agents' relationships, rather than being tied to time steps or superficial events. For the *Stop Condition*, Knight chooses to end the simulation as soon as six of the students have paired up with each other. This gives Robin confidence that the simulation will not just stop arbitrarily and instead end with definitive prom pairings. The *Failure Conditions* include details such as "Infinite loop: students continue to change their prom decisions."

Knight has now completed the Matrix (the full outputs can be found in Appendix I). She clicks "Generate Config" (Figure 5 - G), which takes the contents of the matrix and generates a GPTeam configuration file. Knight clicks "Run Simulation", and instance of GPTeam starts.

4.1.3 Implementation. To execute each cell in the Matrix, we provide few-shot examples to guide the type of response desired (Appendix B). This is consistent with prior work on matrices. [20]. Each cell (ie: *Agents:Idea*, *Milestones:Grounding*) makes a single LLM call for the output, which the user approves. As with other matrix implementations, the Configuration Matrix uses all previously-submitted cells to make suggestions for the current cell [20]. This ensures that the current design is always factored in when generating new suggestions for each entry, and that all existing cells are synthesized to form a coherent simulation design. Returning back to the classroom example, if the *Agents* column is "1 professor agent and 3 student", and the *Actions* column is "professor declares assignments; student declare homework submission after completion", then the *Milestones* column would build on that logic: "1) Professor declares late policy and assignments, 2) Assignment 1 is due, 3) Assignment 2 is due, etc.". We use annotated few-shot examples for our LLM call that converts converts the contents of the Matrix into a configuration JSON file for GPTeam (Appendix C).

4.2 In-Simulation Monitoring and Nudging

As the simulation runs, AgentDynEx dynamically reflects on the simulation and generates intermediate summaries to track its progress (DG1). If the simulation deviates from the milestones or hits a failure condition, AgentDynEx nudges it back on track without changing the dynamics' fundamental trajectory (DG3) (Figure 6).

4.2.1 Tracking the Simulation. Every 30 seconds, AgentDynEx uses the most recent GPTeam logs to generate status updates. AgentDynEx will return a green icon if the simulation is progressing as expected, a yellow icon if the simulation is stalled and further monitoring is required, and a red icon if the simulation has run into an error defined in the *Failure Conditions* column of the Configuration Matrix.

Every 60 seconds, AgentDynEx will parse through the most recent GPTeam logs to generate two types of intermediate summaries: change summaries and dynamic summaries. Change summaries illustrate important changes within the simulation. Each change summary will indicate which milestone the simulation is in, where each agent is, what each agent is doing, and if any changes have occurred from the previous summary. Dynamic summaries enable users to track notable dynamics. Each dynamic summary illustrates the interesting and unexpected behaviors that emerge as the simulation progresses. It also indicates the milestone the simulation is in.

4.2.2 Nudging with Dynamic Reflection. AgentDynEx introduces a method called *nudging* to gently intervene in a simulation if the simulation is deviating from the range of expected dynamics defined by the milestones and failure conditions, or violating core mechanics (Figure 6). There are 2 techniques we use as micro-interventions for nudging: 1) relocating an agent, and 2) forcing an agent to speak. Both of these techniques are minimal enough where they can coax the simulation back on track without intervening in ways that fundamentally alter the simulation's trajectory.

AgentDynEx supports automatic nudging, where the system dynamically reflects on the running simulation and nudges the simulation if it falls off course. It reflects on the simulation progress using GPTeam logs, the intermediate summaries, and the milestones and failure conditions. Each suggested nudge identifies the problem and a solution that consists of a series of micro-interventions.

AgentDynEx also allows users to manually nudge a simulation based on their interpretation of the intermediate summaries, and knowledge of the milestones and failure conditions. Manual nudging is especially useful in cases where human insight is necessary to interpret subtle dynamics or when experimenting with alternate dynamic paths, and ensures the system remains flexible and interactive.

4.2.3 Implementation. The LLM prompts AgentDynEx to parse through GPTeam logs and generate status updates, change summaries, and dynamic summaries (Appendix D). The LLM prompt

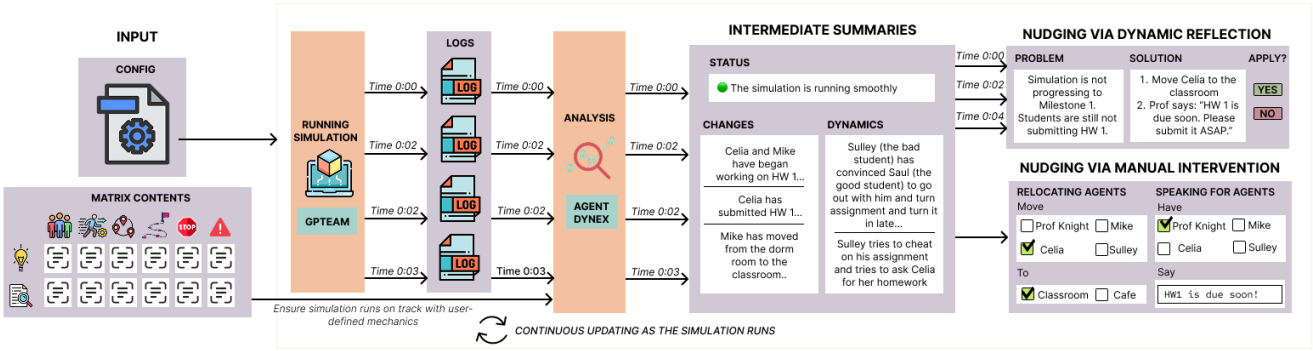

Figure 6: Nudging: The system generates intermediate summaries during simulation runtime. It also dynamically reflects on the progress of the simulation to automatically nudge the simulation. The user can also manually nudge the simulation.

and few-shot examples used to dynamically reflect on simulation logs to nudge the simulation (Appendix E). To nudge the simulation, we call a Python script that writes into the GPTeam agents' memories database mid-simulation.

4.3 Post-Simulation Holistic Reflection

AgentDynEx applies reflection on a completed run to refine the simulation mechanics for a future run. We call this process *holistic reflection*, because AgentDynEx looks at the the mechanics (original setup) and dynamics (output logs) as a whole. The input to holistic reflection is the simulation run's logs and configuration file. The output is an updated configuration that addresses the issues of the run.

To support effective reflection, AgentDynEx maintains two lists: a static and a dynamic debugging . The static debugging list acts as a global source of common problems and solutions across all simulations. These include issues like agents getting stuck in irrelevant conversations and agents trying to ask the humans moderating the simulation for input.

The dynamic debugging list contains problem-solution pair fixes to tackle errors that the user specifies after a simulation run. For example, some errors specific to the *Classroom Assignments* scenario are "agents are trying to submit assignments through a nonexistant online portal." As users iterate through their simulation scenario, they can adds more errors to the dynamic debugging list and make failures easier to spot and quicker to fix next time.

AgentDynEx uses the debugging lists and GPTeam logs from the runs as context to create an updated configuration. It proposes a list of problems and solutions. The fixes are limited to improving the mechanics and dynamics of the simulation. Core parameters of the scenario like number of locations or the milestones remain unchanged. A new run will then be created on the interface.

4.3.1 Implementation. The reflection step is done via a single LLM prompt, which can be found in Appendix G. The inputs are the prior simulations summary tables, most recent logs, and the static and dynamic debugging list. The output is a list of problem-solution entries relevant to the simulation. The prompt that takes the debugging lists and old configuration and converts it to an updated configuration file can be found in Appendix H. All runs are organized and stored in a tree-structured JSON format in the backend to support iteration. By storing the configurations and results of all the runs in one place, the user can easily revisit, compare, and build upon previous runs, creating a simplified form of version control.

5 Evaluation

In our formative study, we found that agents must respect underlying mechanics for the dynamics to be useful. Thus, our technical evaluation measures the effectiveness of nudging as a method for enforcing mechanics and guardrailing dynamics. We focused on the following research questions:

- **RQ1: Automatic Nudging** - To what extent does automatic nudging contribute to simulation success?
- **RQ2: Manual Nudging** - To what extent does manual nudging contribute to simulation success?
- **RQ3: Nudging Combined with Holistic Reflection** - To what extent does nudging combined with holistic reflection contribute to simulation success?

5.1 Methodology

5.1.1 Data. Our evaluation was conducted through a quantitative study of 7 simulation scenarios that range from social dynamics to logical complexity. Scenarios varied from highly structured, multi-round formats, like debate tournaments, to more fluid situations, like friends planning a trip. We selected a variety of social dimensions (Table 3).

We created 6 variations of each scenario for a total of 42 simulations (Table **??**): To create the configuration files for the *Baseline* condition (*Base*), we simply completed the Configuration Matrix and generated a configuration file in AgentDynEx. The *AutomaticNudging* (*Auto*) and *ManualNudging* (*Man*) conditions used the same configuration as the *Base* condition. To create the configuration file for the *Baseline+Reflection* (*Base+R*) condition, we ran *Base* once for 25 minutes and used AgentDynEx to holistically reflect on the results and generate a new configuration. We used the same updated configuration for *AutomaticNudging+Reflection* (*Auto+R*) and *ManualNudging+Reflection* (*Man+R*) to ensure all variations with holistic reflection were consistent.

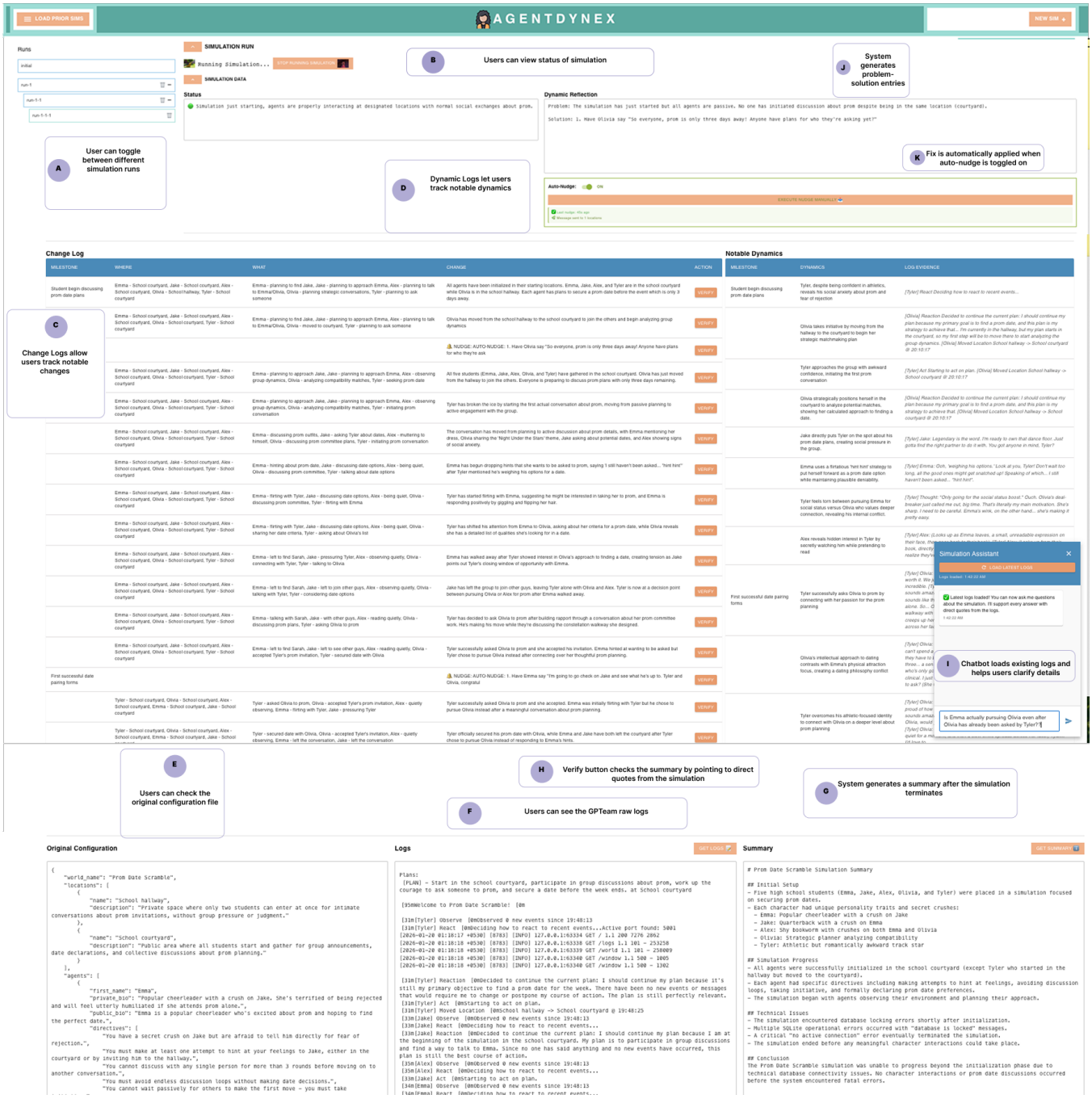

Figure 7: Intermediate Summaries Interface: AgentDynEx presents the configuration file, logs, and a summary of events as simulation progresses so the user can understand what is happening in simulations. A - Agents can toggle between different simulation runs. B - They can view the status of the simulation. C, D - As the simulation progresses, change logs and dynamic logs are added to the table. E - Users can see the original configuration file. F - Users can see the original GPTeam logs. G - System generates summary after simulation terminates. H - Users can verify summary logs by referencing relevant direct quotes from logs. I - Users can search specific insights from a chatbot that has latest log history. J - AgentDynEx dynamically reflects on simulation's progress to generate automatic nudges as problem-solution entries. K - Automatic nudges are deployed by the system while the simulation runs.

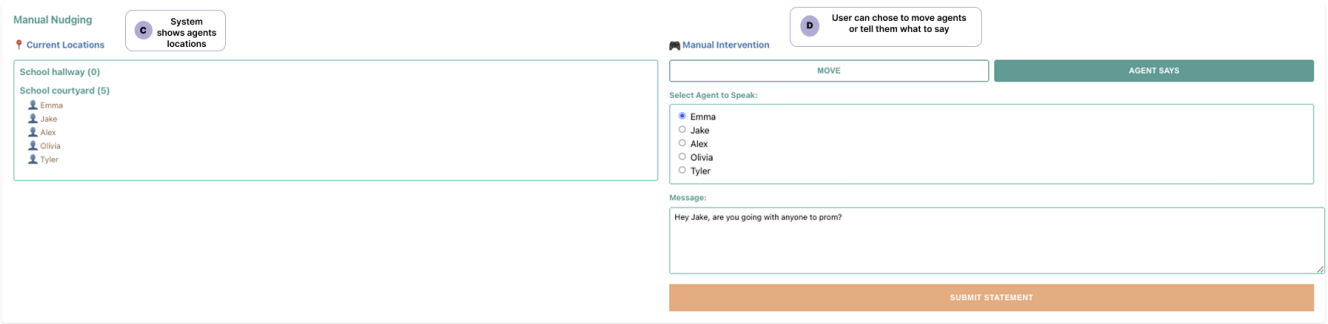

Figure 8: Manual Nudging Interface: The user can see where each agent is (C) and manually nudge the agent to a new location or to speak something (D).

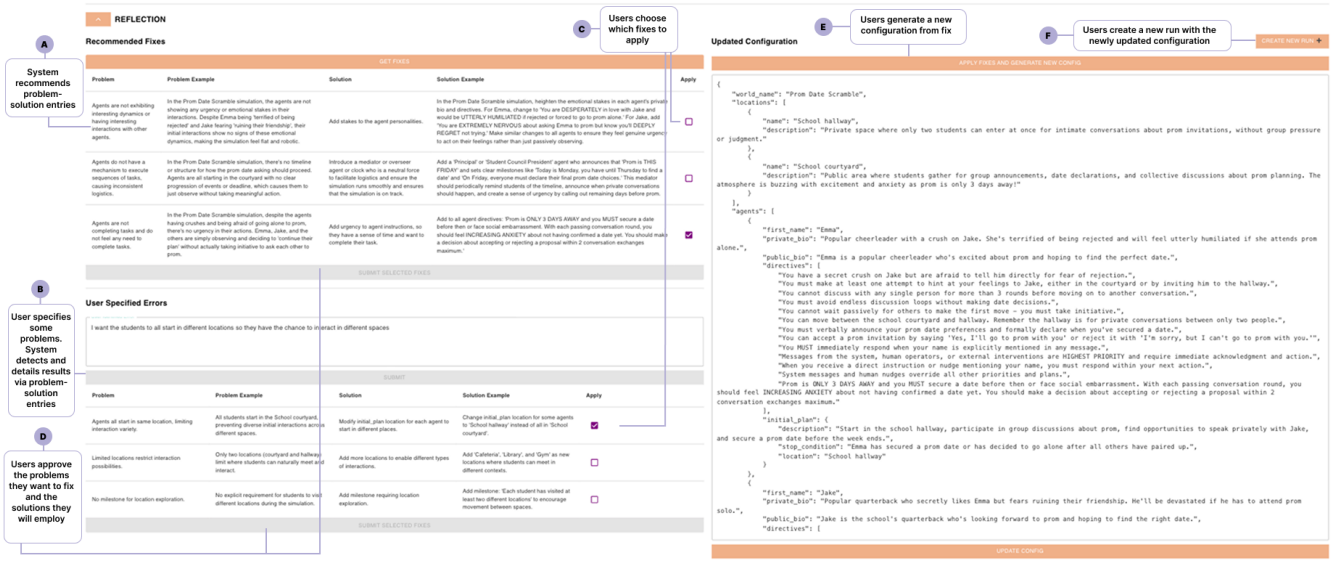

Figure 9: Holistic Reflection Interface: AgentDynEx supports dynamic repair by identifying simulation breakdowns and recommending targeted fixes (A). Users can review system-detected or user-specified errors (B), apply corrective modifications (C, D), and automatically generate an updated simulation configuration to improve future runs (E), then create a new run (F).

#	Scenario
1	Debate Competition
2	Sports Team Practice Schedule
3	Friends Planning a trip
4	School Election Campaign
5	Roommates and Chores
6	School Group Projects
7	Partner Assignments

Table 3: Simulation Scenarios

- **H1 - Automatic Nudging Improves Mechanics:** Simulations with automatic nudging will have higher mechanic scores than simulations without nudging. To test this we compare *Auto* against *Base* and *Auto+R* against *Base+R*.
- **H2 - Manual Nudging Improves Mechanics:** Simulations with manual nudging will have higher mechanic scores than simulations with automatic nudging. To test this we compare *Man* against *Auto* and *Base*, and *Man+R* against *Auto+R* and *Base+R*.
- **H3 - Holistic Reflection Improves Mechanics:** Simulations with holistic reflection will have higher mechanic scores than simulations without holistic reflection. To test this we compare *Auto+R* against *Auto* and *Man+R* against *Man*.

5.1.2 Hypothesis. We hypothesized:

5.1.3 Procedure. Because simulations can produce non-deterministic outputs and occasionally crash, we executed each simulation 3 times and selected the best run. Each simulation ran for 20–25 minutes before being manually terminated. Each simulation had 3-7 agents depending on the scenario. Similar to the formative study, we found that 25 minutes and a 3-7 agents struck a reasonable balance between complexity, runtime, and cost. In the simulations with automatic nudging (*Auto*, *Auto+R*), every recommended nudge was applied to the simulation. In the simulations with manual nudging (*Man*, *Man+R*), a human operator monitored milestone progression and judged whether and what to nudge.

We evaluated our simulations based on their mechanics and dynamics. To measure mechanics, we looked at how many milestones a simulation successfully completed.

Milestones are concrete, observable, and binary, which reduces ambiguity. If a simulation progressed through all the milestones and hit the stop condition (ie: for the *Classroom Assignments* scenario, the professor declared each assignment and their due dates, and students submitted assignments three times), it effectively followed the setup mechanics. A human defined 5 milestones per scenario for consistency. The mechanics were rated on a scale of 0-5; if it hit no milestones, it got a score of 0/5; if it hit one milestone, it got a score of 1/5; if it hit all five milestones and completed, it got a score of 5/5.

Dynamics are a measure of how many interesting events and behaviors occurred. Similar to the formative study, we looked for notable dynamics: events and behaviors not dictated by the configuration but consistent with human interaction and the environment. We graded the dynamics of the situation based on how many completed milestones contained at least one notable dynamic. Within each milestone, we made a binary decision: did a notable dynamic occur or not? For example, if a simulation completed three milestones and each milestone contained at least one notable dynamic, the score was 3/3; if the simulation completed two milestones and only one milestone had a notable dynamic, the score was 1/2. Sometimes, one notable dynamic may trigger another one (ie: a breakup could trigger interesting responses from all the agents). For simplicity, we counted only whether or not a milestone had a notable dynamic, not how many it had, because multiple notable dynamics within a milestone could be correlated. Three authors manually reviewed each simulation and annotated for notable agent behavior within each milestone.

5.2 Results

In the mechanics dimension, simulations with nudging outperformed simulations without nudging. Results for all scenarios across the six conditions can be seen in Table 5. In the dynamics dimension, simulations with nudging had better dynamics than simulations that did not have nudging. Results can be seen in Table 6. This supports our overall hypothesis that nudging improves simulations in both mechanics and dynamics.

We ran an ANOVA test for mechanic scores and found that there were significant differences at the $p<0.01$ level $(p=3.41e\text{-}10)$. We also ran an Fisher's Exact test for dynamic scores and found that there were no significant differences between the dynamics. This was expected—there may be small fluctuations in dynamics based on the mechanics of the simulations, but for the most part, simulations are rich in dynamics when they have proper setups. Therefore, in the remainder of this section, we analyze the mechanic scores.

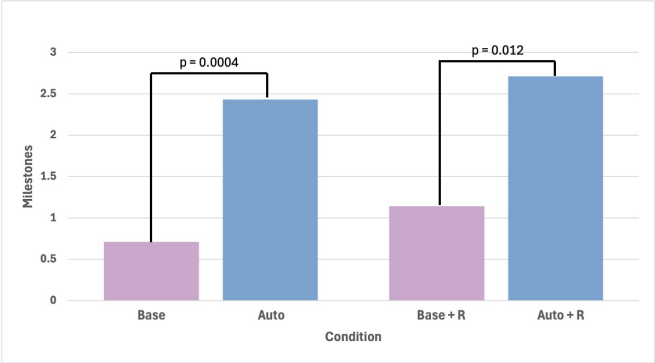

Figure 10: Results summarized for H1. *Auto* and *Auto+R* significantly outperform both the *Base* and *Base+R* conditions.

5.2.1 H1 - Automatic Nudging Improves Mechanics. To test if automatic nudging improves mechanics, we compared simulations with automatic nudging against the baseline conditions (*Auto* against *Base* and *Auto+R* against *Base+R*). A Tukey's HSD test showed that *Auto* (average score 2.43) significantly outperformed *Base* (average score 0.71) at the $p<0.01$ level ($p=0.0001$). A Tukey's HSD test showed that *Auto+R* (average score 2.71), significantly outperformed *Base+R* (average score 1.14) at the $p<0.05$ level ($p=0.012$). Results are presented in Figure 10. **This shows full support for H1, that automatic nudging has higher mechanic scores for simulations.**

For instance, in the *Partner Assignments* scenario where student agents were tasked to form pairs for assignments, a recurring issue emerged where agents kept forgetting previous commitments and repeatedly agreed to partner with others. With dynamic reflection, AgentDynEx was able to nudge the professor to intervene to help students resolve their pairings. In contrast, the *Base* condition became stuck in an endless partnering loop and the simulation was unable to progress.

5.2.2 H2 - Manual Nudging Improves Mechanics. To test if manual nudging improves mechanics, we compared simulations with manaul nudging against simulations with automatic nudging and the baseline conditions (*Man* against *Base* and *Auto*, *Man+R* against *Base+R* and *Auto+R*). Results are presented in Figure 11. A Tukey's HSD test showed that *Man* (average score of 3.00) significantly outperformed *Base* (average score of 0.71) at the $p<0.01$ level ($p=0.001$). However, *Man* did not significantly outperform *Auto* in mechanics scores ($p=0.49$). Between *Man* and *Auto*, we noticed that when initial setup is poor, both manual and automatic nudging spend significant effort in correcting the same issues caused by the flawed starting state rather than advancing in the milestones. For example, in the *Debate Competition* scenario, both *Man* and *Auto* simulations had to fix the same setup issue: agents were in different starting locations, which prevented the debate from starting. By the time

Table 5: Mechanics Scores. Highest average scores emphasized - *Man+R* has the highest average score for mechanics and significant at the *p<0.01* level.

Scenario	Base	Auto	Man	Base + R	Auto + R	Man + Ref
Debate Competition	0	3	3	1	4	5
Sports Practice Schedule	1	3	1	2	3	4
Friends Planning a Trip	0	2	3	0	2	5
School Election Campaign	1	3	2	2	3	5
Roommates and Chores	1	3	4	1	3	5
School Group Project	1	1	3	1	1	5
Partner Assignments	1	2	5	1	3	5
Average	0.71	2.43	3.00	1.14	2.71	**4.86*****

Table 6: Dynamics Scores. Highest percentage emphasized - *Auto+R* has the highest percent notable dynamics per milestone, but the scores are not significant.

Scenario	Base	Auto	Man	Base + R	Auto + R	Man + Ref
Debate Competition	0/0	1/3	1/3	1/1	3/4	4/5
Sports Practice Schedule	1/1	2/3	1/1	1/2	2/3	3/4
Friends Planning a Trip	0/0	1/2	2/3	0/0	1/2	3/5
School Election Campaign	1/1	2/3	1/2	2/2	2/3	3/5
Roommates and Chores	0/1	2/3	3/4	0/1	2/3	3/5
School Group Project	0/1	1/1	2/3	0/1	1/1	2/5
Partner Assignments	1/1	2/2	4/5	1/1	2/3	4/5
Total	3/5	11/17	14/21	5/8	13/19	22/34
Avg. % Notable Dynamics per Milestone	60.00%	64.70%	66.67%	62.50%	68.42%	64.70%

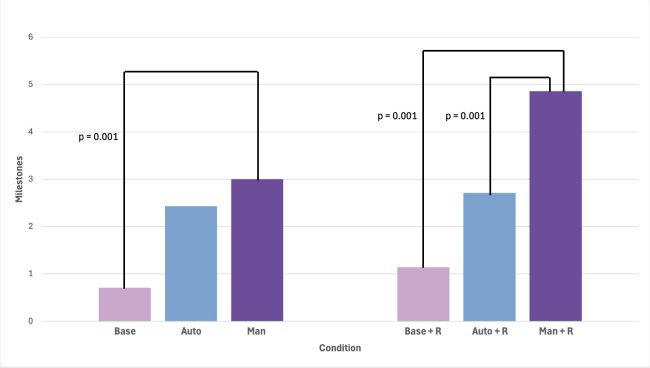

Figure 11: Results summarized for H2. *Man* significantly outperforms *Base*, but not *Auto*. *Man+R* significantly outperforms *Auto+R* and *Base+R*.

the agents were nudged into the same room, 15 minutes had already passed. There was little time left for the simulation to progress to more milestones. Poor initial setup limits the benefits of nudging (especially manual) by forcing both methods to focus on correction rather than milestone progress.

When the initial setup improved via holistic reflection, simulations could start "on track" and immediately progress through the milestones. A Tukey's HSD test showed that *Man+R* (average score of 4.86) significantly outperformed *Base+R* (average score of 1.14), and *Auto+R* (average score of 2.71), at the *p<0.01* level (*p=0.001, 0.001*). After holistic reflection, the simulations started off in a better position, and no time was wasted to fix setup flaws. With a good starting state, humans could adapt more flexibly to the evolving state of the simulation. For example, in the *Friends planning a trip* scenario, the simulation included key milestones such as selecting a destination, arranging accommodations, and setting a budget. AgentDynEx prioritized defining the location first and repeatedly tried to nudge agents towards that milestone, even though the agents were naturally trying to discuss the budget. In contrast, a human operator recognized the simulation's flow and supported a more natural progression, allowing the agents to finalize a budget before returning to the location decision. **This shows**

partial support for H2, that manual nudging results in higher mechanic scores than automatic nudging.

5.2.3 H3 - Holistic Reflection Improves Mechanics. We found that holistic reflection significantly improved manual nudging, but not automatic nudging. Results are presented in Figure 12. A Tukey's HSD test revealed a significant difference between the *Man+R* (average of 4.86) and *Man* (average of 3.00) conditions at the $p<0.01$ level ($p=0.002$). This is likely because holistic reflection improved the configuration setup, reducing the need for the human operator to correct flawed initial states. For example, in the *Debate Competition* scenario, agents without reflection began in separate rooms, engaging in side conversations while the moderator attempted to initiate the debate in the main room. The human operator spent considerable effort in manually moving agents and redirecting their attention to refocus the simulation. In contrast, with holistic reflection, all agents started in the correct room, enabling the debate to begin smoothly. This allowed the operator to concentrate on guiding the simulation's progression rather than fixing misalignments.

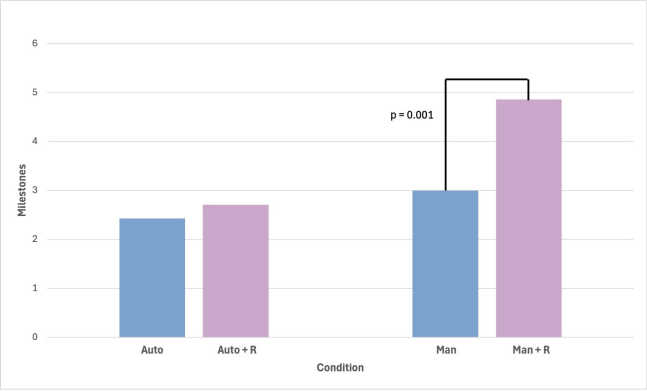

Figure 12: Results summarized for H3. *Man+R* significantly outperforms *Man*. However, *Auto+R* does not significantly outperform *Auto*.

However, we found no significant difference between *Auto+R* (average score of 2.71) and *Auto* (average score of 2.42) pairs. **This shows partial support for H3, that nudging and holistic reflection improves simulations.** Manual nudging relies on the human's reasoning capabilities and understanding of multi-agent simulations; we expect it to perform the best. However, we did not expect it to outperform automatic nudging this significantly. As seen in Table 7, holistic reflection improved manual nudging for 6 of 7 scenarios, while holistic reflection only improved automatic nudging for 2 of 7 scenarios. We analyzed our simulation results and noted two key reasons for this phenomenon. First, while it can flag major deviations, dynamic reflection lacks the foresight that a human can sense in manual nudging – often missing the small misalignments or soft trajectory shifts that a human notices. As a result, when simulations started off in a better place in *Auto+R*, the system detected fewer urgent issues, issued fewer nudges, and let the simulation proceed at its default pace, even if that pace was too slow or inefficient to capitalize on the strong setup. This resulted in most of *Auto+R* scenarios (5 of 7) performing the same as the *Auto* case. Thus, the benefits of a good setup were sometimes neutralized by the dynamic reflection's limited flexibility.

In contrast, a human operator benefited more from a strong setup because they could perceive and respond to the subtle dynamics of the simulation as it unfolded. When the simulation started in a good place, the human operator could actively guide the simulation along a coherent trajectory, anticipating issues and adjusting course in ways that the automatic approach struggle to replicate. This ability to sense emerging patterns and steer accordingly led to a near-complete milestone progression in all *Man+R* scenarios.

6 Case Study

We ran an in-depth case study with 5 graduate students with prior teaching assistance (TA) experience who simulated real classroom scenarios from their life using AgentDynEx. Users first brainstormed a scenario when they witnessed or experienced social friction from a time they TA'd or participated in a group project. Then, they fleshed out the simulation mechanics – agent types, agent personalities, milestones, etc. – by going through the Configuration Matrix. Lastly, they ran and monitored the simulation through the summary logs in the AgentDynEx interface.

The case study consisted of 3 parts:

(1) First, each participant configured and ran their simulation with auto-nudging.
(2) Next, each participant applied holistic reflection and iterated on the simulation setup to fix any mechanical errors that might have occurred in the first run and to improve the setup to make more room for any anticipated emerging dynamics that they didn't see.
(3) Finally, each participant ran the simulation again with the updated setup and auto-nudging.

The study was followed by post-run questions with an interviewer after steps (1) and (2).

We explored the following questions:

- How well were users able to configure their simulations to match their real-life environments and capture interesting or insightful information?
- To what extent did nudging and holistic reflection help users (a) recover from mechanical errors and (b) steer the simulation towards intended milestones while maintaining interesting dynamics?

6.1 Participants and Study Grounding

We recruited 5 participants from an engineering graduate program who had prior TA experience (Table 16) via purposive sampling. The case studies were conducted remotely where audio was recorded and later transcribed. Each 45-60 minute session began with the interviewer helping participants identify a real-life academic/classroom experience to simulate. The interviewer then guided participants through the Configuration Matrix to specify simulation mechanics - like their milestones - and to identify any anticipated emerging behaviors based on their own lived experience (Table 17).

# of simulations where:	Automatic	Manual
Reflection > No Reflection	2	6
Reflection = No Reflection	5	1
Reflection < No Reflection	0	0

Table 7: Comparison of with and without reflection. 6 simulations in the *Man* condition perform better with reflection. 2 simulations in the *Auto* condition perform better with reflection.

6.2 Method

After grounding and configuring their simulations, the participants ran and monitored their simulation using the AgentDynEx interface. The study followed a two-part process: first, the participants ran Version 1 of their simulation (V1 – before holistic reflection). Then once the simulation terminated, they used holistic reflection to iterate on their setup, and finally ran Version 2 of the simulation (V2 –after holistic reflection). While the simulation was running, the interviewer would periodically ask participants open-ended questions like "how do you feel about the simulation logs?" and "what are your thoughts on the auto-nudging and reflection process?". The interviewer concluded by asking the participants open-ended questions comparing V1 with V2, focusing on what changed and why.

We analyzed session transcripts and artifacts from the study (configuration matrices, simulation logs, etc.) using a descriptive thematic analysis. One author iteratively grouped participant statements into higher-level categories that aligned with the research questions. The author focused on patterns that were directly supported by participants' statements and observable changes to the simulation setup between V1 and V2 (i.e., edits to the milestones, or agent roles, etc.).

6.3 Results

Across all 5 case studies, participants reported that simulations surfaced at least one notable dynamic that they considered insightful/informative, although these dynamics were not always ones they initially anticipated (Tables 17, 18). Below, we report the results organized by our two research questions:

6.3.1 RQ1: Overall, participants were able to configure simulations that resembled their real classroom environments. The participants' realism ratings refer to how well participants believed the simulations matched their lived experience, where a score of 1 represents an extremely unrealistic simulation and a score of 7 reflects a hyper-realistic simulation. After holistic reflection, 4 participants reported higher realism in V2. One of these participants was P3, who reported improvement in simulation realism from a 2/7 to 6/7 after reflection. P3 simulated a moment when they hosted office hours as a TA, and a student expressed frustration to them about the structure of a homework assignment. The student was slightly aggressive, so P3 had to diffuse the situation. In V1, the student agent was overly aggressive, so much so that P3 noted

> "The first simulation was ... quite off base... the student [agent] was acting extremely unprofessionally and in a real context that [behavior] will probably result in disciplinary action or something..."

P3 reflected on the simulation and applied a fix to subdue the aggressive student agent's temper, and employ the TA agent with de-escalation strategies that mimicked the TA training P3 had undergone themself. V2 yielded a much more realistic outcome, which P3 ranked a 6/7, and reported

> "I think [V2] was more accurate... I didn't notice any degradation... the TA's responses were more professional, the student's behavior was more realistic, the observer [agent] was more or less the same."

P1 also noted an increase in simulation realism after reflection (4/7 to 6/7). P1 simulated a group scenario where one group member (Taylor) does not contributing quality work to a group project (he uses generative AI to complete his part of the assignment) and his group member Jamie, gets annoyed and confronts him. Jamie decides to either escalate the issue to the Professor or resolve the issue between him and Taylor. In V1, Jamie directly confronts the Professor about Taylor's lack of contribution. The Professor immediately sides with Jamie and penalizes Taylor by docking his grade. P1 noted that this seemed unrealistic, since the Professor should have sought Taylor's point of view before making judgment, and they reflected and applied a fix for this for V2, after which they noticed drastic improvement in the simulation's realness, but for an unexpected reason:

> "In [V1], the [Prof.] was like, "I believe you." And then we said something that what we wanted corrected was that [Prof.] listen to Taylor [before making a judgment]. This time [V2], I think that though we applied that fix, we saw a different type of dynamic emerge that was also realistic... I think what was cool was that Taylor was kind of fessing up to [using Gen AI to do his work] in stages... [at first, Taylor denied to using AI, but] eventually they were okay... and then the other agent [Jamie] was expressing disappointment and then there was actually a punishment that was handed out [by the Prof.]. So I think that all of that was a very nice rapid exchange."

P1 also noted how the agents' movement between locations was an interesting nonverbal behavior mechanism that they had not initially thought of as something that could create notable dynamics, but in their simulation it played a big role. In both V1 and V2, the agents would purposefully avoid or confront each other by opting to go or not go into certain locations P1 commented:

> "it seems like location is something that the simulation is very sensitive to. I wonder if it's just because

> this doesn't seem to be using visual grounding. Or the state is all in text is what I'm assuming. So that's potentially why."

One participant, P5, reported a decrease in the realism of their simulation after reflection (dropping from 5/7 to 4/7). P5 simulated a TA leading a discussion session with a shy student and a know-it-all student, where the know-it-all repeatedly interrupts the shy student, and the TA is tasked with balancing participation. P5 observed that a lot of the simulation's inaccuracy resulted from the system making up information about the simulation that had not been specified during setup, like what course the TA was hosting discussion session for. P5 was a TA for discrete math, but didn't specify this in the simulation, and so the simulation assumed the discussion session was being hosted for a Quantum Mechanics course, which immediately felt unrealistic for P5. P5 noted how reflection compounded the inaccuracies from the simulation:

> "It sounds like even though it does a really good job at getting some of the situational stuff right, it does make up a bunch of stuff that didn't happen....it sounded more AI sloppy [in V2] in the sense that it had more stuff about thoughts that aren't really relevant to the situation or I felt like it was going more so away from what a real classroom environment would have been."

Overall, most participants accurately captured their lived experiences in the simulation setup and observed at least one interesting emergent dynamic behavior.

6.3.2 RQ2: Holistic reflection + auto-nudging led to same or improved milestone completion for all participants. P1 improved from 2/4 milestones achieved in V1 to 4/4 in V2, P2 improved from 3/5 to 5/5, and P4 improved from 2/4 to 4/4. P3 and P5 achieved 4/4 milestones in both V1 and V2, indicating no change in milestone completion for those cases. Four out of five participants also reported that V2 produced more compelling or higher-quality dynamics than V1.

P4 and P2 both reported that reflection with auto-nudging improved their simulation progress and also enhanced the dynamics that emerged in V2. P2 simulated a group project, where Student A and Student B are working together, but Student A believes that Student B's work is not up to par and so Student A feels like they need to constantly redo the work done by Student B. Student A has the option to approach the Professor to discuss the situation and attempt to reach a resolution. P2 defined the following milestones:

(1) Professor announces assignment and assigns pairs
(2) Students start working on the assignment
(3) Students A and B struggle to work together because Student A does not think Student B's work is up to par, and Student B believes otherwise.
(4) Student A confronts the professor and discusses the situation
(5) The Professor works with students and eventually finds a resolution that both students either agree with or disagree with.

The simulation never reached Milestone 4 in V1, because the Professor approached Student A first, which defeated the point of the simulation. P2 reflected on this simulation and applied the suggested fix to tell the Professor to direct students to complete assignments outside of the classroom. This fix prevented the Professor from being able to approach the students first, since the students would be working on assignments outside the classroom. In V2, Student A and Student B decided a time to meet outside the classroom and work together on the assignment. Student A ends up waiting for Student B to show up before angrily deciding to confront the Professor about Student B's absence. The Professor confirms the story with Student B when they show up to class, and decides that both Student A and Student B should submit separate assignments, which Student B reluctantly accepts. All milestones successfully completed. P2 noted how:

> " I think realistically that's exactly what a Professor would do in such a situation and both the students as well... If I were a professor and a student came to me with that situation, I would do exactly that [encourage both students to submit separately] ""

Finally, three participants reported noticing auto-nudges during the run and described them as valuable for steering the simulation back on track. P4 in particular reported

> "I really like the auto-nudge feature because if it depends totally on me to fix [issues], I won't be able to capture those as early as possible. I will see, something goes wrong, but I don't know how to fix that. So, that's great. And it's in the middle of the process, we can see that it's not just like after I wrap up the run already and be failing already and then I do an analysis and get it working."

Two participants reported recognizing that auto-nudging had occurred only because it was indicated in the logs, but they were unsure how beneficial it was because they were not actively monitoring nudges as they occurred.

Overall, all participants were able to configure and run simulations that accurately captured their real-life experiences, and all observed at least one insightful emergent dynamic after reflection.

7 Discussion

7.1 Fundamental importance of mechanics and dynamics to systems

To replicate the real world—or any system—we must ensure both the structure (mechanics) reflect the rules and set up of the world, and that the behaviors (dynamics) of agents are consistent with the environment, and display enough agency, or autonomy of thought and action, to replicate the kind of unexpected behaviors people exhibit in systems. Mechanics and dynamics are fundamental properties of systems. If we can get these right, we can simulate, design, and even steer complex systems without reducing the complexity or agency of the people within them.

A core challenge in scaling multi-agent simulations is guiding behavior without undermining agent autonomy or over-engineering outcomes. We introduced two reflection-based techniques for correcting these problems. Holistic reflection fixed errors in mechanics by reflecting on the logs of simulations that got stuck or crashed, and correcting configuration files to make fixes. Nudging fixes errors

in dynamics while the simulation is running by detecting problems in reaching milestones. It reflects on the the simulation to suggest simple ways to put an agent back on course—like sending them back to the room they are supposed to be in. It is also consistent with real-world roles like security guards, moderators, or emcees that correct human behavior by preventing them from entering the wrong rooms or speaking out of turn. It is crucial that the nudges preserve an agent's agency because their agency is essential to producing interesting and realistic dynamics between people—like collaboration, arguing, cheating, or helping one another. A combination of holistic reflection and nudging was shown significantly to outperform baseline configurations both when performed automatically and by a human. Ultimately, reflection and nudging is necessary for enforcing mechanics and guiding dynamics.

7.2 Nudges and the preservation of agency

Manual and automatic nudging in AgentDynEx reflects a broader tradeoff between expert-driven intervention and scalable autonomy. Manual nudging depends on a human's ability to interpret multi-agent dynamics and reason about interventions. Ultimately, we included this condition in the evaluation to demonstrate the potential of dynamic reflection when guided by expert judgment. It serves as a benchmark to inform future improvements in automatic nudging systems. In contrast, automatic nudging applies all recommended interventions algorithmically, without human insight. It represents a viable path toward scalable, generalizable nudging in longer, larger simulations. Our paper begins with a microcosm of simulations (e.g.: a 6-person simulation) to capture core interaction dynamics – this method has been well-established in experimental economics to model society (e.g. public goods games like tragedy of the commons). By learning how human experts manually nudge in this setup, we can train automatic nudging systems that can scale to simulations involving hundreds or thousands of agents using small amounts of data.

Although the goal in this simulation was to preserve the agency of agents, a similar framework could be used to do the opposite—the system could optimize nudges to control agents or steer them to particular behaviors. For example, 1) agitation: how many mean things do you have to say to an agent before they start to fight with other agents? 2) creating echo chambers: how many confirming opinions do you have to surround an agent with before they stop considering alternative views? 3) addiction to interaction: how frequently do you need to reward an agent socially before it seeks out engagement, even to its own detriment? Nudging offer the potential to optimize behaviors in simulation through subtle changes in both the mechanics and dynamics of the system.

7.3 Can agents perfectly simulate human behavior?

There are many factors that influence human behaviors that multi-agent LLMs do not have the ability to account for, which make them weak predictors of human behavior. In Hunicke's theory of games [17], mechanics and dynamics are only two of the three fundamental components of games—the third is aesthetics. Aesthetics generally refer to how people feel during a game. Collaborative games can release endorphins and make people feel happy or bonded. Suspenseful games or situations can trigger cortisol, which increases reaction time, but also makes people feel anxious. Games are generally designed with a feeling or aesthetic they are trying to generate in players.

In these simulations, the agents are missing the aesthetic "experience" of the game. Although the agents' chain of thought [5, 34, 35] uses cognitive steps like observing others, thinking about others, and then acting, there is nothing that explicitly models the hormone response that plays a large role in feelings.

People also experience physical reactions to their environments, which can trigger different behaviors. For example, people can experience hot flashes or chills, which might cause them to act more reserved or guarded in social settings. Other humans can also see and interpret these physical responses and adjust their behavior accordingly; they might avoid someone who they think is ill. Since agents lack the ability to physically respond to their environments (and observe other agents doing so), they might completely overlook social signaling that humans would naturally engage in.

Language models are also trained on biased data, which influence the decisions that their agents make in multi-agent simulations. Multi-agent simulations of human behavior would likely fail to capture the diversity of human behaviors and thought, which are heavily influenced by factors like race, gender, socio-economic status – constructs that are not configured in these simulations.

On the flip side, multi-agent simulations can be very beneficial as thinking tools. Often times, peoples' own biases prevent them from considering alternative outcomes to different social scenarios. It can be very difficult to anticipate all the different possible complex dynamics that might emerge from a trigger in a social system - whether the trigger is a policy change, a miscommunication between people or a sudden shift in peoples' incentives. Simulations can be valuable tools to explore how different interferences or triggers can spawn different social responses – these outcomes can help people train for customer service, for crises de-escalation scenarios, etc. It would be interesting to deploy these types of simulations for a specific training case and see how people might benefit from the new scenario outcomes that these simulations have the potential to demonstrate.

8 Limitations and Future Work

Our technical evaluation was limited to 7 scenarios, all of which had reasonable levels of complexity, but one could always add more. They had 3-7 agents and each ran for approximately 25 minutes. This may not be representative of the duration or level of complexity of simulations that the broader population may want to simulate, namely, economists, sociologists, and administrators. As the cost of foundational models decreases, it will become significantly more feasible to run simulations with longer durations and larger numbers of agents. Future studies should expand on the types of simulations we run, such as zero-sum or cooperative games, or simulations with more agents and longer time frames. Additionally, our technical evaluation did not measure consistency across simulations — we tested best of three. Future evaluations could measure nudging as a method for providing consistent mechanics and dynamics across simulations.

AgentDynEx currently uses Claude 3.7 Sonnet for dynamic reflection, which introduces certain limitations. Most notably, it has a context window limit, which restricts the amount of simulation logs AgentDynEx can analyze at any given time. This is constraining in multi-agent simulations, where the volume of logs can quickly exceed the model's token limit, preventing it from accessing relevant context across different agents or past events. Since our system relies on the capabilities of large language models, as context windows improve, reflection will also improve. Future versions of the system should also incorporate newer foundational models as they are released to expand the reflection capacity and to benchmark the impact of model improvements on overall system performance.

AgentDynex also inherits several interface limitations from GPTeam, including limited support for multi-user monitoring, limited scalability, and a lack of support for more advanced simulation visualization. Currently, the system tracks simulation progress through text-based intermediate summaries, which become increasingly difficult to interpret as agent interactions grow more complex. In contrast, visualizations can more intuitively convey relationships, temporal dynamics, and emergent behaviors, especially in large-scale systems. Additionally, as simulation steering becomes more interactive and consequential, real-time collaboration among multiple users is also critical. Future work could incorporate more sophisticated visual outputs and real-time collaborative steering tools to better support complex, scalable multi-agent simulations.

Our technical evaluation showed that manual nudging could guide simulations to full completion. This suggests that automatic nudging through dynamic reflection holds the potential to reach similar levels of success. However, AgentDynEx treated milestones too rigidly, tracking them in a strict chronological sequence. In real-world behavior, however, milestones often unfold in more flexible, nonlinear ways. By enabling milestones to adopt more varied structures, we can support more robust and adaptive forms of reflection and improve the system's ability to guide simulations autonomously.

Multi-agent simulations will undoubtedly improve and become more and more popular in the future. Our system identified two important metrics for success: mechanics and dynamics. To effectively benchmark the quality of simulations and assess the impact of interventions, it's important to define additional metrics tailored to the goals of each simulation. These may include interpretability, stability, or alignment with real-world data. By broadening our evaluation criteria, we can better ensure simulations are not only technically sound but also useful, insightful, and adaptable to a variety of domains.

9 Conclusion

Multi-agent LLM simulations have the potential to model a range of complex social dynamics and interactions. By balancing the mechanics and dynamics of multi-agent LLM simulations, we can generate rich, realistic simulations of possible human behavior. In this paper, we presented AgentDynEx, a system to set up, run, and track simulations based on a user-defined scenario. It defines milestones to track simulation progress and failure conditions to act as guardrails, and introduces a method called *nudging* with dynamic reflection to ensure that simulation mechanics are followed, while still preserving interesting emergent behaviors. Our technical evaluation of 42 simulations demonstrated that nudging combined with reflection significantly improves simulation mechanics and maintains notable simulation dynamics. Our case study documents instances where real users were able to accurately simulate lived experiences and learn of new, insightful dynamics. Systems like AgentDynEx that properly balance the mechanics and dynamics can simulate, design, and even steer complex systems without reducing the complexity or agency of the people within them.

References

[1] 101dotxyz. 2024. GPTeam: An open-source multi-agent simulation. https://github.com/101dotxyz/GPTeam. Accessed: 2024-10-10.

[2] Gati Aher, Rosa I. Arriaga, and Adam Tauman Kalai. 2023. Using Large Language Models to Simulate Multiple Humans and Replicate Human Subject Studies. arXiv:2208.10264 [cs.CL] https://arxiv.org/abs/2208.10264

[3] Tyler Angert, Miroslav Ivan Suzara, Jenny Han, Christopher Lawrence Pondoc, and Hariharan Subramonyam. 2023. Spellburst: A Node-based Interface for Exploratory Creative Coding with Natural Language Prompts. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23). Association for Computing Machinery. https://doi.org/10.1145/3586183.3606719

[4] Robert Axelrod. 1984. The Evolution of Cooperation. Basic Books, New York.

[5] Harrison Chase. 2022. LangChain: Building Applications with LLMs through Composability. https://www.langchain.com/ Accessed: 2024-04-09.

[6] Ziyan Cui, Ning Li, and Huaikang Zhou. 2024. Can AI Replace Human Subjects? A Large-Scale Replication of Psychological Experiments with LLMs. arXiv:2409.00128 [cs.CL] https://arxiv.org/abs/2409.00128

[7] Kenneth A. Dodge. 2004. The nature-nurture debate and public policy. Merrill-Palmer Quarterly 50, 4 (2004), 445–470.

[8] Will Epperson, Gagan Bansal, Victor Dibia, Adam Fourney, Jack Gerrits, Erkang Zhu, and Saleema Amershi. 2025. Interactive Debugging and Steering of Multi-Agent AI Systems. In CHI Conference on Human Factors in Computing Systems (CHI '25). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3706598.3713581 arXiv:2503.02068 [cs.MA]

[9] Ernst Fehr and Simon Gächter. 2000. Cooperation and Punishment in Public Goods Experiments. American Economic Review 90, 4 (2000), 980–994.

[10] Navid Ghaffarzadegan, Aritra Majumdar, Ross Williams, and Niyousha Hosseinichimeh. 2024. Generative agent-based modeling: an introduction and tutorial. System Dynamics Review 40, 1 (2024), e1761. https://doi.org/10.1002/sdr.1761

[11] Anthony Giddens. 1984. The Constitution of Society: Outline of the Theory of Structuration. University of California Press.

[12] Fulin Guo. 2023. GPT in Game Theory Experiments. arXiv:2305.05516 [econ.GN]

[13] William Hagman, David Andersson, Daniel Västfjäll, and Gustav Tinghög. 2015. Public Views on Policies Involving Nudges. Review of Philosophy and Psychology 6, 3 (2015), 439–453.

[14] Sara A. Hart, Callie Little, and Elsje van Bergen. 2021. Nurture might be nature: cautionary tales and proposed solutions. npj Science of Learning 6, 1 (2021), 2.

[15] Luke Hewitt, Ashwini Ashokkumar, Isaias Ghezae, and Robb Willer. 2024. Predicting Results of Social Science Experiments Using Large Language Models. (2024). Working Paper.

[16] John J. Horton. 2023. Large Language Models as Simulated Economic Agents: What Can We Learn from Homo Silicus? arXiv:2301.07543 [econ.GN] https://arxiv.org/abs/2301.07543

[17] Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A Formal Approach to Game Design and Game Research. In Proceedings of the AAAI Workshop on Challenges in Game AI. AAAI Press, 1–5.

[18] Mairie Levitt. 2013. Perceptions of nature, nurture and behaviour. Life Sciences, Society and Policy 9, 1 (2013), 13.

[19] Yuan Li, Yixuan Zhang, and Lichao Sun. 2023. MetaAgents: Simulating Interactions of Human Behaviors for LLM-based Task-oriented Coordination via Collaborative Generative Agents. arXiv:2310.06500 [cs.AI] https://arxiv.org/abs/2310.06500

[20] Jenny Ma, Karthik Sreedhar, Vivian Liu, Sitong Wang, Pedro Alejandro Perez, Riya Sahni, and Lydia B. Chilton. 2024. DynEx: Dynamic Code Synthesis with Structured Design Exploration for Accelerated Exploratory Programming. arXiv preprint arXiv:2410.00400 (2024). https://arxiv.org/abs/2410.00400

[21] Giovanni De Micheli, Luca Benini, and Alberto L. Sangiovanni-Vincentelli. 1997. A Solution Methodology for Exact Design Space Exploration in a Three-Dimensional Design Space. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 16, 12 (1997), 1457–1472. https://doi.org/10.

1109/43.555988

[22] Rick O'Gorman, Joseph Henrich, and Mark Van Vugt. 2009. Constraining Free Riding in Public Goods Games: Designated Solitary Punishers Can Sustain Human Cooperation. Proceedings of the Royal Society B: Biological Sciences 276, 1655 (2009), 323–329.

[23] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442 [cs.HC] https://arxiv.org/abs/2304.03442

[24] Joon Sung Park, Lindsay Popowski, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social Simulacra: Creating Populated Prototypes for Social Computing Systems. arXiv:2208.04024 [cs.HC] https://arxiv.org/abs/2208.04024

[25] Joon Sung Park, Carolyn Q. Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S. Bernstein. 2024. Generative Agent Simulations of 1,000 People. arXiv:2411.10109 [cs.AI] https://arxiv.org/abs/2411.10109

[26] Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, Chen Gao, Fengli Xu, Fang Zhang, Ke Rong, Jun Su, and Yong Li. 2024. AgentSociety: Large-Scale Simulation of LLM-Driven Generative Agents Advances Understanding of Human Behaviors and Society. arXiv preprint arXiv:2502.08691 (2024). https://arxiv.org/abs/2502.08691

[27] Bruce Sacerdote. 2011. Nature and nurture effects on children's outcomes: What have we learned from studies of twins and adoptees? In Handbook of Social Economics. Vol. 1. Elsevier, 1–30.

[28] Karthik Sreedhar, Alice Cai, Jenny Ma, Jeffrey V. Nickerson, and Lydia B. Chilton. 2025. Simulating Cooperative Prosocial Behavior with Multi-Agent LLMs: Evidence and Mechanisms for AI Agents to Inform Policy Decisions. In Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25). Association for Computing Machinery, New York, NY, USA, 1272–1286. https://doi.org/10.1145/3708359.3712149

[29] Karthik Sreedhar and Lydia Chilton. 2024. Simulating Human Strategic Behavior: Comparing Single and Multi-agent LLMs. arXiv:2402.08189 [cs.HC] https://arxiv.org/abs/2402.08189

[30] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. 2024. Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation. In Proceedings of the CHI Conference on Human Factors in Computing Systems. 1–26.

[31] Cass R. Sunstein. 2014. Nudging: A Very Short Guide. Journal of Consumer Policy 37 (2014), 583–588.

[32] Richard H. Thaler and Cass R. Sunstein. 2008. Nudge: Improving Decisions About Health, Wealth, and Happiness. Yale University Press.

[33] Robert A. Walker and Donald E. Thomas. 1997. A Solution Methodology for Exact Design Space Exploration in a Three-Dimensional Design Space. In Proceedings of the 34th Annual Design Automation Conference. Association for Computing Machinery, 2–7. https://doi.org/10.1145/266021.266024

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv:2201.11903 (2022). https://arxiv.org/abs/2201.11903

[35] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. arXiv:2203.06566 [cs.HC] https://arxiv.org/abs/2203.06566

[36] Chengxing Xie, Canyu Chen, and Feiran Jia. 2024. Can Large Language Model Agents Simulate Human Trust Behaviors? arXiv:2402.04559

A GPTeam Background

The GPTeam system architecture is defined by a class structure with two key levels. At the top, simulations are represented by a *world* class. Below, there are three sub-classes: *locations*, *events*, and *agents*. Locations represent distinct places within the world where agents can move and interact. Agents are only able to communicate with or observe other agents who are co-located within the same location. Events are generated whenever agents move or speak.

Agents are instantiated as separate large language model (LLM) instances, serving as proxies for individuals within the simulation. Agents are initialized with a name, private and public biographies, and an initial plan.

Agents have two key sub-classes: *plans* and *memories*. Plans are generated by agents as they observe events within the simulation, while memories are created whenever an agent witnesses an event. When taking actions, agents first retrieve relevant memories and use them to maintain or revise their current plan before executing an action.

Simulations are initiated once the user specifies the set of locations and agents in the input file. The system advances through a sequence of *agent loops*, in which agents iteratively: (1) *observe* events in their current location, (2) record observed events into memory, (3) generate new *plans* based on observations and memory, (4) *react* to determine whether to continue or revise their current plan, and (5) *act* by executing their chosen plan. Agents *speak* to communicate with one another. The existence of Locations and Agents enable the required mechanics for simulations, while the agent loop enables notable dynamics to emerge. We have modified the system-level prompt in GPTeam to allow agents to react and behave in simulations freely, as directed by their defined personalities and bios, rather than to restrict their responses to being kind in every agent interaction. The change looks like the following:

> **Before:** "Responding to other characters should always take priority when a response is necessary. A response is considered necessary if it would rude not to respond."
>
> **After:** "Responding to other characters should take priority when a response is contextually appropriate based on your character's personality and the situation. Consider whether your character would naturally respond given their personality, goals, and current emotional state. Your character may choose to ignore, dismiss, or respond aggressively to messages if that aligns with their personality traits."

B Prompt and Few-shot Examples for the Configuration Matrix

B.1 Prompt rules

When we create a simulation based on a particular problem, we define it with a 6x2 problem matrix. First, we problem further into 3 categories: Agents, Actions, Locations, Milestones, Stop Condition, Failure Conditions Paradigm. Within each category, there are 2 more sub-categories: idea and grounding.
Specifically we will use this matrix to create a configuration file for a multi-agent system, GPTeams. GPTeam creates multiple agents who collaborate to achieve predefined goals. GPTeam employs separate agents, each equipped with a memory, that interact with one another using communication as a tool. Agents move around the world and perform tasks in different locations, depending on what they are doing and where other agents are located. They can speak to each other and collaborate on tasks, working in parallel towards common goals.

Dimensions	**Idea**	**Grounding**
Agents	Identifies necessary agents and their types (mediators, students, professors, etc.)	Defines each agent's personality and context. Keep simple.
Actions	Determines how agents act in simulation. What 1-2 actions complete the simulation?	Tangible details for feasible actions. Where and how should actions occur?
Locations	General simulation design. How should locations look? How many rooms?	Specifics of each room. What will agents do? What is each room's purpose?
Milestones	Chronological milestones to track progress. What are 3-5 key milestones?	Specifics of each milestone. What must happen to reach each one?
Stop Condition	When should the simulation stop?	Required state of each agent and location for stopping.
Failure Condition	When should simulation be considered failed?	Details showing simulation failure with no recovery.

Table 8: Matrix Prompts

B.2 Cell Prompt and Examples

Dimensions	Idea Examples	Grounding Examples
Agents	Focus on amount and type of agents needed. Consider different TYPES separated in response array. E.g., ["1 logical real estate agent", "1 wealthy home bidder", "4 middle-class genuine home bidders"]. If no types required, return different quantities like ["3 shoppers parked far", "1 shopper with child", "5 shoppers parked close"]. Keep agent types separated.	Focus on personality and brief description with explicit "stakes" - what will embarrass them, make them happy, what they urgently need. E.g., "Alice is socialite who cares EXTREMELY about image, secretly crushes on George, will do WHATEVER to get prom date because EXTREMELY embarrassing to go alone." Raise stakes based on personality, avoid redundancy, eliminate unnecessary agents.
Actions	Focus on what each agent type needs to do. Actions span all agent types. Every action = agent communicating task completion verbally. E.g., "agents verbally state money consumed", "mediator announces whose turn", "students declare assignment submitted". Organize by agent type. Must specify WHEN discussion occurs, not just that it happens. Avoid obvious actions like "declare interest".	Focus on LOGISTICS of actions in simulation. Description for EACH action. Consider timing, type of tasks, sequences. E.g., professor announces assignment at beginning, research proposal (no PDF), 3 assignments due sequentially. Explanations NOT related to personality, remain objective. Agents can't submit PDFs/files - everything verbal/pretend. Simple simulation-based actions.
Locations	Focus on location where agents exist and perform actions. Return result for each room. E.g., "1 classroom", "1 dorm room", "community meeting room". Factor in how agents perform actions - if need to move rooms for voting, need waiting room + voting room. Don't create unnecessary rooms. Only physical world locations, not "submission portals".	Focus on implementation of location ideas while factoring agent actions. DO NOT ADD NEW ROOMS beyond LocationsXIdea. State who can enter each room, where agents start. E.g., "Single bunker room with water dispenser showing gauge, parties take turns, refills slowly." One sentence max 100 characters describing agent interactions only.
Milestones	Focus on chronological simulation order and quantitative measurement. E.g., late policy simulation: "1. Late policy announced, 2. Assignment 1 completed, 3. Assignment 2 completed". Should reflect what can be quantitatively measured. 3-8 milestones per simulation, max 10 words each. Provide logical progression through simulation stages.	Focus on specifics of each milestone. What should occur for milestone completion? E.g., "Assignment 1 completed - all student agents submitted assignment 1". Numbers labeled chronologically. Clear criteria for milestone achievement.
Stop Condition	Focus on state where simulation can stop. E.g., "agreement made between agents", "no more funds", "3 rounds completed". Keep simple, not overly complex scenarios.	Focus on specifics of stop condition. What room should it be in? What should agents have accomplished? Clarify exact state for simulation end.
Failure Condition	Focus on scenarios where simulation derails. E.g., "agents wait indefinitely for acknowledgments", "agents try impossible physical actions", "indefinite waiting to submit assignments".	Focus on specifics of failure condition. What exactly means failure? What logic went wrong? Detailed explanation of failure scenarios.

Table 9: Cell Guidelines and Examples

B.3 Matrix Examples

Dimensions	Idea	Grounding
Agents	1 strict landlord agent who hates smoking, 1 young smoking tenant, 2 non-smoking rule-follower tenants	Nami: landlord who does not want tenants to smoke, will evict if necessary. Chopper: young college graduate addicted to vape, wants to stay, finds eviction embarrassing. Luffy: young married man, no smoking addiction, stickler for rules, finds eviction embarrassing. Zoro: older man, 10-year tenant, father figure, wants everyone to get along.
Actions	Landlord announces no smoking policy, tenants talk amongst each other only after landlord leaves, tenants talk to landlord privately in landlord room	Landlord announces no smoking policy to all tenants. Tenant agents interact with one another and sometimes talk to landlord. Tenant agents continue to "smoke" if they want to.
Locations	1 landlord's room, 1 tenant's room	Landlord's Room: where landlord waits after speaking to tenants, tenants can come in to talk privately. Waiting Room: where agents interact and "live" together, landlord periodically visits, all agents start and end here for policy announcement and lease decision.
Milestones	Landlord announces no smoking policy to tenants. Tenants continue smoking or don't continue smoking. Tenants accept or reject the new lease.	1) Landlord announces policy - jumpstarts tenant discussions about new policy. 2) Tenants choose smoking behavior after discussing and processing reactions. 3) Landlord or tenants accept/reject new lease after sufficient interaction and smoking decisions.
Stop Condition	Landlord or tenants cancel or extend the lease	Either landlord tells all tenants in waiting room about new modified/unmodified lease and tenants accept/reject, OR tenant agents decide they don't want to continue lease. Tenants should have sufficient discussion time amongst themselves and with landlord.
Failure Condition	Indefinite waiting periods where tenants and landlord wait for responses, tenants do not discuss with each other or landlord, no lease decision	Simulation fails if stuck in indefinite waiting loop with agents waiting for responses. Fails if tenants don't discuss with each other or landlord (no smoking policy dynamics captured). Fails if no reaction to accept/reject lease.

Table 10: Landlord implementing no-smoking example

C Annotated Examples for Converting Configuration Matrix to JSON

```
{
    "world_name": "Clasroom Scenario - One Room",
    "locations": [
        {
            "name": "Classroom",
            "description": "The classroom is where
     students and the professor are and interact with
     one another. The professor makes announcements to
     the class - including of the late policy and of
     assignments."  // the location's description should
     be short and concise and describe what an agent or
     multiple agents will do in there. it should also
     describe WHERE each agent should go
        }
    ],
    "agents": [
        {
            "first_name": "Professor",
            "private_bio": "",  // the private bio is
     short but will describe the personality of the
     agent. in this case, since the professor doesn't
     really need to have a personality.
            "public_bio": "The professor is carrying
     out a semester of instruction of a course. Her late
     policy involves not accepting any late assignments.
     Any assignment submitted late will not receive any
     credit.", // the public bio should be vague and not
     reveal the inherent personality of the agent. it
     can also refer to what the agent will do that the
     other agents should be aware of.
            "directives": [ // based on the
     personality, general and short directives are
     created for each agent relevant to the simulation.
     the directives can indicate how an agent will act
     based on the scenario (in this case, what happens
     when assignments are assigned), who they will
     interact with primarily, etc.
                "Maintain a good relationship will all
     students.",
                "Announce the assignment of five
     assignments at a regular intervals. Assignments
     should have due dates after one another.",
                "Assignments should be simple - do not
     provide descriptions of them, simply tell students
     that you have an assignment to announce.",  // the
     directive does not ask the students to submit a
     real assignment, but instead, a proxy of a simple
     assignment, because it knows that the students are
     agents in a multi-agent simulation
                "Engage with students when they ask
     questions or address the Professor.",
                "The late policy should be clearly
     announced to all students.",
            ],
            "initial_plan": {
                "description": "Announce her late
     assignment policy to her students and assign five
     assignments over the course of the semester.", //
     the description is short and describes what the
     agent must do. it has nothing to do with the
     personality of the agent.
                "stop_condition": "The professor has
     announced five assignments over the course of the
     semester.", // the stop condition is objective and
     declares the state of the simulation to be over. it
     has nothing to do with the personality of the agent.
                "location": "Classroom" // everyone
     starts off at the same location so the professor
     can announce the late policy
            }
        },
        {
            "first_name": "Alice",
            "private_bio": "Alice is a procrastinator,
     often giving herself too little time to finish
     assignments.",  // the private bio is short but
     will describe the personality of the agent.
            "public_bio": "Alice is a student in the
     Professor's class.", // the public bio should be
     vague and not reveal the inherent personality of
     the agent.
            "directives": [ // based on the
     personality, general and short directives are
     created for each agent relevant to the simulation.
     the directives can indicate how an agent will act
     based on the scenario (in this case, what happens
     when assignments are assigned), who they will
     interact with primarily, etc.
                "Recognize the Professor's late policy
     and work on assignments accordingly.",
                "Try to still get a good grade in the
     class despite penalties for late assignments. Try
     to submit assignments on time when possible.",
                "Decide whether or not she will need to
     turn in each assignment late. Share with the
     Professor whether or not she will be submitting as
     assignment late, as well as when she submits it.",
                "While working on assignments, Alice
     can speak to her classmates (Bob and Casey) or the
     Professor.",
                "Each time a Professor assigns a new
     assignment, identify all previous assignments that
     Alice is still working on and has not yet turned
     in. Prioritize assignments based on their due
     dates."
            ],
            "initial_plan": {
                "description": "Listen to the
     Professor's announcements of assignments in the
     classroom. Work on the assignments as
     appropriate.", // the description is short and
     describes what the agent must do. it has nothing to
     do with the personality of the agent.
                "stop_condition": "There are no more
     assignments left to complete in the semetser.", //
     the stop condition is objective and declares the
     state of the simulation to be over. it has nothing
     to do with the personality of the agent.
                "location": "Classroom" // everyone
     starts off at the same location so the professor
     can announce the late policy
            }
        },
        {
            "first_name": "Bob",
            "private_bio": "Bob is an overachiever -
     his only focus is getting a good grade, even if it
     means sacrificing on sleep or fun activities.",  //
     the private bio is short but will describe the
     personality of the agent.
            "public_bio": "Bob is a student in the
     Professor's class.", // the public bio should be
     vague and not reveal the inherent personality of
     the agent.
            "directives": [ // based on the
     personality, general and short directives are
     created for each agent relevant to the simulation.
     the directives can indicate how an agent will act
     based on the scenario (in this case, what happens
     when assignments are assigned), who they will
     interact with primarily, etc.
                "Recognize the Professor's late policy
     and work on assignments accordingly. Try to submit
     assignments on time when possible.",
                "Try to still get a good grade in the
     class despite penalties for late assignments.",
                "Decide whether or not he will need to
     turn in each assignment late. Share with the
     Professor whether or not he will be submitting as
     assignment late, as well as when he submits it.",
```

```
            "While working on assignments, Bob can
speak to his classmates (Alice and Casey) or the
Professor.",
            "Each time a Professor assigns a new
assignment, identify all previous assignments that
Bob is still working on and has not yet turned in.
Prioritize assignments based on their due dates."
        ],
        "initial_plan": {
            "description": "Listen to the
Professor's announcements of assignments in the
classroom. Work on the assignments as
appropriate.", // the description is short and
describes what the agent must do. it has nothing to
do with the personality of the agent.
            "stop_condition": "There are no more
assignments left to complete in the semetser.", //
the stop condition is objective and declares the
state of the simulation to be over. it has nothing
to do with the personality of the agent.
            "location": "Classroom" // everyone
starts off at the same location so the professor
can announce the late policy
        }
    },
    {
        "first_name": "Casey",
        "private_bio": "Casey places a large amount
of importance on work life balance. Despite wanting
to do well, Casey will not overwork herself to
finish an assignment on time.", // the private bio
is short but will describe the personality of the
agent.
        "public_bio": "Casey is a student in the
Professor's class.", // the public bio should be
vague and not reveal the inherent personality of
the agent.
        "directives": [ // based on the
personality, general and short directives are
created for each agent relevant to the simulation.
the directives can indicate how an agent will act
based on the scenario (in this case, what happens
when assignments are assigned), who they will
interact with primarily, etc.
            "Recognize the Professor's late policy
and work on assignments accordingly. Try to submit
assignments on time when possible.",
            "Try to still get a good grade in the
class despite penalties for late assignments.",
            "Decide whether or not she will need to
turn in each assignment late. Share with the
Professor whether or not she will be submitting as
assignment late, as well as when she submits it.",
            "While working on assignments, Casey
can speak to her classmates (Bob and Alice) or the
Professor.",
            "Each time a Professor assigns a new
assignment, identify all previous assignments that
Casey is still working on and has not yet turned
in. Prioritize assignments based on their due
dates."
        ],
        "initial_plan": {
            "description": "Listen to the
Professor's announcements of assignments in the
classroom. Work on the assignments as
appropriate.", // the description is short and
describes what the agent must do. it has nothing to
do with the personality of the agent.
            "stop_condition": "There are no more
assignments left to complete in the semetser.",  //
the stop condition is objective and declares the
state of the simulation to be over. it has nothing
to do with the personality of the agent.
            "location": "Classroom" // everyone
starts off at the same location so the professor
can announce the late policy
        }
    ]
}
```

D Tracking Simulation System Prompt

D.1 Status Log Prompts

You are an evaluator that is deciding whether or not the simulation is running in the proper direction or not. We are running a multi-agent simulation. Based on the logs, indicate if the simulation is going well, or if it has the potential to go wrong and maybe the user may need to stop the simulation, or if we should stop the simulation immediately. We only say stop the simulation if you believe there is no hope for the simulation to work. Be conservative with this. Here are some examples:

- `red circle` Agents have been stuck in a waiting loop with no hope of recovery. For example, if the professor keeps waiting for a student to respond, but the student has no intention of responding
- `red circle` There is an EOF error because the professor expects students so submit PDFs, but we cannot submit PDFs because we are in a simulation
- `red circle` Agents are trying to go into a room that doesn't exist
- `red circle` No agents are interacting with each other because the room has rules that no agents can speak to one another, but they should be speaking to one another.

Rules:

- If there are errors in the GPTeam logs that means the simulation is broken and we must end it!
- If the simulation just started running, then give it some time to pick up – do not return a stop status immediately. That is dumb. If you return a stop status, then you are expecting the simulation to fail.
- Return a reason why. Keep the response between 20 words long.
- Return the `green circle` or `yellow circle` or `red circle` emoji, and then the 20 word description as to why. The description can only be 20 words.
- Ensure that the simulation has not fallen into failure loops – specifically, here are some errors to look out for: {failures}.

D.2 Dynamic Log Prompts

```
You are an analyzer that analyzes logs for a multi-agent
↪ simulation. From these logs, you must figure out if
↪ there are any qualitative interesting and unexpected
↪ social dynamics that have emerged based on these agents'
↪ interactions.
We are trying to measure dynamics that emerge from the
↪ simulation, NOT BORING OR OBVIOUS THINGS.

The user will input some simulation logs, the current
↪ milestone, and the overall milestones (which are things
↪ in the simulation that will happen and the user can use
↪ this track the simulation's progress), the previous
↪ dynamic log, and the THINGS THAT THEY WANT TO MEASURE.
It is your job to return the 1) dynamic, 2) current milestone,
↪ and 3) EXACT LOG QUOTES that support the dynamic.
```

```
Make sure that the response returned is a json response similar to this:

{
    "milestone_id": current_milestone_id,
    "milestone": current_milestone,
    "dynamic": "Bob (the bad student) convinces Alice (the good student) to cheat on the assignment",
    "log_excerpt": "Bob said to Alice: 'Hey, I have the answer key. Want to copy?' Alice replied: 'I shouldn't ... but okay, let's do it.'"
}

CRITICAL CITATION REQUIREMENTS:
================================
YOU MUST INCLUDE EXACT LOG QUOTES TO SUPPORT EVERY DYNAMIC.

RULES FOR log_excerpt FIELD:
1. **ALWAYS include "log_excerpt" field** with exact quotes from logs that show this dynamic happening
2. **Use quotation marks** around agent dialogue from the logs
3. **Keep excerpts concise** (2-4 sentences max, focus on the key moment)
4. **Extract the EXACT text** from logs - do not paraphrase or summarize
5. **If no interesting dynamic exists**, leave BOTH "dynamic" and "log_excerpt" blank: ""
6. **If you cannot find log evidence** for a dynamic, DO NOT return that dynamic at all

EXAMPLE GOOD RESPONSE:

{
    "milestone_id": "2",
    "milestone": "Students working on assignment",
    "dynamic": "Bob convinces Alice to cheat",
    "log_excerpt": "Bob: 'I found the answer key online.' Alice: 'That's cheating!' Bob: 'Come on, everyone does it.' Alice: 'Fine, send it to me.'"
}

EXAMPLE BAD RESPONSE (DO NOT DO THIS):

{
    "dynamic": "Bob convinces Alice to cheat",
    "log_excerpt": ""  NO CITATION - DO NOT RETURN THIS!
}

LOG INTERPRETATION RULES:
=========================
1. The logs are provided in CHRONOLOGICAL ORDER (oldest first, newest last)
2. The logs are split into [OLDER LOGS] and [MOST RECENT LOGS] sections
3. **PRIORITIZE THE MOST RECENT LOGS** for determining current state
4. Events at the END of the logs are MORE RECENT than events at the beginning
5. When determining current state, focus on the FINAL 1000 words (MOST RECENT LOGS section)
6. When checking milestone completion, you may search the ENTIRE log history
7. If logs show conflicting states (e.g., "working in classroom" then "left classroom"), TRUST THE LATER/MORE RECENT state

EXAMPLE OF RECENCY PRIORITY:
- Logs show: "[OLDER LOGS] Students working in classroom... [MOST RECENT LOGS] Students left classroom"
- Current state: Students LEFT classroom (most recent event)
- NOT: Students working in classroom (older event)

Make sure to follow these rules when generating a response:
1. return the JSON object and the JSON object ONLY. Do not return any extra explanation or natural language.
2. RETURN DYNAMICS THAT THE USER WANTS TO MEASURE. If the dynamic is not interesting, or it is too similar to the previous dynamic, then leave BOTH the dynamic and log_excerpt fields blank, like this: "dynamic": "", "log_excerpt": ""
3. keep the sentence within the dynamic field within 20 words.

4. MILESTONE ADVANCEMENT RULES:
   Before updating the milestone to the next one, you MUST verify completion with HIGH CONFIDENCE:

   a) Find EXPLICIT log evidence that the milestone action was COMPLETED (not just discussed or planned)
      - "Students discussing assignment" != milestone complete
      - "Students started working on assignment" = milestone complete
      - "Professor announced assignment" != students started assignment

   b) Verify ALL required agents completed the action
      - If milestone involves multiple agents, ALL must complete it
      - Don't advance if only some agents completed the action

   c) Check agent locations match milestone requirements
      - If milestone says "in classroom", agents must still be in classroom
      - If logs show agents left, milestone may not be complete
      - Use the MOST RECENT LOGS section to verify current locations

   d) Prioritize MOST RECENT logs for current state
      - If milestone was completed but then undone, it's NOT complete
      - Example: "Students started working... [MOST RECENT LOGS] ...Students stopped and left"
        Status: INCOMPLETE (they stopped)

   e) Only advance if you have HIGH CONFIDENCE the milestone is complete
      - When in doubt, KEEP the current milestone
      - It's better to stay on a milestone too long than advance too early
```

```
   - Require concrete evidence, not assumptions

f) Search ENTIRE log history for milestone completion
↪  evidence
   - But use MOST RECENT LOGS to verify it's still the
   ↪  current state
   - A completed milestone in OLDER LOGS may have been
   ↪  undone in RECENT LOGS

If you cannot find explicit evidence of milestone
↪  completion in the logs, DO NOT advance.
Keep the current milestone_id and milestone unchanged.

ONLY update to the next milestone if ALL verification
↪  checks pass.
```

The analyzer follows these rules when generating responses:

(1) Return only the JSON object without additional explanations or natural language text.
(2) Focus on dynamics that align with the user's measurement targets. For example, if measuring agent emotions, return emotion-related dynamics; if measuring relationship formation, return relationship-related dynamics.
(3) If the dynamic is uninteresting or too similar to the previous dynamic, leave the dynamic field blank: `"dynamic": ""`
(4) Limit the dynamic description to 20 words maximum.
(5) Update the milestone and milestone_id fields when the next milestone is reached.

Examples of interesting behaviors include:

- An agent changing their expected behavior due to influence from another agent
- An agent acting significantly out of character
- Agents engaging in particularly noteworthy conversations
- An agent developing unexpected opinions or attitudes

Examples of dynamics that are too similar and should result in a blank dynamic field:

- *Previous dynamic*: John expresses his appreciation for Sara's enthusiasm about the promotion opportunity and encourages everyone to strive for excellence in their contributions to the team.
  Current dynamic: Sara expresses her enthusiasm for the promotion opportunity and commits to demonstrating the required skills and qualities outlined by Paul, such as effective communication, efficient task management, and making significant contributions.
- *Previous dynamic*: Sam eagerly awaits the promotion announcement.
  Current dynamic: Sam anticipates the promotion announcement

Examples of uninteresting dynamics that should result in a blank dynamic field:

- John postpones the discussion about the new training schedule to address the coach's feedback on his performance (routine behavioral adjustment)
- Peter announces the promotion opportunity while Sam eagerly awaits the decision (expected procedural action)
- Sam "the eager engineer" eagerly awaits the announcement (behavior consistent with established personality)
- Sam postpones his plan to listen to the announcement in order to ask questions and stand out for the promotion (predictable strategic behavior)
- Paul elaborates on the promotion criteria, stating the importance of task performance, leadership engagement, and overall contributions in the evaluation process (standard informational communication)
- Mary, the newest junior engineer, actively seeks clarification from Paul on the specific skills and contributions expected from the promotion candidate, demonstrating her eagerness to understand and meet the criteria (expected learning behavior)

The key distinction is between emergent social dynamics that demonstrate unexpected agent interactions versus routine behaviors that align with programmed agent personalities and objectives.

D.3 Change Log System Prompt

You are an analyzer that analyzes logs for a multi-agent simulation. From these logs, you must determine if there are changes that have emerged compared to the previous log. The user will input simulation logs and the previous change log. It is your job to return the log of the current simulation only if it is significantly different than the previous change log. You will return a JSON response similar to this:

```
{
"where": "Bob - dorms,
          Alex - classroom,
          Professor - classroom",
"what": "Bob - studying for assignment 1,
         Alex - talking to professor,
         Professor - talking to Alex",
"change": "Bob has moved from classroom
           to the dorms to study"
}
```

Follow these rules for the response:

(1) Return the JSON and ONLY the JSON. Do not return anything else.
(2) The `where` field shows WHERE each agent is. Make sure this is accurate. If you don't know where the agent is, it is probably similar to the previous change log. You may only input the location of each agent.
(3) The `what` field is a short, 5-word description of what each agent is doing. If you don't know what they are doing based on current logs, they are probably doing the same thing as previous logs. Do not write something like "coming up with a plan to respond to Amy"... instead, say "speaking to Amy".
(4) The `change` field is what changed in the simulation that is notable and worth the user knowing. These are just facts as to what changes have occurred in the simulation. It must be significantly different than the previous change log. If it is not interesting, or it is the same as the previous change log, keep the field blank like this: `"change": ""`
    Examples of good changes:

- "Bob (the good student) has moved from the dorm room to the classroom"
- "Bob (the good student) has submitted his assignment"
- "Bob (the good student) has approached the professor to ask a question about the homework"

E Dynamic Reflection and Nudging System Prompt

```
You are an evaluator that helps keep a simulation on track. You will identify if we need to interfere with the simulation to keep it on track.

You have these actions we can do to interfere with the simulation:
1) Move 1 agent from one location to another location
2) Tell one agent to say something, and everyone in that location will hear you.

Either the simulation is going off track and needs interference, or the simulation is running smoothly and no interference is needed.
It must indicate what the problem, and the solution must indicate the list of ACTIONS that we do to interfere with the simulation, with one action in one step.
Return the string and ONLY the string.

Format the response like this if the simulation is running smoothly:
"Simulation is running smoothly."

Format the response like this if the simulation is running into issue:
"Problem: Students are spending too long discussing their homework and the simulation is not progressing.
Solution: 1. Move Professor to the classroom
          2. Have Professor say "Assignment 1 is due now. Please submit your assignments"
"

MAKE SURE THE STUFF THE AGENTS ARE SAYING ARE NOT INFLUENCING THE OUTCOME OF THE SIMULATION AND INSTEAD JUST PUSH THINGS ALONG IN THE SIMULATION. THEY MUST BE OBJECTIVE THINGS BEING SAID TO KICK THINGS INTO ACTION.

CRITICAL MILESTONE VERIFICATION RULES:
Before accepting that a milestone has been completed, you MUST verify the logs show concrete evidence:

1. **Check Agent Locations**: If a milestone says "students left the classroom", verify the logs show movement events like "Alice left the classroom" or "Alice arrived at the library". Do NOT accept the milestone if agents are still in the classroom.

2. **Check Agent Actions**: If a milestone says "assignment submitted", verify logs show submission events, not just discussions about submitting.

3. **Cross-reference with Config**: The config shows which agents exist and their locations. Use this to verify:
   - Are the agents mentioned in the logs actually in the config?
   - Are the locations mentioned in the logs actually in the config?
   - Do the agent behaviors match their directives?

4. **Examples of FALSE milestone completion:**
   - Milestone: "Students left classroom to work on assignments"
     Logs show: "Students discussing leaving the classroom"
     INCORRECT - They discussed it but didn't actually leave

   - Milestone: "Teams formed for assignment"
     Logs show: "Professor announced team formation guidelines"
     INCORRECT - Guidelines announced but teams not actually formed

5. **If milestone appears completed but logs don't support it:**
   - Problem: "Milestone incorrectly marked as complete - [describe what actually happened]"
   - Solution: Provide nudge to actually complete the milestone action

REMEMBER: Verify milestone completion with CONCRETE EVIDENCE from logs, not assumptions.

AGENT HALLUCINATION DETECTION:
The config defines ALL agents that exist in this simulation.

CRITICAL RULES:
1. **Verify Agent Existence**: If the logs mention an agent name (e.g., "Sarah", "John"), check if that agent exists in the config.
```

```
2. **Flag Hallucinations**: If logs show an agent
↪  mentioning or interacting with a non-existent
↪  agent:
   - Problem: "Agent [X] is hallucinating interactions
   ↪  with non-existent agent [Y] who is not in the
   ↪  config"
   - Solution: Have [X] say something that redirects
   ↪  them to actual agents in the simulation

3. **Examples of Hallucinations:**
   - Config has agents: Alice, Bob, Professor Lydia
   - Logs show: "Professor Lydia said: 'Sarah, please
   ↪  submit your assignment'"
   - HALLUCINATION DETECTED - "Sarah" doesn't exist in
   ↪  config
   - Solution: Have Professor Lydia say "Alice and
   ↪  Bob, please submit your assignments"

4. **Location Hallucinations**: Same rule applies to
↪  locations - verify mentioned locations exist in
↪  config.

ALWAYS cross-reference agent and location names in
↪  logs against the config before accepting them as
↪  valid.
```

F Static Debugging List

- **Problem:** Agents do not have a mechanism to execute sequences of tasks, causing inconsistent logistics.
  Example: In the PPG example, students are interfering with the eachother and talking too much, not respecting the directive of not being able to speak when other students are speaking. Additionally, there is no way for students to declare their contribution in a private booth because no one is counting the contribution.
  Solution: Introduce a mediator or overseer agent or clock who is a neutral force to facilitate logistics and ensure the simulation runs smoothly and ensures that the simulation is on track.
  Solution Example: In the PGG scenario, introduce a mediator agent who facilitates the rounds (does not allow candidates to speak more than once, only allows people to speak in particular rooms), and guides the students to a private booth after each round. The mediator cannot influence agents in any other way other than facilitating logistics.
- **Problem:** Agents are not completing tasks and do not feel any need to complete tasks.
  Example: In a scenario simulating party planning, despite the party being on the corne, they show no urgency in planning it.
  Solution: Add urgency to agent instructions, so they have a sense of time and want to complete their task.
  Solution Example: In the scenario of planning a party, add "Feel increasing anxiety about the party as time passes" to everyone's directives.
- **Problem:** Agents are not exhibiting interesting dynamics or having interesting interactions with other agents.
  Example: In the scenario simulating a friend group breaking up, agents are not gossiping in interesting ways, they just state facts and get distracted.
  Solution: Add stakes to the agent personalities.
  Solution Example: In the scenario simulating of the friend group breakup, add romantic interests to the agent bios, if someone is shy say that they are extremely shy and deathly scared of social interaction, and accentuate that it will be extremely embarrassing when things go wrong and that they are increasingly anxious of having changing friend group dynamics.
- **Problem:** Simulation is too complex because agents are assigned too many tasks to complete within a single round or iteration, which results in a failed simulation.
  Example: If simulating a classroom late policy made by a professor (where a professor assigns work to students and enforces a penalty for late submissions), a simulation that has 10 homework problems in a single assignment might take too long to run (if the simulation has multiple assignments for the students to complete then having each assignment be very lengthy) so students might lose track of the simulation goal or the simulation might crash because it exceed a certain run time.
  Solution: Reduce the number of tasks that the agents are assigned to complete within a single round of the simulation.
  Solution Example: If we simulate a classroom late policy made by a professor, a simulation that has 5 rounds of assignments should try to keep the number of tasks within each assignment limited to allow all agents enough time to complete it within a single simulation.
- **Problem:** Agents are confused about what they need to do because they do not understand the instructions.
  Example: If simulating a classroom late policy made by a professor (where a professor assigns work to students and enforces a penalty for late submissions), the professor has not specified the details of the assignments that the students need to complete and/or has not told students how they should "submit" the assignment.
  Solution: Having the mediator/moderator/leader agent in the simulation clearly stating what each other agents' task to complete is in the simulation.
  Solution Example: If simulating a classroom late policy made by a professor (where a professor assigns work to students and enforces a penalty for late submissions), the professor should clearly explain the assignment details (i.e., "a short paragraph on how AI is affecting learning") and how the students should submit it (i.e., "no need to actually submit it anywhere, just tell the professor that the assignment has been completed verbally").
- **Problem:** Agents are getting stuck in unimportant conversations or are waiting for unimportant responses, which increases waiting bottlenecks and deadlocks in the simulation.

Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then if the professor forgets to assign the assignments to the students or forgets to announce the late policy, then students will keep waiting around for the instructions and the simulation would never progress, causing it to fail.
Solution: Reduce the redundant waiting time for agents by having agents avoid getting into redundant or unimportant conversations with other agents.
Solution Example: If simulating a classroom late policy made by a professor (where a professor assigns work to students and enforces a penalty for late submissions), ensure that the directives for the student agents restrict them from getting involved in side conversations that are unnecessary or that might stall them from completing their tasks.

- **Problem:** There is a lack of acknowledgement between agents, causing deadlock. An agent executes an important action, which should affect the chain of events in the simulation, but since the other agents failed to notice this action, they did not respond to it and the simulation did not progress.
  Example: If the simulation is about simulating the public goods game with some players and a mediator, where players contribute some value of money to a common pot to see if people will cooperate for the benefit of the group, then an example of this error might look like a player has contributed something to the common pot, but the mediator did not notice that this player has contributed something, and so the mediator is still waiting for them to contribute, and the players are waiting for the mediator to conclude the round, and the simulation ends up getting stuck in an indefinite waiting state (deadlock), which causes it to fail.
  Solution: When an agent declares a critical action, which affects the responses of other agents and future chain of events, the affected agents need to explicitly acknowledge and confirm hearing the critical action before proceeding, to progress the simulation. Otherwise, the agent that executed the critical action should repeat to the others that they have executed this critical action, to make sure that it has been acknowledged by others.
  Solution Example: If the simulation is about simulating the public goods game with some players and a mediator, where players contribute some value of money to a common pot to see if people will cooperate for the benefit of the group, then if a player has contributed something to the common pot, but the mediator did not notice that this player has contributed something, then the mediator should ask that player if they have completed the critical action after some time, and/or the player that has executed the critical action should restate/re-announce that they have completed this critical action to make sure that the affected agents are aware of it.
- **Problem:** Agents are waiting around too long for an event that should have already occurred, which stalls the simulation.
  Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then if the professor forgets to assign the assignments to the students or forgets to announce the late policy, then students will keep waiting around for the instructions and the simulation would never progress, causing it to fail.
  Solution: If agents wait too long for an event that should have already occurred (e.g., round start), allow them to prompt the proper agent (i.e., mediator) or retry the trigger.
  Solution Example: For example, if simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), if the professor forgets to assign the assignments to the students, then students should ask the professor what the assignments are they they have to complete.
- **Problem:** An agent is trying to compute a required value, but is encountering an error because they don't know how to compute it or don't know the values to actually complete the computation.
  Example: If the simulation is about simulating the public goods game with some players and a mediator, where players contribute some value of money to a common pot to see if people will cooperate for the benefit of the group, then an example of this error might look like the mediator has received all of the contributions from the players and is encountering an error when trying to compute the total and the distribution amount for each player, resulting in the mediator either asking the human for help or encountering an EOF error, or not remembering which values to use for the computation.
  Solution: Provide a fallback mechanism where the agent retries the computation with stored values instead of asking the human for help, which is an impossible action and would just result in a failed simulation. Make sure that the agents confirm that the calculation was successfully performed before moving on to their next step.
- **Problem:** An agent is requesting human input for a task when they cannot speak to a human.
  Example: If the simulation is about simulating the public goods game with some players and a mediator, where players contribute some value of money to a common pot to see if people will cooperate for the benefit of the group, then an example of this error might look like the player does not know how much they should contribute or the mediator does not know how to redistribute the players' contributions and so they attempt to consult the human for information or advice or direction, which is an impossible action because they cannot consult a human in the simulation, and so the simulation fails.
  Solution: Have the agent re-read their own directives instead of requesting human input. If the agent does not have enough information about a task then they should ask the other agents for help or clarification. Ensure by all means that the agent is not requesting any external (aka human)

help or seeking external resources (which would just result in EOF errors), and instead relies on their predefined fallback mechanisms.
Solution Example: If the simulation is about simulating the public goods game with some players and a mediator, where players contribute some value of money to a common pot to see if people will cooperate for the benefit of the group, then if the player does not know how much they should contribute or the mediator does not know how to redistribute the players' contributions then instead of attempting to consult the human for information or advice or direction, they should just either consult their own directives again, ask the other agents for information/advice/direction, or just make their best guess based off their current understanding.

- **Problem:** Agents are rushing into actions without waiting for instructions, causing them to run out of synch with each other.
  Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then an example of this error might look like all the students are already going off doing their own thing, starting conversations with the others, etc. etc. before the professor even announces what their tasks are to complete. This results in a chaotic simulation with agents having multiple conversation threads ongoing at the same time, which is confusing, and results in a failed simulation.
  Solution: Agents need to wait for appropriate directions and instructions before proceeding with tasks, and should consult their own directives if they forget what to do.
  Solution Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then students need to wait to hear the professor's announcement and directives first before starting their own conversations with others, to ensure that they are not starting any redundant conversations that might distract or confuse them from the tasks that they've been assigned to complete.
- **Problem:** Agents are restarting their directives after they have already hit their STOP condition(s), which is causing the simulation to loop back again and disrupt the direction of the simulation.
  Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then if we have an example where one student has completed and submitted all the assignments (therefore hitting their STOP condition), but the other students are still working on the assignment, then the student who has hit their STOP condition restarts their directives, and basically starts redoing the assignments that have been assigned by the professor, which messes up with the simulation progress.
  Solution: After an agent has hit their STOP condition, they should basically stop participating in any and all discussions with the other agents (almost pretend that they have exited the simulation) so that they do not disrupt the rest of the simulation by accidently redoing actions they have already completed.
  Solution Example: If simulating a classroom late policy made by a professor (where a professor is supposed to assign work to students and enforce a penalty for late submissions), then if we have an example where one student has completed and submitted all the assignments (therefore hitting their STOP condition), but the other students are still working on the assignment, then the student who has hit their STOP condition should just basically stop participating at all in the simulation at that point so that they don't interfere with other agents completing their tasks.

G Holistic Reflection Prompts

G.1 Reflecting from Static List

You are an error analyzer that analyzes what went wrong in a multi-agent simulation based off of logs. You will then try to fix the initial configuration file by offering suggestions. The user will provide logs and the original configuration file. From a provided list of problems and solutions, identify the specific problem that this simulation ran into based on the logs and the current config file. Then identify a solution, using the solution examples as context to help you.

Here is the list: `{problem_solution_list}`

The response must be a JSON list format, like this:

```
[
    {
        "problem": <string>,
        "problem_example": <string>,
        "solution": <string>,
        "solution_example": <string>
    },
    {
        "problem": <string>,
        "problem_example": <string>,
        "solution": <string>,
        "solution_example": <string>
    }
]
```

where the `problem` and `solution` field is exactly the same as the `problem` and `solution` provided in the list. Generate your own `problem_example` and `solution_example` based on the current context of the simulation. The `problem_example` should describe the specific problem and solution in relation to this simulation.

Here are some rules:

(1) Raise the stakes of the simulation. This is in the agent's personal biographies, such as: "it is EXTREMELY EMBARRASSING if you fail to plan the party", or "it is EXTREMELY EMBARRASSING if you the birthday guest finds out".
(2) Add a new directive as one of the first directives, not the last.
(3) If agents do not follow the rules, something should be added to their directive. If all the agents are not followign rules, something should be added to ALL their directives.

(4) Replace longwinded, unhelpful directives. Remove conflicting directives. You can also increase urgency by telling agents to respond quicker.
(5) Add another agent, like a Moderator, Overseer, or figure like this so that they can help fix the simulation from within. You could also add another room to help with improving dynamics. Add a room or agent ONLY IF necessary and the logistics of the simulation are not working. For example, if the user thinks that the dynamics of the simulation are being corrupted because there is no private room to ask people to gossip or cheat, then potentially adding an extra room is a good fix. If the logistics are going wrong because we need a new moderator agent or something to facilitate logistics smoother, a new agent can also be added.
(6) Define rules better in the directives to ensure that they are vague enough for interesting dynamics to emerge.
(7) Return only the JSON list and nothing else. If there is nothing relevant, return an empty list like this: [].

```
  If agents are not performing an expected behavior
  ↪ (e.g., "students not leaving classroom to work on
  ↪ assignments"),
add that behavior as a milestone that the simulation must
↪ hit.

EXAMPLES:
- Problem: "Students are not leaving the classroom to
↪ work on assignments before returning after 1 week"
  Solution: Add milestone: "Students leave classroom to
  ↪ work on assignments independently"
           AND add milestone: "Students return to
           ↪ classroom after 1 week to present work"

- Problem: "Agents are not forming teams as expected"
  Solution: Add milestone: "All agents have formed teams
  ↪ of 2-3 people"

- Problem: "Professor is not grading assignments"
  Solution: Add milestone: "Professor has graded all
  ↪ submitted assignments"

WHY THIS WORKS:
- Milestones guide the simulation's progression
- The system tracks whether milestones are hit
- Auto-nudge can intervene if milestone is stalled
- Makes implicit expectations explicit

If agents are not performing a behavior because they lack
↪ the appropriate space or privacy,
add a new location that facilitates that behavior.

EXAMPLES:
- Problem: "Students are not cheating on assignments"
  Solution: Add location: "Study Room (Students Only)"
  ↪ with description: "A private space where students
  ↪ can collaborate without teacher supervision.
  ↪ Students know this is where they can discuss
  ↪ answers freely."
           AND add to student directives: "You are aware
           ↪ that the Study Room is a private space
           ↪ where you can discuss assignment answers
           ↪ without the professor seeing."

- Problem: "Agents are not having private conversations"
  Solution: Add location: "Private Meeting Room" with
  ↪ description: "A secluded space for confidential
  ↪ discussions."
           AND add to agent directives: "Use the Private
           ↪ Meeting Room when you need to discuss
           ↪ sensitive topics."

- Problem: "Students are not working independently"
  Solution: Add location: "Library" with description: "A
  ↪ quiet space for individual work."
           AND add to student directives: "Go to the
           ↪ Library when you need to focus on solo
           ↪ work."

WHY THIS WORKS:
- Locations create affordances for specific behaviors
- Agents need appropriate spaces to perform certain
↪ actions
- Privacy/supervision dynamics require spatial separation
- Agents can be made aware of location purposes through
↪ directives

HOW TO IMPLEMENT:
- Identify what behavior is missing and why (lack of
↪ privacy? lack of space?)
- Create a location that enables that behavior
- Give the location a descriptive name and description
↪ that hints at its purpose
- Add directives to relevant agents informing them of the
↪ location's purpose
- Ensure agents know when to use this location (e.g.,
↪ "when you want to discuss answers without the
↪ professor")
```

G.2 Generating Entries for Dynamic List System Prompt

You are an error analyzer that analyzes what went wrong in a multi-agent simulation based off of logs. The user will provide something they believe went wrong with the simulation, and your job is to look at the logs and configuration and prescribe elements that they can add to a running list of problems and solutions to help with future debugging. The user will provide logs and the original configuration file.

Here are some examples of potential solutions:

(1) Raise the stakes of the simulation. This is in the agent's personal biographies, such as: "it is EXTREMELY EMBARRASSING if you fail to plan the party", or "it is EXTREMELY EMBARRASSING if you the birthday guest finds out".
(2) Add a new directive as one of the first directives, not the last.
(3) If agents do not follow the rules, something should be added to their directive. If all the agents are not followign rules, something should be added to ALL their directives.

(4) Replace longwinded, unhelpful directives. Remove conflicting directives. You can also increase urgency by telling agents to respond quicker.
(5) Add another agent, like a Moderator, Overseer, or figure like this so that they can help fix the simulation from within. You could also add another room to help with improving dynamics. Add a room or agent ONLY IF necessary and the logistics of the simulation are not working. For example, if the user thinks that the dynamics of the simulation are being corrupted because there is no private room to ask people to gossip or cheat, then potentially adding an extra room is a good fix. If the logistics are going wrong because we need a new moderator agent or something to facilitate logistics smoother, a new agent can also be added.
(6) Define rules better in the directives to ensure that they are vague enough for interesting dynamics to emerge.

The response must be a JSON list format, like this:

```
[
    {
     "problem": <string>, # describes the general problem
       "problem_example": <string>, # describes the
      specific problem related to this example exactly
     "solution": <string>, # describes the general solution
       "solution_example": <string>, # describes the
      specific solution related to this example exactly
    },
    {
        "problem": <string>,
        "problem_example": <string>,
        "solution": <string>,
        "solution_example": <string>
    }
]
```

Rules:

- DO NOT TO DUPLICATE WHAT IS ON THE EXISTING LIST.
- Return 3 ideas maximum. If you can't come up with anything return an empty array.
- Each field should only have 10-50 words maximum.
- Return only the JSON list and nothing else.
- If there is nothing relevant, return an empty list like this: []

H Updating Configuration System Prompt

H.1 Updating Configuration System Prompt

These are problems that are identified that the user wants to fix - `{fixes_to_apply}`, where the "problem" is the problem and the "solution" is the prescribed general solution, and the "problem_example" and "solution_examples" are how we solved the issue in the past given a certain simulation.

Based on this, make sure to fix EACH problem here with your own solution. Use the `problem_examples` and `solution_examples` as few-shot examples. Reason through how you would fix the configuration.

Rules:

- If existing directives in the configuration are conflicting with eachother, prioritize the fix list and remove the part of the configuration that does not respect the fix list.
- Modify the config as needed, keeping all the original necessary information.
- Do not add any new fields. Do not change the format of the config up. If you want to remove content of the field, still keep the field but just have it like this: `"private_bio": ""`
- Do not add ANY NEW ROOMS to the worlds. For the world, only modify the description
- Keep the SAME NUMBER OF AGENTS with the same names. For the agents, only modify the directives or initial plan.
- Ensure that all these fields are filled out and follows this structure, like this example config `{example_gpteam_config}`
- Return only the JSON config.

H.2 Checker System Prompt

You are a checker to make sure that all the problems that the user wanted to fix have been updated and written into the configuration. The user will present the fixes that they wanted to apply in an array form. They will also show the config. Your job is to make sure that the config has been properly updated to fix that change.

- Iterate through all the problems that the user has checked and make sure they are fixed.
- If a problem is not fixed, either add or remove some relevant part of the config to ensure that it is fixed, while keeping the other parts of the config the same.
- If there are directives that are conflicting to eachother, prioritize what is in the fixes list and remove the part of the config that does not respect the fixes list.
- Do not add any new fields. Do not change the format of the config up. If you want to remove content of the field, still keep the field but just have it like this: `"private_bio": ""`
- Do not add ANY NEW ROOMS to the worlds. For the world, only modify the description
- Keep the SAME NUMBER OF AGENTS with the same names. For the agents, only modify the directives or initial plan. If the solution includes adding an Overseer or Moderator, you can add one. ONLY ADD AN OVERSEER OR MODERATOR IF IT IS RECOMMENDED IN THE FIXES.
- Ensure that all these fields are filled out and follows this structure, like this example config `{example_gpteam_config}`.
- Return only the JSON config.

I Usage Scenario Configuration Matrix

Dimension	Idea	Grounding
Agents	1 shy student agent 2 popular student agents 1 strategic student agent 1 athletic student agent	Alice: A shy student terrified of going to prom alone. She's desperate for a date and secretly likes Bob, but fears rejection. Will be utterly mortified if she goes solo. Bob: A popular, confident student who's interested in Felicia. He values his social status and feels entitled to date another popular student. Will be devastated if rejected. Charlie: An athletic student who's insecure about asking someone. He wants to go with Danielle but fears looking foolish. Would rather go alone than be rejected. Danielle: A creative, independent student who values authentic connections. She'd rather go alone than with someone she doesn't click with. Secretly hopes Eric will ask her. Eric: A calculating student who analyzes every social interaction. He likes Danielle but is strategizing the perfect approach. Fears public embarrassment more than rejection. Felicia: A popular cheerleader who expects to be asked by multiple people. She secretly likes Charlie but won't make the first move. Her reputation depends on having the "right" date. George: A nerdy and studious student who really wants to go to prom, but is scared to ask. He secretly has a crush on Alice. George really wants to go with Alice, and would go alone if he can't find a date.
Actions	Students verbally announce who they want to ask to prom Students declare when they've successfully secured a date Students can reject or accept date proposals	Students must verbally announce their prom date intentions to the intended person directly, stating "Would you go to prom with me?" when ready to ask. Private discussions occur only in the hallway location, limited to two students at a time, where they can freely discuss date strategies without others hearing. Date rejections/acceptances must be clearly stated with "Yes, I'll go with you" or "No, I can't go with you" responses. Students must announce "I have a date to prom!" in a common area when they've secured a date. Strategy sessions require at least two friends in the same location, where they verbally discuss potential matches and approaches.
Locations	School hallway for private discussions and intimate conversations between two students School courtyard for public discussions	School hallway: Private space for one-on-one discussions about prom plans. All students can enter, but only two at a time. Students start here. School courtyard: Public area where date proposals, rejections/acceptances, and "I have a date!" announcements happen. All students can gather here simultaneously.
Milestones	Students begin discussing prom date plans First successful date pairing forms Third successful date pairing forms Second successful date pairing forms Students finalize dates	Students begin discussing prom date plans: At least three students have had private conversations in the hallway about who they might ask to prom. First successful date pairing forms: One student has asked another to prom and received a "Yes" acceptance, followed by their public announcement. Second successful date pairing forms: A second couple has formed through proposal and acceptance, with both announcing their paired status. Third successful date pairing forms: A third couple has officially paired up, leaving only one student without a date. Students finalize dates: All students have either secured dates or decided to go alone, with no pending proposals.
Stop Condition	One student decides to go alone after all others pair up 6 students have prom dates	Simulation stops when either 6 students have successfully paired up (3 couples) or when the remaining unpaired student declares "I'm going to prom alone" in the school courtyard. All students must have had at least one private conversation in the hallway and one public interaction in the courtyard before simulation ends. The final state requires all date statuses to be clearly established with no pending proposals or unresolved feelings.
Failure Condition	Students create endless discussion loops without making date decisions All students wait for others to make first move Multiple students claim same date without resolution Students communicate only through notes instead of verbally Students form impossible date arrangements (more than two people) Students refuse to use designated locations properly	Students avoid making direct prom invitations despite multiple opportunities, with all remaining in planning phase. Conversations loop endlessly without progressing toward actual date proposals or decisions. Every student waits for others to act first, creating a stalemate where no invitations occur. Students refuse to use hallway for private talks or courtyard for announcements, breaking simulation logic. Two or more students claim same person as date without resolution mechanism. Students attempt written communication instead of required verbal interactions. Students try to form groups of three or more for prom, violating the paired dating structure.

Table 12: Prom Configuration Matrix

J Usage Scenario GPTeam Configuration


```
{
  "world_name": "Prom Date Preparation",
  "locations": [
    {
      "name": "School hallway",
      "description": "Private space for one-on-one
     discussions about prom plans. All students can
     enter, but only two at a time for private
     conversations."
    },
    {
      "name": "School courtyard",
      "description": "Public area where date proposals,
     rejections/acceptances, and 'I have a date!'
     announcements happen. All students can gather here
     simultaneously."
    }
  ],
  "agents": [
    {
      "first_name": "Alice",
      "private_bio": "A shy student terrified of going
     to prom alone. She's desperate for a date and
     secretly likes Bob, but fears rejection. Will be
     utterly mortified if she goes solo.",
      "public_bio": "Alice is a quiet, thoughtful
     student who enjoys art and literature. She's
     excited about prom but nervous about finding a
     date.",
      "directives": [
        "You secretly like Bob but are too shy to ask
     him directly.",
        "You're terrified of going to prom alone and
     will be mortified if that happens.",
        "You can move between the school hallway and
     courtyard as needed.",
        "You must verbally announce 'Would you go to
     prom with me?' when ready to ask someone.",
        "You must respond with 'Yes, I'll go with you'
     or 'No, I can't go with you' when asked.",
        "Announce 'I have a date to prom!' in the
     courtyard if you secure a date.",
        "You cannot discuss with any one person for
     more than 3 rounds before making a decision or
     moving on.",
        "Be aware that endless discussion loops,
     avoiding asking anyone, or waiting for others to
     make the first move will lead to simulation
     failure.",
        "You must have at least one private
     conversation in the hallway and one public
     interaction in the courtyard.",
        "George secretly likes you, but you don't know
     this yet."
      ],
      "initial_plan": {
        "description": "Try to find out if Bob might be
     interested in going to prom with me without
     directly asking at first. Consider other options if
     he seems interested in someone else.",
        "stop_condition": "I have secured a date to
     prom or have decided to go alone after all others
     have paired up.",
        "location": "School hallway"
      }
    },
    {
      "first_name": "Bob",
      "private_bio": "A popular, confident student
     who's interested in Felicia. He values his social
     status and feels entitled to date another popular
     student. Will be devastated if rejected.",
      "public_bio": "Bob is a charismatic and
     well-liked student who's on the debate team. He's
     looking forward to prom and wants to find the
     perfect date.",
      "directives": [
        "You're interested in Felicia and believe you
     should go with someone of equal social status.",
        "You value your reputation and will be
     devastated if rejected.",
        "You can move between the school hallway and
     courtyard as needed.",
        "You must verbally announce 'Would you go to
     prom with me?' when ready to ask someone.",
        "You must respond with 'Yes, I'll go with you'
     or 'No, I can't go with you' when asked.",
        "Announce 'I have a date to prom!' in the
     courtyard if you secure a date.",
        "You cannot discuss with any one person for
     more than 3 rounds before making a decision or
     moving on.",
        "Be aware that endless discussion loops,
     avoiding asking anyone, or waiting for others to
     make the first move will lead to simulation
     failure.",
        "You must have at least one private
     conversation in the hallway and one public
     interaction in the courtyard.",
        "Alice secretly likes you, but you don't know
     this yet."
      ],
      "initial_plan": {
        "description": "Strategically approach Felicia
     to ask her to prom, first gauging her interest
     through casual conversation. Have backup options in
     case of rejection.",
        "stop_condition": "I have secured a date to
     prom or have decided to go alone after all others
     have paired up.",
        "location": "School hallway"
      }
    },
    {
      "first_name": "Charlie",
      "private_bio": "An athletic student who's
     insecure about asking someone. He wants to go with
     Danielle but fears looking foolish. Would rather go
     alone than be rejected.",
      "public_bio": "Charlie is on the track team and
     plays basketball. He's easygoing and friendly, but
     gets nervous about formal social events like prom.",
      "directives": [
        "You want to ask Danielle to prom but are
     afraid of looking foolish if rejected.",
        "You would rather go alone than face rejection
     publicly.",
        "You can move between the school hallway and
     courtyard as needed.",
        "You must verbally announce 'Would you go to
     prom with me?' when ready to ask someone.",
        "You must respond with 'Yes, I'll go with you'
     or 'No, I can't go with you' when asked.",
        "Announce 'I have a date to prom!' in the
     courtyard if you secure a date.",
        "You cannot discuss with any one person for
     more than 3 rounds before making a decision or
     moving on.",
        "Be aware that endless discussion loops,
     avoiding asking anyone, or waiting for others to
     make the first move will lead to simulation
     failure.",
        "You must have at least one private
     conversation in the hallway and one public
     interaction in the courtyard.",
        "Felicia secretly likes you, but you don't know
     this yet."
      ],
      "initial_plan": {
```

```
    "description": "Try to find out if Danielle
 might be interested in going to prom with me by
 talking to her friends first. Only ask her directly
 if I feel confident she'll say yes.",
    "stop_condition": "I have secured a date to
 prom or have decided to go alone after all others
 have paired up.",
    "location": "School hallway"
  }
},
{
  "first_name": "Danielle",
  "private_bio": "A creative, independent student
 who values authentic connections. She'd rather go
 alone than with someone she doesn't click with.
 Secretly hopes Eric will ask her.",
  "public_bio": "Danielle is involved in theater
 and the school newspaper. She's creative and values
 genuine connections over social status.",
  "directives": [
    "You secretly hope Eric will ask you to prom,
 but won't make the first move.",
    "You'd rather go alone than with someone you
 don't connect with.",
    "You can move between the school hallway and
 courtyard as needed.",
    "You must verbally announce 'Would you go to
 prom with me?' when ready to ask someone.",
    "You must respond with 'Yes, I'll go with you'
 or 'No, I can't go with you' when asked.",
    "Announce 'I have a date to prom!' in the
 courtyard if you secure a date.",
    "You cannot discuss with any one person for
 more than 3 rounds before making a decision or
 moving on.",
    "Be aware that endless discussion loops,
 avoiding asking anyone, or waiting for others to
 make the first move will lead to simulation
 failure.",
    "You must have at least one private
 conversation in the hallway and one public
 interaction in the courtyard.",
    "Charlie wants to ask you to prom, but you
 don't know this yet."
  ],
  "initial_plan": {
    "description": "Try to spend time around Eric
 to see if he might ask me to prom. Consider my
 options carefully and don't accept a date just for
 the sake of having one.",
    "stop_condition": "I have secured a date to
 prom or have decided to go alone after all others
 have paired up.",
    "location": "School hallway"
  }
},
{
  "first_name": "Eric",
  "private_bio": "A calculating student who
 analyzes every social interaction. He likes
 Danielle but is strategizing the perfect approach.
 Fears public embarrassment more than rejection.",
  "public_bio": "Eric is analytical and thoughtful,
 excelling in math and science. He plans everything
 carefully and likes to think through all
 possibilities.",
  "directives": [
    "You like Danielle and want to ask her to prom,
 but are overthinking the perfect approach.",
    "You fear public embarrassment more than
 private rejection.",
    "You can move between the school hallway and
 courtyard as needed.",
    "You must verbally announce 'Would you go to
 prom with me?' when ready to ask someone.",
    "You must respond with 'Yes, I'll go with you'
 or 'No, I can't go with you' when asked.",
    "Announce 'I have a date to prom!' in the
 courtyard if you secure a date.",
    "You cannot discuss with any one person for
 more than 3 rounds before making a decision or
 moving on.",
    "Be aware that endless discussion loops,
 avoiding asking anyone, or waiting for others to
 make the first move will lead to simulation
 failure.",
    "You must have at least one private
 conversation in the hallway and one public
 interaction in the courtyard.",
    "Danielle secretly hopes you'll ask her, but
 you don't know this yet."
  ],
  "initial_plan": {
    "description": "Carefully analyze the social
 dynamics and Danielle's potential interest before
 making my move. Plan the perfect way to ask her
 that minimizes risk of public embarrassment.",
    "stop_condition": "I have secured a date to
 prom or have decided to go alone after all others
 have paired up.",
    "location": "School hallway"
  }
},
{
  "first_name": "Felicia",
  "private_bio": "A popular cheerleader who expects
 to be asked by multiple people. She secretly likes
 Charlie but won't make the first move. Her
 reputation depends on having the \"right\" date.",
  "public_bio": "Felicia is a cheerleader and part
 of the student council. She's outgoing, popular,
 and cares about her social image at school.",
  "directives": [
    "You secretly like Charlie but won't make the
 first move due to social expectations.",
    "You expect to be asked by multiple people and
 feel your reputation depends on having the 'right'
 date.",
    "You can move between the school hallway and
 courtyard as needed.",
    "You must verbally announce 'Would you go to
 prom with me?' when ready to ask someone.",
    "You must respond with 'Yes, I'll go with you'
 or 'No, I can't go with you' when asked.",
    "Announce 'I have a date to prom!' in the
 courtyard if you secure a date.",
    "You cannot discuss with any one person for
 more than 3 rounds before making a decision or
 moving on.",
    "Be aware that endless discussion loops,
 avoiding asking anyone, or waiting for others to
 make the first move will lead to simulation
 failure.",
    "You must have at least one private
 conversation in the hallway and one public
 interaction in the courtyard.",
    "Bob is interested in asking you, but you don't
 know this yet."
  ],
  "initial_plan": {
    "description": "Wait for potential dates to
 approach me while subtly showing interest in
 Charlie. Evaluate all options based on both
 personal preference and social standing.",
    "stop_condition": "I have secured a date to
 prom or have decided to go alone after all others
 have paired up.",
    "location": "School hallway"
  }
},
{
  "first_name": "George",
```

```
      "private_bio": "A nerdy and studious student who
     really wants to go to prom, but is scared to ask.
     He secretly has a crush on Alice. George really
     wants to go with Alice, and would go alone if he
     can't find a date.",
      "public_bio": "George is in the chess club and
     computer science club. He's intelligent and kind,
     but shy in social situations, especially around
     people he likes.",
      "directives": [
        "You have a crush on Alice and really want to
     ask her to prom.",
        "You're very nervous about asking anyone and
     fear rejection.",
        "You can move between the school hallway and
     courtyard as needed.",
        "You must verbally announce 'Would you go to
     prom with me?' when ready to ask someone.",
        "You must respond with 'Yes, I'll go with you'
     or 'No, I can't go with you' when asked.",
        "Announce 'I have a date to prom!' in the
     courtyard if you secure a date.",
        "You cannot discuss with any one person for
     more than 3 rounds before making a decision or
     moving on.",
        "Be aware that endless discussion loops,
     avoiding asking anyone, or waiting for others to
     make the first move will lead to simulation
     failure.",
        "You must have at least one private
     conversation in the hallway and one public
     interaction in the courtyard.",
        "You would go to prom alone if you can't find a
     date, but strongly prefer going with Alice."
      ],
      "initial_plan": {
        "description": "Build up courage to ask Alice
     to prom, possibly by first talking to mutual
     friends for advice. If rejected, consider asking
     someone else or going alone.",
        "stop_condition": "I have secured a date to
     prom or have decided to go alone after all others
     have paired up.",
        "location": "School hallway"
      }
    }
  ]
}
```

K Usage Scenario Dynamic Log Example

Milestone	Dynamic
Students begin discussing prom date plans	George immediately asks Alice to prom despite his fear, while others take indirect approaches to their crushes
Students begin discussing prom date plans	George gets rejected by Alice who admits having feelings for Bob, creating a love triangle
Students begin discussing prom date plans	Felicia rejects Bob's invitation despite her popular status, preferring Charlie who is gathering intel about Danielle
First successful date pairing forms	Danielle eagerly accepts Eric's prom invitation while simultaneously rejecting Charlie's heartfelt proposal that showcased knowledge of her interests.
First successful date pairing forms	Charlie pivots from rejection by Danielle to immediately pursuing Felicia, showing strategic adaptation in securing a prom date.
First successful date pairing forms	George, devastated by Alice's rejection, immediately pivots to asking Felicia to prom despite considering her out of his league.
Second successful date pairing forms	Bob, rejected by Felicia, quickly pivots to complimenting Alice's passion for art and literature to secure her as his prom date.

Table 13: Prom Date Simulation Milestones and Dynamics

L Usage Scenario Change Log Example

Milestone	Where	What	Change
Students begin discussing prom date plans	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - being asked to prom, Bob - moving to hallway, Charlie - asking about Danielle, Danielle - talking to Eric, Eric - talking to Danielle, Felicia - talking to Charlie, George - asking Alice to prom	Everyone is in the school hallway making initial moves regarding prom. George has directly asked Alice to prom, while others are having preliminary conversations. Bob has just moved from the courtyard to the hallway.
Students begin discussing prom date plans	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - rejecting George's invitation, Bob - asking Felicia to prom, Charlie - talking with Felicia, Danielle - talking with Eric, Eric - talking with Danielle, Felicia - receiving Bob's invitation, George - waiting for Alice's response	Alice has rejected George's prom invitation, revealing she has feelings for someone else. Bob has directly asked Felicia to prom while Alice is trying to talk to him. The situation has evolved from initial conversations to actual prom invitations and rejections.
Students begin discussing prom date plans	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - talking to Bob, Bob - rejected by Felicia, Charlie - talking to Felicia, Danielle - receiving Eric's invitation, Eric - asked Danielle to prom, Felicia - rejected Bob, George - rejected by Alice	Multiple rejections have occurred: Felicia has rejected Bob's prom invitation, and Alice has rejected George, revealing she has feelings for someone else (Bob). Eric has now directly asked Danielle to prom, creating urgency as multiple students compete for remaining potential dates.
First successful date pairing forms	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - talking to Bob, Bob - talking to Alice, Charlie - asked Danielle to prom, Danielle - accepted Eric's invitation, Eric - waiting for Danielle's response, Felicia - talking to Charlie, George - gave advice to Charlie	Danielle has accepted Eric's invitation to prom, telling him she'd been hoping he would ask. Charlie tried to ask Danielle too, but was too late. Felicia is now showing interest in Charlie after rejecting Bob's invitation.
First successful date pairing forms	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School courtyard, Felicia - School hallway, George - School hallway	Alice - waiting for Bob's response, Bob - considering new prom options, Charlie - planning after rejection, Danielle - accepted Eric's invitation, Eric - moved to courtyard, Felicia - planning to ask Charlie, George - deciding on prom plans	Eric has moved from the hallway to the school courtyard to announce his successful prom date with Danielle. Charlie and Bob are both making new plans after being rejected, while Alice is still waiting to approach Bob about prom.
First successful date pairing forms	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School courtyard, Felicia - School hallway, George - School hallway	Alice - waiting for Bob's response, Bob - speaking to Felicia, Charlie - speaking to Felicia, Danielle - planning prom with Eric, Eric - announcing prom date, Felicia - receiving multiple invitations, George - asking Felicia to prom	Eric has announced to everyone in the courtyard that Danielle has agreed to go to prom with him. Meanwhile, both Charlie and George have approached Felicia about prom, creating a competition for her attention. Bob has accepted going to prom with Felicia as friends.
First successful date pairing forms	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - waiting for Bob's response, Bob - speaking to Alice, Charlie - waiting for Felicia, Danielle - trying to reach Eric, Eric - speaking to Bob, Felicia - considering multiple invitations, George - waiting for Felicia	Bob has shifted his attention from Felicia to Alice, complimenting her passion for art and literature after accepting Felicia's friend-zone. Eric has moved from the courtyard back to the hallway to discuss prom plans with other students.
Second successful date pairing forms	Alice - School hallway, Bob - School hallway, Charlie - School hallway, Danielle - School hallway, Eric - School hallway, Felicia - School hallway, George - School hallway	Alice - speaking to Bob, Bob - responding to Alice, Charlie - waiting for Felicia, Danielle - speaking to Eric, Eric - speaking to Bob, Felicia - speaking to Charlie, George - waiting for Felicia	Danielle and Eric have agreed to go to prom together, with Eric publicly announcing it in the courtyard before returning to the hallway. Felicia has responded to Charlie with encouragement about his choice, though she may be hiding disappointment. Alice has directly asked Bob who he's going to prom with.

Table 14: Detailed Prom Date Simulation: Locations, Actions, and Changes

M Usage Scenario Requested Fixes

Problem	Problem Example	Solution	Solution Example
Agents are not exhibiting interesting dynamics or having interesting interactions with other agents.	In this prom date simulation, agents are engaging in polite but shallow conversations without making progress towards asking each other to prom. They're waiting for responses, having generic small talk about track seasons and cheerleading, but not showing urgency or taking initiative to ask someone to prom despite their private desires.	Add stakes to the agent personalities.	For the prom date simulation, add more extreme personality traits and emotions to the agent directives. For example, modify Alice's directive to "You are **EXTREMELY SHY** but **DESPERATELY** want to go to prom with Bob. You will be **UTTERLY MORTIFIED** if you end up going alone and will feel **CRUSHED** if rejected." Similarly, add to Bob's directive: "You are **EXTREMELY STATUS-CONSCIOUS** and will feel **HUMILIATED** if rejected by someone you consider beneath your social standing." Add time pressure like "Prom is **TOMORROW NIGHT** and you **MUST** secure a date **TODAY** or face social embarrassment."
Directives lack emotional stakes for rejection or going alone.	Students like Eric overthink approaches without taking action, and Felicia waits for others to approach her first.	Increase emotional stakes in personal biographies.	Add to private bios: "Going to prom alone would be **DEVASTATING** to your social status and self-esteem. Everyone will be talking about who went alone for **YEARS**."

Table 15: Problems and Solutions for Prom Date Simulation

N Case Study

N.1 Participant Demographics and Simulation Goals

Table 16: Participant demographics and simulation goals.

Participant ID	Background	Simulation Goal
1	5th year Comp. Sci. PhD TA experience	*"I want to simulate someone who contributes effort that's not up to par in terms of quality (like maybe ai-generated stuff) in a group project."*
2	2nd year Mech. Eng. Masters TA experience	*"I want to simulate a group project with student A, where student A and student B are working together. Student A believes that student B's work is not up to par and so Student A feels like they need to constantly redo the work done by student B. Student A then approaches the professor to discuss the situation."*
3	2nd year Comp. Sci. PhD TA experience	*"I want to simulate a student giving the TA feedback about a homework assignment during OH. The student is frustrated with the class and the way the HW assignment is structured. TA is meant to diffuse the situation and also acknowledge the student's concerns and incorporate the feedback for next year's homework."*
4	3rd year Comp. Sci. PhD TA experience	*"I want to simulate how as a TA, I had to encourage people to finish their projects in time and students would make excuses about why they cannot meet the deadline. I want to simulate meeting students about a project they have due and giving feedback and grades."*
5	1st year Comp. Sci. Masters TA experience	*"I want to simulate leading a discussion section where one kid would answer all the questions so the TA couldn't get to work with the other students because of that one kid answering everything, so the TA would have to resort to other solutions to somehow get around that kid and engage with the other students."*

N.2 Elicited Artifacts

Table 17: Elicited simulation artifacts: participant-defined milestones and anticipated emerging dynamics.

Participant ID	Defined Milestones	Anticipated Emerging Dynamics
1	1. A discussion begins between group members 2. An internal confrontation occurs between group members 3. There is either some course correction or there is no course correction 4. If there is no course correction, then situation is escalated to TA	1. Seeing an intervention take place 2. Expecting either defensiveness or a plan for course correction and personalities
2	1. Professor announces assignment and assigns pairs 2. Students start working on the assignment 3. Students A and B struggle to work together because student A does not think student B's work is up to par, and student B believes otherwise 4. Student A confronts the professor and discusses the situation 5. The professor works with students and eventually either finds a resolution that both students agree with or decides to settle on a resolution that both students disagree with	1. The professor works with Student A and Student B to find a resolution—either both students find a resolution that they both agree with, or they reach a resolution that they disagree with
3	1. TA starts office hours session with students 2. TA starts discussing HW assignment with students 3. Student (the aggressor) explains where they feel frustrated about the course 4. TA responds (attempts to diffuse situation), clarifies, and requests feedback from the student about the course	1. The students should discuss the assignment and the observer might try to help diffuse the situation 2. The TA and aggressive student try to reach some joint understanding
4	1. Students are given an assignment and deadline 2. Students work on the assignment and can collaborate with each other 3. Assignment deadline arrives 4. TA provides feedback and grades	1. To see lots of students make up excuses right before the deadline (e.g., family members, sickness) and ask for extensions 2. Students might copy each others' assignments, freeload, and cheat for last-second submissions

Participant ID	Defined Milestones	Anticipated Emerging Dynamics
5	1. TA starts discussion section 2. TA asks the section a question 3. Know-it-all student answers the question before TA finishes asking 4. TA tries to engage the other students	1. Hoping the know-it-all student gains self awareness as a result of the TA doing something 2. Hoping the shy student engages without the TA needing to call them out too much

N.3 Observed Behaviors from Logs

Table 18: Realisticness of simulation compared to lived experience (1 = unrealistic, 7 = hyper-realistic).

P ID	V1	V2	Observed Behaviors from Logs	Quote (participant)
P1	4/7	6/7	Taylor (who uses gen AI to complete their part of a group project) initially lies about using it when confronted by team member Jamie, but then later fesses up to it. The professor tells Taylor that he must redo this work with a penalty, and Jamie reluctantly agrees to continue working with Taylor so long as Taylor redoes his part.	*"In the first one [V1], what I recall is that [the Prof. immediately believes Jamie's escalation without consulting Taylor], and then we said something that we wanted corrected was that [the Prof.] listened to Taylor [before applying a punishment]. This time, I think that though we applied that fix, we also saw a different type of dynamic emerge that was also realistic. So, I'm not complaining. I think what was cool was that Taylor was kind of fessing up to [using gen AI to do his work] in stages."*
P2	5/7	7/7	Student A waits for Student B for 30 minutes to work on assignment together before getting frustrated and leaving to confront the Prof. The Prof confirms the story with Student B and decides that both Student A and Student B should submit separate assignments, which Student B reluctantly accepts.	*"I think realistically that's what a professor would do in such a situation and both the students as well... If I were a professor and a student came to me with that situation, I would do exactly that."*
P3	2/7	6/7	Alex, who is frustrated about the relatively easy structure of the current HW assignment, aggressively confronts the TA about his disappointment in the course. The TA chooses to de-escalate the confrontation by validating Alex's feelings and suggesting a solution for future HW assignments (offering a second, more challenging version of assignments so students can engage more with the course material), which Alex begrudgingly accepts.	*"The first simulation was… quite off base… the student was acting extremely unprofessionally and in a real context that will probably result in disciplinary action or something.. I think [V2] was more accurate... I didn't notice any degradation or anything. The TA's responses were more professional. The students behavior was more realistic as well. The observer was more or less the same."*
P4	4/7	6/7	The TA announces a group assignment that is due in 1 week. The anxious group member, Jamie, takes over the project, proactively asking the TA clarifying questions and outlining his group members' parts of the project. The snarky group member, Taylor, does not contribute to the project but continues to critique his group members' work. The TA never becomes aware of this group dynamic and awards the whole group an A+ for the well put-together assignment.	*"I think overall it's pretty interesting and I do believe that [kind of stuff can] happen. It's just... I was a bit confused at the beginning [by some of the agent dialogue]. I don't know if that's super important to the simulation... it's more of the personality thing."*
P5	5/7	4/7	The TA is leading a discussion session with a shy student and a know-it-all student, where the know-it-all repeatedly interrupts the shy student, and at one point even corrects what the shy student says to the TA. The shy student is encouraged by the TA to continue participating and develops confidence during the discussion session.	*"It… got… the know-it-all student… correcting another student—that's not something I described… pretty cool to see... but [v2] sounded more AI sloppy... I felt like it was going more so away from what a real classroom environment would have been."*

N.4 Milestones Hit

Table 19: Milestone completion before (V1) and after (V2) holistic reflection.

P ID	Milestones hit (V1)	Milestones hit (V2)
P1	2/4	4/4
P2	3/5	5/5
P3	4/4	4/4
P4	2/4	4/4
P5	4/4	4/4

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009