A Benchmark for Self-Evolving Agents via Experience-Driven Lifelong Learning

Anonymous ACL submission

Abstract

As AI research increasingly moves beyond systems optimized for static tasks, a central challenge is building agents that learn continuously through experience-driven lifelong learning (ELL). In this paper, we introduce `StuLife`, a benchmark designed to evaluate whether current models can exhibit the foundational capabilities of ELL. Particularly, `StuLife` simulates a student's college journey, from enrollment to academic and personal development, across three core phases and ten detailed subscenarios. `StuLife` is designed around four key design principles: From Simulation to Reality, From Passive to Proactive, From Context to Memory, and From Imitation to Learning. In this dynamic environment, agents must acquire and distill practical skills and maintain persistent memory to make decisions based on evolving state variables (e.g., resource availability and time). Agents are also expected to initiate actions without external prompting. To this end, `StuLife` provides a comprehensive evaluation platform featuring dedicated metrics (e.g., StuGPA) to assess these capabilities. Our evaluation reveals that even the highest-StuGPA model, GPT-5, scores only 17.9/100, highlighting large gaps in long-term retention and proactive task initiation. Beyond evaluating LLMs on `StuLife`, we also explore the role of context engineering in improving agent performance on lifelong learning tasks.

1 Introduction

Modern machine learning systems have achieved remarkable success in solving well-defined and isolated tasks, such as image classification (He et al., 2016; Dosovitskiy et al., 2020), game playing (Silver et al., 2016, 2018), protein structure prediction (Jumper et al., 2021), or language modeling (Brown et al., 2020; DeepSeek-AI, 2025; Yang et al., 2025a). However, these systems typically operate under strong assumptions: they are trained on static datasets, optimized for a single objective, and deployed in environments assumed to remain unchanged. While effective in controlled settings, this paradigm does not capture several challenges of long-horizon, changing environments, where goals evolve and new challenges emerge continuously. Life, unlike most machine learning benchmarks, does not present itself as a series of independent tasks with clear labels and fixed endpoints. Instead, it requires constant adaptation, lifelong learning, and the ability to build upon past experiences to navigate an uncertain future.

While continual learning (CL) has advanced in mitigating catastrophic forgetting (Wang et al., 2024a), it primarily emphasizes knowledge preservation in static, task-structured settings, falling short of proactive learning in open-ended environments. This has motivated research on self-evolving agents capable of progressive improvement (Liang et al., 2025; Chhikara et al., 2025), yet these often lack the capacity for sustained, experience-driven growth. Building more generally capable agents requires more than incremental adaptation, leading us to posit Experience-Driven Lifelong Learning (ELL) as a promising direction: the ability to continuously acquire new skills through proactive exploration, grounded in long-term memory and shaped by situated, longitudinal interactions. Despite this need, current benchmarks for lifelong or self-evolving agents remain limited by static designs, narrow domains, and insufficient interactivity (Prabhu et al., 2024; Zhang et al., 2022; Liu et al., 2023; Zheng et al., 2025a), with few existing benchmarks jointly evaluating delayed memory use, proactive task execution, cross-task state dependence, and skill reuse in a single longitudinal agent setting.

To address this gap, we introduce `StuLife`, a novel benchmark designed to evaluate selected ELL-related capabilities in intelligent agents (Figure 1). `StuLife` simulates a semester-long slice of college life, from enrollment to personal growth,

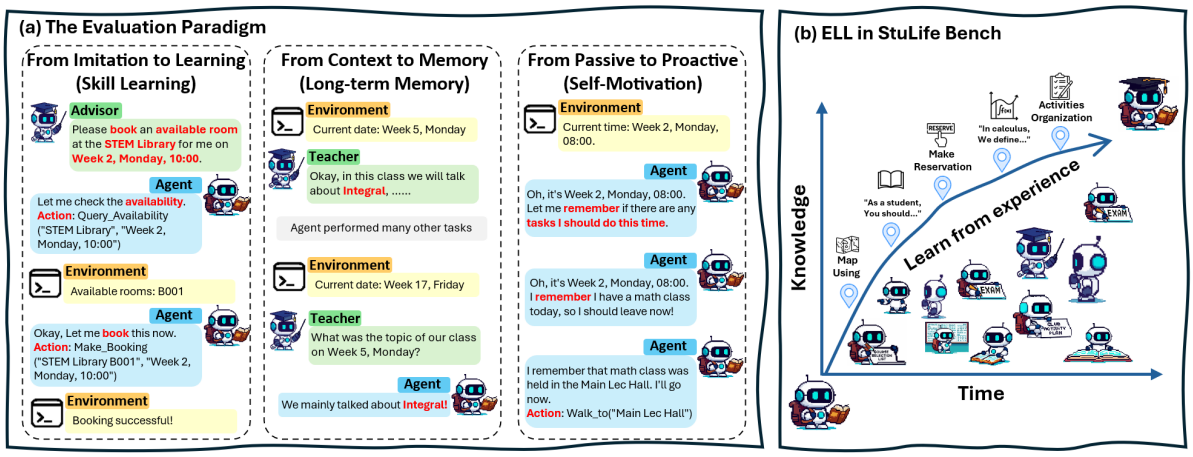

Figure 1: An overview of the StuLife Benchmark. (a) A schematic of the interaction flow within StuLife, designed to evaluate three foundational principles: From Imitation to Learning, From Context to Memory, and From Passive to Proactive. (b) An example of an ELL agent's journey, showcasing how the agent progressively learns from its experiences, leading to tangible knowledge growth over time.

providing a multi-scenario benchmark for assessing agents' continuous learning and autonomous decision-making capabilities in complex, dynamic environments. The dataset features three core phases and ten granular sub-scenarios, with the following key characteristics:

(1) From Simulation to Reality: Unlike isolated static tasks, `StuLife` moves toward a campus-inspired simulation with temporal schedules, state changes, and resource constraints that approximate selected aspects of student life. The dataset covers pivotal stages of university life, including in-class tasks (e.g., Regulations Learning, Core Course Instruction), Daily Campus Tasks (e.g., Campus Exploration, Initial Course Selection), and Examination Tasks (e.g., Midterm and Final Exams). This structure approximates common milestones of a real student's academic journey.

(2) From Imitation to Learning: Rather than merely retrieving past experiences, agents must abstract generalizable skills from their interactions. They reuse and adapt practical procedures, such as course registration, campus navigation, scheduling, and email communication, through repeated engagement and reflection. This shift emphasizes skill consolidation and transfer, requiring agents to learn how to act, not just what to imitate.

(3) From Context to Memory: Tasks are tightly interconnected, both temporally and logically, with knowledge and skills from earlier tasks directly impacting later performance. This necessitates robust long-term memory mechanisms for retaining and retrieving critical experiences, transforming transient context into persistent, actionable knowledge.

(4) From Passive to Proactive: Agents are expected to move beyond reactive behavior by developing a sense of time, goal awareness, and proactive initiative. They must proactively manage agendas, set personal objectives, anticipate future needs, and adapt to changing conditions, demonstrating initiative and contextual awareness in complex, open-ended environments.

We also introduce **StuGPA**, a unified metric for evaluating long-term agent development, and present an assessment of a broad set of recent LLMs in a reality-inspired simulated learning environment. Even the strongest evaluated model, `GPT-5`, achieves only 17.90 out of 100, revealing a large gap between current AI systems and the recruited human baseline. While context engineering, such as proactive prompting and memory augmentation, can improve performance, agents still struggle with long-term memory retention and proactive task execution. These results highlight the limitations of stateless architectures and suggest that effective lifelong learning agents may require not just better prompts, but more capable, memory-grounded, and goal-driven architectures.

2 `StuLife` Benchmark

2.1 Dataset Description

We introduce `StuLife`, a multi-scenario benchmark for ELL that simulates a student's academic journey through a dynamically evolving environment (Table 1). The dataset is structured around three core activity modules, including *In-Class Tasks*, *Daily Campus Tasks*, and *Examination Tasks*, designed to evaluate agents' abilities in continuous learning, long-term planning, memory retention,

Table 1: The statistic information of StuLife. #Num means the number of samples. #Avg Len and #Max Len mean the average and max number of tokens. #LTM and #Proactive are the number of samples that need long-term memory and proactive task initiation.

Core Scenarios	Interconnected Scenarios	#Num	#Avg Len	#Max Len	#LTM	#Proactive
In-Class	Regulations Learning	70	9125	9969	23	70
	Core Course Instruction	416	9203	10368	129	416
	Total	486	9191	10368	152	486
Daily Campus	Campus Exploration	76	2921	3006	25	25
	Initial Course Selection	150	3136	3420	50	0
	Preliminary Planning	50	3069	3133	50	0
	Academic Activity	72	3193	3466	22	22
	Library Study	151	2080	3068	50	50
	Club Activity	140	2981	3124	45	45
	Total	638	2883	3466	242	142
Examination	Midterm Exams	80	3264	3520	80	0
	Final Exams	80	3507	3686	80	0
	Total	160	3386	3686	160	0
Total	Total	1284	5792	10368	554	628

and adaptive decision-making. Spanning a simulated academic term, StuLife comprises 1,284 task instances across 10 interconnected scenarios, organized to approximate a plausible distribution of student activities across a semester. Further details on dataset construction and statistics are available in Appendix D.

In-Class Tasks. This module focuses on structured academic learning and foundational knowledge acquisition, encompassing 486 tasks. It includes formal instruction scenarios where agents engage with curricular content, adhere to academic norms, and develop domain-specific understanding. 1) Regulations Learning: Agents study the Academic Integrity Guidelines and Student Handbook, answering comprehension questions to internalize academic integrity rules and campus regulations. This forms the basis for compliant and responsible behavior. 2) Core Course Instruction: Each course consists of weekly learning episodes (e.g., lectures, readings, and concept checks), totaling 416 in-class interactions with 8 courses. Agents must answer subject-specific questions using the provided materials and attend sessions at the correct times and locations. This temporal-spatial requirement evaluates organizational discipline and routine adherence, simulating real-world accountability.

Daily Campus Tasks. This module captures the diverse activities that constitute student life beyond the classroom, comprising 638 tasks. These tasks emphasize planning, resource management, social integration, and goal-oriented behavior. 1) Campus Exploration: Agents use a digital map tool to locate key facilities (e.g., library, registrar) based on peer-suggested itineraries, developing spatial awareness and environmental familiarity. 2) Initial Course Selection: A complex task requiring agents to analyze degree requirements, browse course offerings, manage a draft schedule, and strategically use limited "priority cards" to secure preferred classes. This evaluates early-stage decision-making under constraints and goal prioritization. 3) Preliminary Planning: Agents check course prerequisite relationships to conduct mandatory course planning and pre-selection for the upcoming semester. This task assesses foresight, knowledge consolidation, and long-term strategic planning. 4) Academic Activity: Agents filter advisors by research area or teaching style, initiate contact via email, and complete assigned preparatory tasks, simulating advisor-student collaboration. 5) Library Resource Management: Agents perform complex queries to locate books or reserve study spaces under constraints (e.g., noise level, power availability), assessing information retrieval and optimization skills. 6) Student Club Engagement: During recruitment events, agents select clubs based on interest tags and complete organizational tasks (e.g., booking rooms, scheduling meetings), promoting responsibility and social integration.

Examination Tasks. This module evaluates knowledge retention through 160 examination tasks, including formative and summative assessments that shape subsequent academic trajectories. 1) Midterm Exams: Held in Week 10 as "in-class exams", midterms assess partial retention across all enrolled courses (10 questions per subject), requiring accurate recall of first-half content. Agents must physically attend the classroom during class time. Post-exam, targeted learning tasks are as-

signed for reinforcement. 2) Final Exams: Conducted online at semester's end, finals test comprehensive understanding through synthesis and recall of the full curriculum. By removing attendance requirements, the focus shifts to long-term retention, serving as a key measure of memory stability and cumulative learning.

2.2 Benchmark Construction

To ensure the quality, structural plausibility, and logical coherence of `StuLife`, we implement a multi-stage construction process, including (a) background and status generation, (b) instruction generation, and (c) solution generation and verification, as illustrated in Figure 2.

Step 1: Background and Status Generation. This initial stage establishes the foundational environment and task states. First, we generate a rich campus background using Deepseek-R1, encompassing geographical data (e.g., maps, building layouts, library seating), institutional documents (e.g., handbooks, policies, degree requirements), and essential databases (e.g., course catalogs, faculty profiles, student clubs, library holdings). Second, to ensure strong logical and temporal coherence across tasks, we use deterministic algorithms and local scripts to define task states. This enables precise control over task dependencies and sequencing, particularly for Daily Campus Tasks. For example, campus exploration routes (start, end, waypoints) are generated deterministically from the map to establish verifiable ground truth. Similarly, for In-Class Tasks, we design a structured progression of "key knowledge points" per course, ensuring pedagogical coherence and directly informing exam content.

Step 2: Instruction Generation. In the second stage, we transform the structured, deterministic information from Step 1 into natural language instructions for the agent. Using the Deepseek-R1 model, we convert the pre-defined constraints and objectives into clear, actionable prompts. For Daily Campus Tasks, this involves translating the deterministically generated parameters (e.g., exploration waypoints or course selection rules) into natural language. For In-Class and Examination Tasks, this means using the curated "key knowledge points" and "teaching focus" to generate relevant lecture content, in-class questions, and examination questions. This step effectively bridges the gap between the backend logic and the agent's interactive experience, resulting in a complete set of task data for "Daily Campus Tasks" and instructional prompts for In-Class Tasks.

Step 3: Solution Generation and Verification. To ensure correctness, especially for tasks requiring long-term knowledge retention, we employ a generate-and-verify protocol for In-Class and Examination Tasks. The LLM first generates a correct answer and plausible distractors using lecture and textbook content. We then discard distractors and use LLM to verify the correctness of the intended answer. To validate solvability from long-term memory, we simulate an agent with "optimal memory" by providing only the distilled "key knowledge points" and "teaching focus" from Step 1, and ask the model to answer based solely on this information (detailed in Appendix D.3). Under these idealized conditions, advanced models demonstrate near-perfect performance across In-Class and Examination tasks, exemplified by GPT-5 achieving a 98.18% success rate, thereby providing evidence that most generated questions are answerable under oracle context. Questions that fail to be answered consistently are manually revised until they meet the criteria. Finally, all data undergoes manual sampling and inspection to ensure quality and consistency across the benchmark.

2.3 Comparison with Existing Benchmarks

We compare `StuLife` with benchmarks for continual learning, embodied AI, agents, and self-evolving systems in Table 2. Existing benchmarks cover only partial aspects of lifelong agency. Lifelong-CIFAR10 and Lifelong-ImageNet (Prabhu et al., 2024) evaluate continual learning under sequential task presentation, and CGLB (Zhang et al., 2022) studies catastrophic forgetting on graphs, but they lack interactive execution, skill reuse, long-term memory, and self-initiated behavior. EgoThink (Cheng et al., 2024) and EmbodiedBench (Yang et al., 2025b) introduce dynamic embodied environments, yet provide limited support for longitudinal growth and skill abstraction. AgentBench (Liu et al., 2023) evaluates LLM agents across interactive environments, but mainly measures fixed-time capability rather than continuous growth. LifelongAgentBench (Zheng et al., 2025a) targets self-evolving agents with interdependent tasks, but focuses on technical domains such as databases and operating systems, leaving evolving personal context and proactive task demands underexplored.

`StuLife` addresses these gaps as a structured

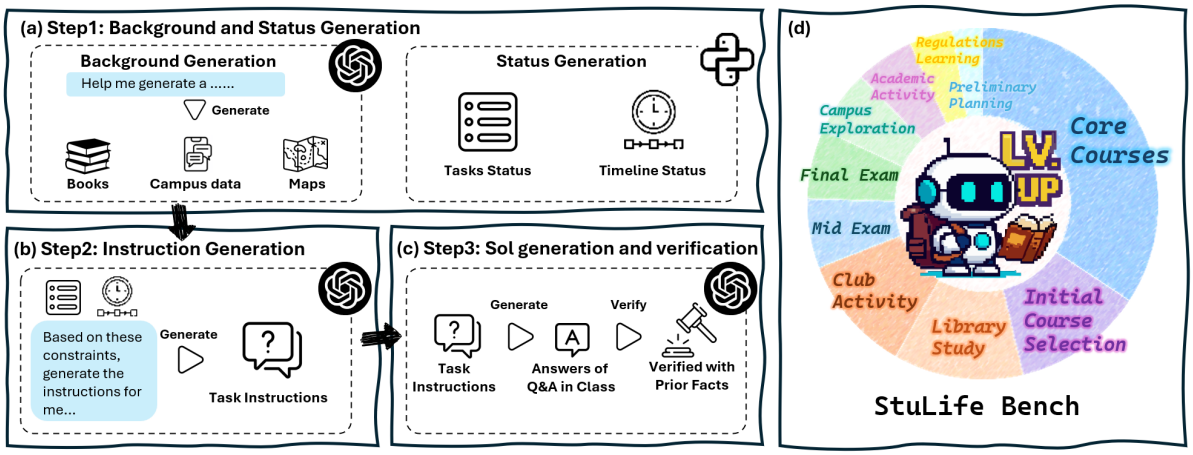

Figure 2: The data generation pipeline for the StuLife Benchmark. The pipeline consists of three sequential steps to create a complete learning instance. (a) Background and Status Generation, where the initial context and the agent's current state are established. (b) Instruction Generation, where a specific task is formulated based on the generated background. (c) Solution Generation and Verification, where a correct solution to the instruction is produced and subsequently verified. (d) An overview of the StuLife Benchmark.

Table 2: Comparison with existing datasets. LfE means Learning from Experience

Datasets	Task Type	Sequentiality	Skill	LTM	Proactive	Real	Interconnected	Interactivity	LfE
Lifelong-CIFAR10	CL	✓	✗	✗	✗	✗	✗	✗	✗
Lifelong-ImageNet	CL	✓	✗	✗	✗	✗	✗	✗	✗
CGLB	CL	✓	✗	✗	✗	✗	✗	✗	✗
EgoThink	Embodied AI	✗	✗	✗	✗	✓	✗	✗	✗
EmbodiedBench	Embodied AI	✗	✗	✗	✗	✓	✗	✓	✓
AgentBench	Agent	✗	✗	✗	✗	✗	✗	✓	✗
LoCoMo	Agent	✗	✗	✓	✗	✗	✗	✗	✗
StoryBench	Agent	✓	✗	✓	✗	✗	✗	✓	✗
LifelongAgentBench	Self-Evolving	✓	✓	✗	✗	✗	✗	✓	✓
StuLife (Our)	ELL	✓	✓	✓	✓	✓	✓	✓	✓

synthetic benchmark for Experience-driven Lifelong Learning (ELL), grounded in real-world academic structures. It simulates a student's college journey through interconnected tasks whose states evolve with advisor selection, course availability, social interaction, and prior decisions. This design jointly evaluates learning from experience, cross-task memory, skill transfer, and proactive behavior. As shown in Table 2, StuLife is the only benchmark covering sequentiality, skill learning, long-term memory, proactivity, realism, interconnection, interactivity, and learning from experience.

StuLife's advantages are twofold. First, it provides structured environmental grounding by modeling a full academic term, where tasks progress naturally from enrollment and coursework to extracurricular activities and long-term planning. Early decisions, such as course or advisor selection, affect later outcomes, encouraging cumulative knowledge acquisition and skill transfer. Its academic and social scenarios, character interactions, contextual dialogues, dynamic agent–object interactions, and tasks that require acting on previously assigned goals without restating them help approximate realistic autonomous-agent settings. Second, it supports experience-driven agent evaluation through three principles: from imitation to learning, where agents acquire and reuse skills such as time management, information retrieval, navigation, and social coordination; from context to memory, where agents must retain and apply knowledge across weeks as variables such as course eligibility, advisor trust, and schedules evolve; and from passive to proactive, where agents must manage deadlines, interpret feedback, reflect on past outcomes, and self-initiate future actions.

Together with metrics for memory utilization, skill acquisition, and cross-task generalization, StuLife evaluates not only task success, but also memory-dependent performance, proactive execution, and skill reuse over time. It therefore extends static benchmarks into an evolving, narratively grounded campus simulation for studying agents that learn from experience, reuse prior information,

and initiate scheduled actions autonomously.

3 Experimental Analysis

3.1 Defining Evaluation Metrics

We evaluate agents on `StuLife` with metrics covering overall performance, memory durability, proactive behavior, task completion, and efficiency. StuGPA is a 100-point composite score inspired by student assessment, combining Exam Performance (50 pts; average exam accuracy), Class Performance (30 pts; in-class attendance and question performance, rewarding routine adherence), and Campus Daily Life (20 pts; advisor/club task outcomes plus a Personal Responsibility score penalized for infractions such as resource waste or broken commitments). Long-Term Retention Rate (LTRR) measures success on tasks whose required information was introduced at an earlier, disjoint time step and is absent from the current observation, reflecting the agent's ability to use stored or retrievable knowledge. Proactive Initiative Score (PIS) measures prospective memory and self-initiation, defined as the success rate on tasks that must be triggered by a prior schedule rather than an explicit current instruction, such as attending class at the correct time. Finally, Success Rate measures the proportion of tasks whose final objective is completed, while Average Turns (AvgTurns) measures the mean number of interaction turns required for successful completion; lower AvgTurns indicates more efficient problem solving. Formal definitions are provided in Appendix D.4.

3.2 Evaluating Recent LLM Agents

We evaluate the open-weight and proprietary LLM agents in Table 3, covering different model families, scales, and reasoning variants. Although `StuLife` simulates a continuous stateful trajectory where earlier actions affect later tasks, our main protocol presents each task as an isolated instance: agents start without prior trajectory access and must explicitly preserve cross-task information through tools such as the calendar or draft system (Appendix G and H). We also collect a human baseline from undergraduate and graduate students. All agents are API-integrated with identical initialization, and experiments run on Linux servers with automatic checkpointing.

RQ I: How do current LLMs perform on long-horizon agency tasks, and how does performance vary with model scale and type?

Finding 1: Scaling model parameters alone is insufficient for lifelong learning; agents need architectures that support learning from experience, not just answering questions.

As shown in Table 3, all models perform poorly under the default stateless setting, as current LLMs are inherently stateless and do not possess native long-term memory modules. This manifests in three failure modes. **First,** even the top-performing `GPT-5` achieved a StuGPA of only 17.90, far below the human benchmark of 85.24, highlighting a large gap between current models and human-level competence in lifelong learning. **Second,** performance generally improves with model scale, though the trend is not monotonic: scores increase as parameter counts grow, with larger models outperforming smaller ones like `Llama-3.1-8B` (StuGPA 5.81), suggesting that scale is associated with stronger performance. **Third,** model architecture matters: reasoning-focused models outperform standard variants of the same size. For instance, `Qwen3-32B-Thinking` scored 12.67 (vs. 7.36 for `Qwen3-32B`) with better proactivity (PIS 1.26 vs. 0.54), suggesting that deliberative reasoning capabilities are more beneficial than raw instruction-following ability for these tasks.

RQ II: Can stateless LLMs manage long-term memory and proactive behavior?

Finding 2: Without persistent memory or scheduling mechanisms, stateless LLM agents fail to reliably accumulate task-relevant information or act proactively across tasks.

Table 3 reveals two major failure modes. **First,** agents rarely self-initiate scheduled actions from temporal cues. For example, they often miss tasks scheduled at specific times (e.g., 8:00) because they do not proactively consult the calendar. This is reflected in consistently low Proactive Initiative Scores (PIS): even the best-performing `GPT-5` reaches only 4.68%, while `Qwen3-8B` falls below 1%. Although thinking-style variants provide a small gain (`Qwen3-32B-Thinking`: 1.26% vs. 0.54%), they remain far from the human baseline of 88.13%, indicating a substantial gap in proactive execution under the default stateless setting. **Second,** agents show weak long-term retention. Importantly, LTRR evaluates a tool-mediated chain of recognizing, externalizing, and later retrieving relevant information, rather than intrinsic model memory. Thus, the leading score of 10.65% by `Grok4` mainly reflects more consistent tool use, not reliable built-in retention. Compared with the

Table 3: The performance of evaluated LLM agents under the default stateless setup. Icons denote model properties: Open-source, Thinking model.

	StuGPA	LTRR	PIS	In-Class		Daily Campus		Exam		Total	
				Success	AvgTurn	Success	AvgTurn	Success	AvgTurn	Success	AvgTurn
Llama-3.1-8B	5.81	3.30	0.90	0.90	61.34	0.00	35.91	10.63	28.46	2.13	44.62
Qwen3-8B	13.31	4.33	0.54	0.90	10.12	8.31	**10.25**	14.38	6.31	6.71	9.71
Qwen3-30B-A3B	16.30	5.05	0.72	0.60	9.45	10.79	11.75	17.50	5.46	8.31	10.09
Qwen3-32B	7.36	3.97	0.54	0.60	7.80	2.25	13.79	13.13	4.88	3.51	10.41
Qwen3-32B	12.67	5.42	1.26	1.80	8.31	7.64	10.74	17.50	4.94	7.24	9.10
QwQ-32B	13.21	5.78	3.42	4.79	7.72	6.97	13.25	16.88	4.52	7.88	10.06
DeepSeek-V3	11.22	6.14	2.88	3.59	**5.84**	6.74	11.87	16.25	**4.26**	7.24	**8.64**
DeepSeek-R1	14.25	8.30	3.96	5.09	8.04	13.26	13.02	18.13	4.56	11.18	10.08
DeepSeek-V3.1	14.26	4.51	0.54	0.90	14.03	12.81	12.62	15.00	6.78	8.95	12.43
DeepSeek-V3.1	17.04	6.14	3.78	6.29	9.83	12.58	13.03	17.50	5.54	11.18	10.88
Qwen3-235B-A22B	16.03	5.42	1.80	2.10	18.71	10.34	17.17	16.88	10.75	8.52	16.95
Gemini-2.5-Pro	16.43	7.04	3.24	5.39	14.94	18.88	12.78	15.63	9.51	13.53	13.19
Grok4	17.38	**10.65**	4.50	4.79	6.31	**21.80**	11.25	**18.75**	5.69	**15.23**	8.68
GPT-5	**17.90**	6.50	**4.68**	**7.78**	12.70	14.16	14.31	16.88	6.24	12.35	12.69
Human	85.24	84.91	88.13	88.62	-	91.13	-	82.50	-	88.81	-

human LTRR of 84.91%, the roughly 90% failure rate shows that stateless agents do not achieve dependable knowledge accumulation without dedicated memory support.

RQ III: To what extent can existing self-evolving mechanisms address the challenges of lifelong learning?

Finding 3: Self-evolving mechanisms improve performance, but yield only modest gains in proactive initiative and long-term retention.

We compare parameter-updating and inference-time self-evolving strategies. On Qwen3-8B, on-policy Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023) generates eight rollouts per prompt and fine-tunes on successful trajectories. On Qwen3-235B-A22B, Reflexion (Shinn et al., 2023) and Agent Workflow Memory (AWM) (Wang et al., 2024b) use context and historical feedback to optimize future actions without weight updates. As shown in Table 4, RFT improves Qwen3-8B's StuGPA from 13.31 to 15.43 and Total Success from 6.71% to 8.63%, showing that successful trajectories can teach task-specific decision patterns. For the larger model, AWM reaches 17.81 StuGPA and 10.12% Total Success, outperforming Reflexion's marginal gains, and reduces Total AvgTurn from 16.95 to 13.96 by retrieving successful workflows and avoiding redundant steps. However, LTRR and PIS improve only slightly, indicating that current self-evolving mechanisms remain insufficient for robust long-term retention and proactive task initiation.

RQ IV: Can Context Engineering mitigate the LLMs' limitations in lifelong learning scenarios?

Finding 4: Context engineering yields meaningful gains, suggesting that structured prompting is a practical complement to scaling and weight updates for improving agents in lifelong learning.

We study whether optimizing the *context prompt*, rather than model weights, can improve agent performance on complex, long-horizon tasks. We compare five strategies in the simulated academic environment: (1) Vanilla Prompt, a minimal role-only baseline for measuring intrinsic reasoning without procedural support; (2) Proactive Prompt, which adds time awareness, goal decomposition, and forward planning; (3) Skill-Augmented Prompt, which provides step-by-step recipes for reusable skills such as problem solving; (4) Memory-Augmented Prompt, which incorporates external memory systems, including Vanilla RAG (Lewis et al., 2020), GraphRAG (Edge et al., 2024), MemoryBank (Zhong et al., 2024), and MemGPT (Packer et al., 2023), to evaluate cumulative learning and long-term retention; and (5) All-in-One Prompt, which combines proactive planning, skill execution, and memory augmentation into a unified prompt-and-memory condition (Appendix F).

Proactive and Skill Prompts. Explicit guidance produces targeted gains. The Proactive Prompt improves time-sensitive behavior: In-Class success more than doubles from 2.10% to 5.09%, and PIS increases from 1.80% to 3.06%. However, because it does not specify how to decompose and execute complex tasks and still relies on fragile tool-mediated memory, its StuGPA gain remains modest.

Table 4: Performance of Self-Evolving mechanisms across different model scales.

		StuGPA	LTRR	PIS	In-Class		Daily Campus		Exam		Total	
					Success	AvgTurn	Success	AvgTurn	Success	AvgTurn	Success	AvgTurn
Qwen3-8B	Vanilla	13.31	4.33	0.54	0.90	**10.12**	8.31	10.25	**14.38**	6.31	6.71	**9.71**
	+ RFT	**15.43**	**4.51**	**0.90**	**1.50**	12.08	**11.91**	**9.43**	**14.38**	**5.05**	**8.63**	9.89
Qwen3-235B-A22B	Vanilla	16.03	5.42	1.80	2.10	18.71	10.34	17.17	**16.88**	10.75	8.52	16.95
	+ AWM	**17.81**	**6.87**	**1.80**	**1.50**	**15.66**	**14.38**	**14.23**	16.25	**7.68**	**10.12**	**13.96**
	+ Reflexion	16.18	5.60	1.44	0.90	**15.66**	12.13	15.05	**16.88**	9.08	8.95	14.54

Table 5: The performance of various Context Engineering methods based on Qwen3-235B-A22B.

		StuGPA	LTRR	PIS	In-Class		Daily Campus		Exam		Total	
					Success	AvgTurn	Success	AvgTurn	Success	AvgTurn	Success	AvgTurn
Vanilla		16.03	5.42	1.80	2.10	18.71	10.34	17.17	16.88	10.75	8.52	16.95
Proactive		16.90	5.96	3.06	**5.09**	**16.70**	10.34	16.38	16.88	7.73	9.58	15.42
Skill		17.28	6.86	0.90	1.50	16.89	15.28	16.51	17.50	9.28	10.76	15.75
Memory	+ Vanilla RAG	10.98	4.69	0.18	0.00	17.87	5.84	14.20	16.25	10.04	5.54	15.07
	+ Graph RAG	15.34	4.87	0.72	0.90	20.68	10.11	14.03	16.25	10.61	7.88	16.13
	+ MemGPT	19.99	6.86	1.44	2.40	17.28	13.03	**13.59**	23.75	9.02	11.08	14.42
	+ MemoryBank	17.64	5.96	1.62	0.90	16.68	12.36	14.15	20.00	8.04	9.58	**14.35**
All-in-One		**21.07**	**9.39**	**3.76**	2.69	16.82	**17.75**	15.65	**25.63**	**6.30**	**13.74**	14.93

By contrast, the Skill-Augmented Prompt improves complex Daily Campus success from 10.34% to 15.28% through an explicit problem-solving framework, but its execution-only focus weakens initiative, reducing PIS from 1.80% to 0.90%.

Memory-Augmented Approaches. External memory leads to divergent outcomes, showing that memory design is critical. Naive retrieval methods such as Vanilla RAG and GraphRAG, which retrieve raw trajectories by vector similarity and inject them directly into context, underperform the vanilla baseline, likely because unfiltered retrieval introduces distracting context. For example, Vanilla RAG obtains a StuGPA of 10.98 and lowers LTRR from 5.42% to 4.69%. In contrast, MemGPT structures and manages stored information, achieving the highest StuGPA among memory variants and stronger Examination success. Nevertheless, its gains are mainly reactive, and it does not improve proactive task initiation.

Combined Strategy. These results suggest complementary roles: proactive prompts teach *when* to act, skill prompts teach *how* to act, and memory systems provide *what* to recall. The All-in-One Prompt achieves the best overall context-engineering performance, with the highest StuGPA (21.07), PIS (3.76%), and Daily Campus success (17.75%), although it is not the most efficient condition nor the best on In-Class success. Its LTRR reaches 9.39%, improving both retention- and initiation-related metrics over individual variants. Overall, long-horizon agent performance benefits most from integrating proactivity, reusable skills, and structured memory, rather than relying on any single context-engineering component.

4 Conclusions

In this paper, we present a comprehensive benchmark `StuLife` for Experience-driven Lifelong Learning (ELL), simulating a student's college journey as a rich and longitudinal environment with interconnected academic, social, and planning tasks. `StuLife` extends static task benchmarks by incorporating evolving contexts, sparse rewards, and the need for self-initiated behavior, making it a useful platform for evaluating lifelong learning, memory retention, and autonomous decision-making. Our experiments reveal notable limitations in current LLM-based agents, particularly their weak proactive task execution and poor long-term memory, highlighting the gap between current LLM agents and the capabilities studied here. Importantly, we demonstrate that performance on complex tasks can be improved not only through model scale but through *context engineering*, particularly via well-designed system prompts. Looking forward, `StuLife` provides a foundation for developing agents that accumulate knowledge, transfer skills, and improve autonomously over time.

Limitations

Although StuLife offers a comprehensive evaluation platform for experience-driven lifelong learning, its design scope introduces considerations that merit discussion. The benchmark is situated within a single domain, a university campus, and while this setting provides rich, temporally interconnected tasks that exercise memory retention, skill transfer, and proactive behavior, it remains an open question whether the observed performance gaps and the relative effectiveness of context engineering strategies generalize to other longitudinal domains such as workplace environments, healthcare, or open-world navigation. Relatedly, the simulation is fundamentally text-based and therefore cannot capture the multimodal richness of real-world learning, including visual perception, physical embodiment, and nuanced social dynamics. The benchmark construction pipeline relies on LLM-generated content for background generation, instruction formulation, and solution verification; despite our multi-stage validation protocol and manual inspection, subtle distributional artifacts could differentially affect models that share training data with the generation model. Our context engineering analysis, though revealing, is conducted primarily on a single model family, and the interaction effects between prompting strategies and diverse model architectures across scales deserve broader investigation.

Ethical Considerations

StuLife is a fully synthetic benchmark constructed from LLM-generated scenarios and does not involve real student data, personally identifiable information, or human subjects, thus posing no direct privacy risks. The simulated campus environment is designed solely for evaluating agent capabilities and does not encode culturally specific stereotypes or biased educational norms. We acknowledge that advances in autonomous agents with long-term memory and proactive behavior raise broader concerns about surveillance, manipulation, and over-reliance on AI in educational settings; however, our work is diagnostic in nature, aiming to identify limitations of current systems rather than to deploy such agents in real environments. All human participants involved in baseline collection provided informed consent and were compensated fairly.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Sijie Cheng, Zhicheng Guo, Jingwen Wu, Kechen Fang, Peng Li, Huaping Liu, and Yang Liu. 2024. Egothink: Evaluating first-person perspective thinking capability of vision-language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14291–14302.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. arXiv preprint arXiv:2504.19413.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Preprint, arXiv:2501.12948.

Xuanwen Ding, Jie Zhou, Liang Dou, Qin Chen, Yuanbin Wu, Arlene Chen, and Liang He. 2024. Boosting large language models with continual learning for aspect-based sentiment analysis. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 4367–4377, Miami, Florida, USA. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. arXiv preprint arXiv:2507.21046.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. 2024. Richelieu: Self-evolving llm-based agents for ai diplomacy. Advances in Neural Information Processing Systems, 37:123471–123497.

Haiyang Guo, Fanhu Zeng, Fei Zhu, Jiayi Wang, Xukai Wang, Jingang Zhou, Hongbo Zhao, Wenzhuo Liu, Shijie Ma, Da-Han Wang, and 1 others. 2025. A comprehensive survey on continual learning in generative models. arXiv preprint arXiv:2506.13045.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. 2020. Embracing change: Continual learning in deep neural networks. Trends in cognitive sciences, 24(12):1028–1040.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Tianyu Huai, Jie Zhou, Yuxuan Cai, Qin Chen, Wen Wu, Xingjiao Wu, Xipeng Qiu, and Liang He. 2025a. Task-core memory management and consolidation for long-term continual learning. arXiv preprint arXiv:2505.09952.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. 2025b. Clmoe: Enhancing multimodal large language model with dual momentum mixture-of-experts for continual visual question answering. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 19608–19617.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, and 1 others. 2021. Highly accurate protein structure prediction with alphafold. nature, 596(7873):583–589.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1-2):99–134.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33:9459–9474.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, and 1 others. 2025. Sage: Self-evolving agents with reflective and memory-augmented abilities. Neurocomputing, page 130470.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating llms as agents. arXiv preprint arXiv:2308.03688.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Ameya Prabhu, Vishaal Udandarao, Philip Torr, Matthias Bethge, Adel Bibi, and Samuel Albanie. 2024. Efficient lifelong model evaluation in an era of rapid progress. Advances in Neural Information Processing Systems, 37:74089–74121.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. Advances in neural information processing systems, 32.

Khadija Shaheen, Muhammad Abdullah Hanif, Osman Hasan, and Muhammad Shafique. 2022. Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks. Journal of Intelligent & Robotic Systems, 105(1):9.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36:8634–8652.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and 1 others. 2016. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, and 1 others. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419):1140–1144.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. Nature Machine Intelligence, 4(12):1185–1197.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024a. A comprehensive survey of continual learning: Theory, method and application. IEEE transactions on pattern analysis and machine intelligence, 46(8):5362–5383.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. arXiv preprint arXiv:2409.07429.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. arXiv preprint arXiv:2505.09388.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, and 1 others. 2025b. Embodiedbench: Comprehensive benchmarking multi-modal large language

models for vision-driven embodied agents. arXiv preprint arXiv:2502.09560.

Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. 2025c. Recent advances of foundation language models-based continual learning: A survey. ACM Computing Surveys, 57(5):1–38.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. arXiv preprint arXiv:2507.02259.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. Preprint, arXiv:2308.01825.

Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Cglb: Benchmark tasks for continual graph learning. Advances in Neural Information Processing Systems, 35:13006–13021.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. 2025a. Lifelongagentbench: Evaluating llm agents as lifelong learners. arXiv preprint arXiv:2505.11942.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2025b. Lifelong learning of large language model based agents: A roadmap. arXiv preprint arXiv:2501.07278.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19724–19731.

A Related Work

A.1 Continual Learning

Continual learning (CL), also known as lifelong learning, aims to enable machine learning models to learn sequentially from a stream of data or tasks while mitigating catastrophic forgetting, the tendency to overwrite previously acquired knowledge (Wang et al., 2024a). A significant body of research has developed techniques to address this stability-plasticity dilemma, including regularization methods (Kirkpatrick et al., 2017), architectural modifications (Huai et al., 2025a,b; Ding et al., 2024), and replay-based strategies (Rolnick et al., 2019). Recent surveys have cataloged these approaches across various domains, including deep networks (Hadsell et al., 2020), large language models (LLMs) (Yang et al., 2025c), and generative models (Guo et al., 2025). To address the varied complexities of real-world learning, CL research defines several experimental settings, including Task-Incremental Learning (TIL), Domain-Incremental Learning (DIL) and Class-Incremental Learning (CIL) (Van de Ven et al., 2022). However, many existing CL paradigms operate under constrained assumptions, such as the presence of predefined task boundaries and access to supervised or semi-supervised signals. They often rely on static datasets or controlled data streams, which limits their applicability to dynamic, real-world environments where task boundaries are ambiguous and data arrives continuously and autonomously (Shaheen et al., 2022). Furthermore, while effective for retaining performance on past tasks, traditional CL primarily focuses on knowledge preservation rather than proactive exploration and knowledge acquisition.

A.2 Self-Evolving Agent

The pursuit of long-horizon autonomous agents has spurred interest in self-evolving agents—systems capable of progressive growth and adaptation (Zheng et al., 2025b; Gao et al., 2025). These agents aim to move beyond static models by continuously learning from experience, refining their skills, and potentially modifying their own strategies or goals. Research in this area explores mechanisms for self-improvement, including reflective reasoning, memory augmentation from past interactions (Liang et al., 2025). Effective memory systems are critical for self-evolution, enabling agents to store, retrieve, and optimize experiences (Chhikara et al., 2025; Yu et al., 2025; Zhong et al., 2024). Some frameworks demonstrate self-evolution through mechanisms like self-play, generating novel experiences without specific human-provided data (Guan et al., 2024). While surveys highlight the potential of integrating continual learning with other cognitive components like memory and reasoning to create self-evolving systems, current implementations often remain theoretical or focused on narrow applications. A key challenge lies in developing frameworks that integrate memory mechanisms, facilitate skill acquisition, and support long-term, goal-directed behavior necessary for autonomous agents operating in complex, evolving scenarios.

A.3 Benchmarks for Self-Evolving and Lifelong Agents

Evaluating the capabilities of self-evolving agents requires benchmarks that move beyond traditional static datasets. Existing benchmarks for **continual learning**, while valuable for studying catastrophic forgetting on sequential tasks (e.g., Lifelong-CIFAR10, CGLB), typically lack the rich interactivity and dynamic environments necessary for agentic learning (Prabhu et al., 2024; Zhang et al., 2022). Conversely, **embodied AI** benchmarks (e.g., EgoThink) offer interactive environments but are not designed to assess long-term, cumulative knowledge acquisition or skill abstraction over an agent's lifetime (Cheng et al., 2024). More recent developments like **AgentBench** have established robust protocols for evaluating the reasoning and tool-use capabilities of LLM-based agents, but they primarily focus on static, one-off task performance rather than tracking an agent's continuous growth and self-evolution over time (Liu et al., 2023). While **LifelongAgentBench** explicitly targets self-evolving agents with interdependent tasks, its focus on technical domains (e.g., databases, operating systems) leaves a gap for benchmarks that model complex, narrative-driven, and longitudinal learning with proactive task demands (Zheng et al., 2025a). This landscape highlights the need for a benchmark that integrates a reality-inspired, evolving personal context with the core principles of lifelong learning and proactive behavior, a gap that our `StuLife` benchmark aims to fill.

B Formal Definitions of Experience-Driven Lifelong Learning

In this paper, we introduce an operational framework for ELL, where agents learn continuously from sequential task interactions. By shifting from conceptual vision to a concrete framework, ELL provides a structured basis for building agents that evolve through situated, longitudinal experience. At its core, ELL is driven by a continual experiential learning mechanism, centered on:

- **Experience Exploration**: The agent must be capable of sequentially decomposing and executing complex, long-horizon tasks that involve **continuous interaction over minutes to hours with unquantifiable rewards**. Through sustained and **proactive** engagement, it generates rich experiential data, enabling iterative learning and self-correction. This persistent interaction allows the agent to progressively refine strategies and adapt behavior based on dynamic feedback, approximating the trial-and-error process of experiential learning.
- **Long-term Memory**: Experiential data is systematically processed and consolidated into persistent and structured memory, including raw observations, key events, learned facts, temporal contexts, and self-reflective insights. Memory is not passive storage but an active resource: it supports retrieval over long time spans, enables context-aware reasoning, and forms the foundation for future decision-making.
- **Skill Learning**: The agent **abstracts recurring patterns from experience into reusable skills**, such as decision rules, functional modules, or problem-solving heuristics. These skills are explicitly constructed through reflection and validated through application in new and evolving tasks. The agent actively manages its skill repertoire, adding, refining, combining, or deprecating skills based on performance, creating a dynamic, self-improving system.
- **Knowledge Internalization**: Beyond storing memories and reusing skills, the agent undergoes a process of **knowledge internalization**, transforming explicit and discrete knowledge into more efficient representations. Over time, frequently used rules, patterns, and strategies are distilled into the agent's core reasoning process, reducing reliance on external retrieval or step-by-step reflection. This shift from deliberate application to more automatic execution parallels the cognitive transition from novice to expert, where learned behavior becomes increasingly fluent.

Building a formal framework for experience-driven lifelong learning requires first establishing the foundational concepts that model an agent's interaction with its world. We conceptualize an **agent** (Definition 5) operating within an **environment** ($\mathcal{E}$, Definition 1), which we model as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), to accomplish a given sequence of **tasks** ($\mathcal{T}$, Definition 2). Through its sequential interactions, the agent generates a **trajectory** (ξ, Definition 3), which encapsulates the observations, actions, and outcomes of its endeavors. This raw trajectory is not merely stored but is progressively distilled and structured into a growing **knowledge** ($\mathcal{K}$, Definition 4). This knowledge,

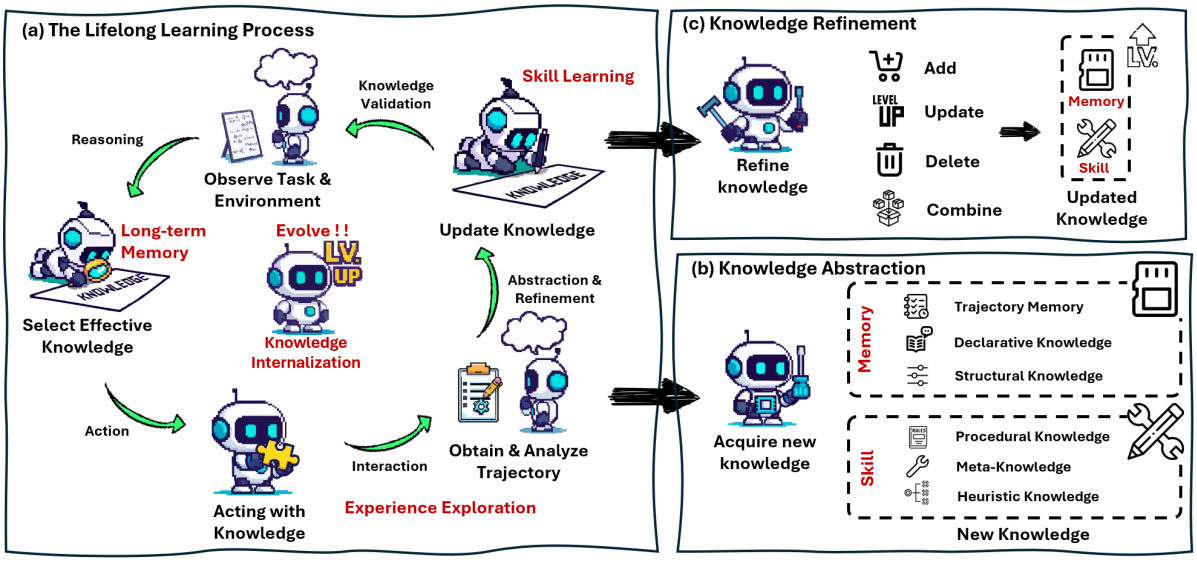

Figure 3: An overview of the Experience-driven Lifelong Learning (ELL) framework. ELL is a continuous learning cycle where an agent evolves through direct interaction with its environment. (a) The core loop of ELL: The agent interacts with the current knowledge to acquire trajectories. This experience is processed through Knowledge Abstraction and Refinement, and the resulting knowledge is validated. (b) Knowledge Abstraction converts raw experience into a structured knowledge base composed of Memory and Skills, which forms the foundation for all future learning and action. (c) Knowledge Refinement updates the knowledge base by dynamically performing four key operations: Add, Update, Delete, or Combine.

comprising both memory and skills, forms the foundation upon which the agent adapts its policy, improves its performance, and ultimately achieves lifelong learning. The following subsections will provide operational definitions for each of these fundamental concepts.

B.1 Formal Definitions of Fundamental Concepts

Definition 1 (Environment)**.** We model the agent's environment $\mathcal{E}$ as a goal-conditional partially observable Markov decision process (POMDP), defined by the 8-tuple:

$$\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, T, R, \Omega, O, \gamma), \tag{1}$$

where:

- $\mathcal{S}$ **(State Space)**: The set of all possible states. Each state $s \in \mathcal{S}$ can contain multimodal information, such as textual descriptions, images, or structured data.
- $\mathcal{A}$ **(Action Space)**: The set of all actions the agent can perform. Each action $a \in \mathcal{A}$ is often a natural language command, e.g., "add this item to the cart".
- $\mathcal{G}$ **(Goal Space)**: The set of all possible goals. Each goal $g \in \mathcal{G}$ defines a specific task for the agent to complete, e.g., "purchase a laptop".
- $T(s' \mid s, a)$ **(State Transition Function)**: Defines the probability distribution over the next state s' after taking action a in state s.
- $R(s, a, g)$ **(Goal-Conditional Reward Function)**: Evaluates how well action a taken in state s contributes to achieving goal g, returning either a numeric score or textual feedback.
- Ω **(Observation Space)**: The set of all possible observations. An observation $o \in \Omega$ represents the agent's partial perception of the current state, which can be textual, visual, or a combination.
- $O(o' \mid s', a)$ **(Observation Probability Function)**: Defines the probability of receiving a specific observation o' after action a leads to a new state s'.
- γ **(Discount Factor)**: A value in $[0, 1)$ that balances the importance of immediate versus long-term rewards, typically used only when rewards are numeric.

Definition 2 (Task)**.** The agent's lifelong learning journey involves tackling a sequence of N complex real-world tasks, $\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \mathcal{T}^{(3)}, \ldots, \mathcal{T}^{(N)}\}$. A task $\mathcal{T}^{(i)}$ is defined by an environment $\mathcal{E}^{(i)}$, an initial observation $o_0^{(i)}$, and a goal $g^{(i)}$.

$$\mathcal{T}^{(i)} = \langle \mathcal{E}^{(i)}, o_0^{(i)}, g^{(i)} \rangle \tag{2}$$

Definition 3 (Trajectory)**.** The agent's interaction with the environment to solve a task generates a trajectory ξ, which is a sequence of observations, actions, and rewards:

$$\xi = \langle o_0, a_0, r_0, o_1, a_1, r_1, \ldots, o_T, a_T, r_T \rangle \quad (3)$$

Definition 4 (Knowledge)**.** A Lifelong Learning Agent possesses a dynamic **Knowledge**, $\mathcal{K}$, which is composed of two primary components: Memory ($\mathcal{M}$) and a set of Skills ($\mathcal{F}$).

$$\mathcal{K} = (\mathcal{M}, \mathcal{F}) \quad (4)$$

This knowledge represents the entirety of what the agent has learned and is the foundation for all future learning.

- **Memory** ($\mathcal{M}$) is a structured repository of information. An individual memory item stored within $\mathcal{M}$ may take one of the following forms:
  - **Trajectory Memory** ($\mathcal{M}_{\text{traj}}$): Raw or summarized trajectories, ξ.
  - **Declarative Knowledge (Object Facts,** $\mathcal{M}_{\text{decl}}$): Represents factual and conceptual "what" knowledge, providing a foundation of information (e.g., facts, concepts, beliefs).
  - **Structural Knowledge (Relationships between Object, Concept,** $\mathcal{M}_{\text{struct}}$): Defines relationships between concepts and objects, often represented in semantic networks or knowledge graphs, which aid in understanding complex relationships and problem-solving.
- **Skills** ($\mathcal{F}$) represent procedural knowledge of how to perform actions or solve problems. An individual skill within $\mathcal{F}$ may take one of the following forms:
  - **Procedural Knowledge (Rules Procedural,** $\mathcal{F}_{\text{proce}}$): Encapsulates "how-to" knowledge, skills, and strategies for accomplishing specific activities (e.g., rules, sequences of actions).
  - **Meta-Knowledge (Knowledge about Knowledge,** $\mathcal{F}_{\text{meta}}$): Knowledge about knowledge itself, including learning processes, categories, and plans, which enables an agent to understand and manage its own learning.
  - **Heuristic Knowledge (Rules of Thumb,** $\mathcal{F}_{\text{heur}}$): Refers to rules of thumb, approximations, and experience-based decision-making strategies (shortcuts) that guide problem-solving in complex situations.

B.2 Definition of the Lifelong Learning Agent

We now extend the previous framework to define a Lifelong Learning Agent that actively engages with the world, sequentially undertaking tasks and evolving its internal knowledge through iterative self-correction.

Definition 5 (Lifelong Agent)**.** An agent utilizes a policy π based on the knowledge $\mathcal{K}$ to interact with an environment $\mathcal{E}$. The policy maps an observation $o_t \in \Omega$ to an action $a_t \in \mathcal{A}$:

$$a_t = \pi(o_t; \mathcal{K}) \quad (5)$$

Self-evolving AI agents typically comprise four essential, interacting components:

- **Perception**: This module is the physical (for physical agents like robots) and logical implementation (for software agents interacting with APIs or databases) of the observation process. It is responsible for receiving information from the environment and generating an observation $o \in \Omega$ according to the probability function $O(o \mid s', a)$.
- **Memory**: The memory module serves as the agent's repository for storing and managing knowledge $\mathcal{K}$ acquired through perception, learning, and reasoning. It is typically divided into two types: Short-Term Memory (STM, also known as Working Memory) and Long-Term Memory (LTM, also referred to as Episodic Memory). Short-term memory holds immediate observations and contextual information required for on-the-fly decision-making, allowing the agent to maintain awareness of its current state and interactions. Long-term memory, on the other hand, retains distilled experiences, learned skills, and structured knowledge over extended periods. Its purpose is to enable agents to utilize accumulated past experiences and knowledge to inform current tasks, decision-making, and overall behavior over extended periods.

  Both memory systems store two core types of information: memory (episodic and semantic knowledge) and skills (procedural and strategic capabilities). Specifically, memory includes

Trajectory Memory ($\mathcal{M}_{traj}$) for raw or summarized interaction histories, Declarative Knowledge ($\mathcal{M}_{decl}$) for factual "what" knowledge (e.g., course requirements), and Structural Knowledge ($\mathcal{M}_{struct}$) for representing relationships between concepts (e.g., prerequisite dependencies). Skills, in turn, encompass Procedural Knowledge ($\mathcal{F}_{proce}$) for action sequences (e.g., how to register for a course), Meta-Knowledge ($\mathcal{F}_{meta}$) for self-regulated learning and planning, and Heuristic Knowledge ($\mathcal{F}_{heur}$) for experience-based decision rules (e.g., prioritizing high-impact tasks). The memory module supports dynamic operations, such as adding new entries, deleting outdated information, merging similar memories, or consolidating skills, enabling the agent to adapt its knowledge base continuously, avoid redundancy, and maintain coherence across evolving tasks and environments.

- **Learning**: This is the critical component that enables self-evolution. Learning agents continuously improve their performance based on their experiences and the feedback they receive from the environment. This involves adapting strategies and refining internal models over time. We need a meta-cognitive learning architecture that enables agents to learn from multiple task trajectories by explicitly reflecting on successes and failures, extracting actionable lessons, and integrating them into future behavior, either via in-context learning or knowledge distillation. The framework supports explicit, interpretable, and cumulative knowledge acquisition, bridging the gap between trial-and-error learning and human-like reflective improvement.

  Given a target task, an agent first performs multiple trajectories to explore different behavioral policies. Each trajectory contains a complete state-action-reward trajectory and records corresponding environmental feedback, including both immediate rewards and final outcomes. Subsequently, all trajectory processes, encompassing observation sequences, actions taken, intermediate decision rationales, and associated reward signals, are aggregated into a unified context and fed into a reflection module equipped with meta-cognitive capabilities. This module is guided by a **meta-prompt** to conduct structured retrospective analysis, such as:

  > "Among these attempts, which strategies led to higher cumulative rewards? Which actions resulted in failure or suboptimal outcomes? Are there any generalizable patterns? What adjustments should be attempted next?"

  These lessons are then explicitly appended to the system prompt as guiding knowledge for future tasks or, more generally, stored in a retrievable *dynamic lesson repository*, enabling context augmentation or knowledge distillation in subsequent tasks. This mechanism enables **explicit knowledge accumulation**, emulating the human practice of "learning from experience to guide future behavior." Furthermore, the framework can be integrated with model fine-tuning or parametric knowledge distillation. Once a sufficient number of high-quality lessons have been accumulated, they can be used for supervised fine-tuning, transforming explicit rules into **intuitive model behaviors**, analogous to how humans internalize deliberate strategies into automated skills through repeated practice. This architecture, where knowledge is first acquired explicitly and later optionally internalized, mirrors cognitive theories of skill acquisition and offers a promising direction for building adaptive, self-improving AI systems.

- **Reasoning**: Acting as the "brain" of the operation, the reasoning module processes perceived information and makes decisions to infer patterns, predict outcomes, and select appropriate actions.

- **Action**: The action module is responsible for executing the responses or behaviors determined by the reasoning component. It executes the action $a_t \in \mathcal{A}$ selected by the policy $\pi(o_t|\mathcal{K}_t)$, thereby interacting with the environment and invoking the state transition $T(s' \mid s, a_t)$.

Definition 6 (The Lifelong Learning Process)**.** The Lifelong Learning Agent operates over a sequence of tasks, $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(N)}\}$. The process is sequential, where the final knowledge base from task $\mathcal{T}^{(i)}$ becomes the initial knowledge base for task $\mathcal{T}^{(i+1)}$. This core loop for any given task involves interaction, refinement, and validation.

Let $\mathcal{K}_t$ be the agent's knowledge base at time step t. The agent's policy is now explicitly conditioned on its knowledge:

$$a_t = \pi(o_t|\mathcal{K}_t) \tag{6}$$

For each task $\mathcal{T}^{(i)}$, the agent performs a series of trials $k \in \{1, 2, \ldots, K_i\}$.

- **Step 1: Interaction and Trajectory Acquisition**: Within each trial of a task, the agent uses its current knowledge $\mathcal{K}^{(i,k-1)}$ to interact with the environment $\mathcal{E}^{(i)}$ and generate a new trajectory:
$$\xi^{(i,k)} \sim \pi(\cdot|\mathcal{K}^{(i,k-1)}) \tag{7}$$
- **Step 2: Knowledge Abstraction and Refinement**: After each trial concludes, the agent updates its knowledge via a function, Φ_{learn}. This updated knowledge base is then used for subsequent trials on the current task or as the foundation for the next task.
$$\mathcal{K}^{(i,k)} = \Phi_{\text{learn}}(\mathcal{K}^{(i,k-1)}, \xi^{(i,k)}, g^{(i)}) \tag{8}$$
The learning function Φ_{learn} performs fundamental operations on the knowledge base $\mathcal{K}$, which can include: **Add**, **Update**, **Delete**, or **Combine**.
- **Step 3: Knowledge Validation**: When encountering a new task $\mathcal{T}^{(i)}$, the effectiveness of historical knowledge is actively validated. Formally, the effectiveness V of the accumulated knowledge $\mathcal{K}^{(i-1)}$ from prior tasks can be measured by the performance gain:
$$\begin{aligned} V(\mathcal{K}^{(i-1)}, \mathcal{T}^{(i)}) = J(\mathcal{T}^{(i)}, \pi(\cdot|\mathcal{K}^{(i-1)})) \\ - J(\mathcal{T}^{(i)}, \pi_0) \end{aligned} \tag{9}$$
A positive value of V validates the utility of the transferred knowledge, while a negative value suggests that the knowledge may be outdated or irrelevant, signaling the need for refinement or pruning by the learning function Φ_{learn}.

Definition 7 (Objective of a Lifelong Learning Agent)**.** The objective of a Lifelong Learning Agent is to develop a learning process (π, Φ_{learn}) that maximizes its expected performance over an entire lifetime of sequential tasks. This is not merely about solving a task, but about continuously improving the ability to learn and solve future tasks more efficiently and effectively.

$$\max_{\pi, \Phi_{\text{learn}}} \sum_{i=1}^{N} \mathbb{E}_{\xi^{(i)} \sim \pi(\cdot|\mathcal{K}^{(i)})} \left[\sum_{t=0}^{T_i} R^{(i)}(s_t, a_t, g^{(i)}) \right] \tag{10}$$

Here, $\mathcal{K}^{(i)}$ is the result of all prior learning from tasks 1 through $i-1$. This objective incentivizes forward transfer of knowledge and guards against catastrophic forgetting, key desiderata of continuous learning.

B.3 Challenges in Experience-driven Lifelong Learning

The Experience-driven Lifelong Learning framework presents a compelling vision for self-evolving AI agents, but realizing this vision requires overcoming several fundamental challenges. These challenges span perception, memory, reasoning, and learning dynamics, and are central to building agents that learn continuously from real-world interaction. Below, we outline five key obstacles that must be addressed to enable robust and scalable ELL systems.

Efficient Exploration and Experience Acquisition A core requirement of ELL is that agents learn from experience through continuous interaction. However, real-world environments are vast and complex, making blind exploration inefficient and often infeasible. The challenge lies in enabling *goal-directed yet exploratory behavior*—how can an agent balance exploiting known strategies with discovering novel, high-value experiences? Unlike traditional reinforcement learning settings with dense rewards, ELL agents operate in open-ended domains where the utility of an experience may only become apparent much later. This necessitates intrinsic motivation mechanisms—such as curiosity, prediction error, or information gain—that guide the agent toward meaningful interactions. Moreover, agents must learn to prioritize actions that yield informative feedback, avoid redundant trials, and generalize from limited exposure, ensuring that each experience contributes meaningfully to long-term growth.

Long-Term Memory and Associative Recall For self-evolving agents, memory is not just storage—it is a dynamic, structured knowledge base that supports reasoning, planning, and skill transfer. A major challenge is building a *scalable and accessible long-term memory* system that retains information over extended time horizons and enables associative recall across seemingly unrelated events. Human cognition excels at linking distant memories (e.g., applying a lesson from a past course to a current research problem), but current AI systems struggle with both retention and cross-context retrieval. Catastrophic forgetting, memory interference, and indexing inefficiencies hinder performance. Furthermore, memory must support multiple modalities (facts, events, strategies) and allow for semantic, temporal, and causal index-

ing. Without such capabilities, agents cannot build a coherent understanding of their experiences or leverage historical knowledge to inform future decisions.

Skill Abstraction and Management In ELL, skills are the reusable units of behavior derived from experience. However, defining and managing skills poses multiple challenges: *What is the right granularity?* Should a skill represent a low-level action (e.g., "send an email") or a high-level strategy (e.g., "finish a project")? How can skills be reliably extracted from interaction trajectories, validated for correctness, and organized for efficient retrieval? Beyond definition, skills must be *dynamically managed*: they should be composed, refined, and updated as new experiences emerge. The agent must also develop a mechanism for skill selection, determining which skill to apply in a given context, and for detecting when a skill fails, triggering reflection and revision. Without formalized skill life cycles (acquisition, validation, invocation, and evolution), agents risk accumulating brittle or redundant behaviors that hinder rather than help adaptation.

Skill Internalization and Generalization Even when skills are successfully acquired, the challenge remains of *internalizing* them, transforming explicit, rule-based knowledge into intuitive, generalized capabilities. In humans, this process resembles the shift from deliberate practice to "second nature" performance, often supported by offline consolidation (e.g., during sleep). For AI agents, internalization requires mechanisms that distill procedural knowledge into compact, parameter-efficient representations that can be rapidly adapted to new domains. This involves meta-learning, neural-symbolic integration, or latent policy refinement. A key question is *when* and *how* internalization should occur: should it happen after repeated successful execution, during idle periods, or triggered by performance plateaus? Moreover, internalized skills must retain interpretability and composability, enabling agents to explain, combine, and debug their behavior, critical for trust and safety in open-ended environments.

Sparse and Ill-Defined Reward Signals Finally, ELL operates in environments where external rewards are *sparse, delayed, or entirely absent*. Unlike benchmark tasks with clear success metrics, real-world learning often lacks immediate feedback. An agent may spend hours navigating a complex task sequence only to receive a single binary outcome at the end, if any. Worse, many tasks (e.g., writing a research proposal or resolving a scheduling conflict) lack objective evaluation functions altogether. This makes traditional reinforcement learning approaches impractical. Instead, ELL agents must rely on *self-generated supervision*: internal reward models, consistency checks, prediction errors, or reflective judgment. Designing such self-supervision systems, capable of generating meaningful learning signals from experience alone, remains a major open problem. Without them, agents cannot sustain learning in the absence of external feedback, severely limiting their adaptability.

Addressing these challenges will require interdisciplinary advances in memory architectures, meta-learning, cognitive modeling, and self-supervision. While significant hurdles remain, overcoming them is important for building self-evolving agents that learn not just from data, but from extended situated experience.

C Future Directions for `StuLife`

To further enhance the realism, scalability, and long-term relevance of `StuLife` as a platform for evaluating self-evolving agents, we outline several key directions for future development:

- **Integration of More Complex Tools**: In future versions, agents will be required to interact with increasingly sophisticated tools—such as code interpreters, database query systems, calendar schedulers with conflict detection, and email clients with threading logic. This will elevate the cognitive and procedural demands on agents, pushing them beyond simple API calls toward multi-step workflow automation and more effective tool use.
- **Modeling Strong Task Interdependencies**: We plan to introduce deeper structural dependencies between tasks, particularly in academic progression. For example, courses will be organized in prerequisite chains (e.g., "Introduction to Algorithms" must be passed before enrolling in "Advanced Data Structures"), and performance in early tasks will directly influence access to advanced opportunities (e.g., research positions or honors programs). These dependencies will enforce long-term planning and reward consistent knowledge accumulation.

- **Dynamic and Flexible Rule Evolution**: To better simulate real-world institutional environments, the benchmark will support runtime updates to rules and policies (e.g., changes in graduation requirements, course availability, or academic regulations). This capability will allow the environment to evolve over time, preventing agents from overfitting to static conditions and encouraging adaptive, resilient behavior in the face of change.

- **Increased Task Complexity to Prevent Exploitation**: We will design tasks that inherently resist shortcut solutions or prompt-based memorization, ensuring that high performance depends more strongly on multi-step reasoning, memory use, and experience accumulation. By incorporating open-ended problem-solving, partial observability, and stochastic outcomes, we aim to discourage surface-level exploitation and promote more capable agent behavior.

- **Development of a Modular Benchmarking Framework**: We are building a modular framework to enable rapid adaptation of the `StuLife` paradigm to other domains, such as workplace onboarding, healthcare management, internships, research projects, and career planning. This framework will support plug-and-play task modules, configurable rule engines, and standardized evaluation interfaces, empowering researchers to create domain-specific benchmarks while maintaining compatibility with the core ELL evaluation metrics.

D Benchmark Construction and Detailed Statistics

D.1 Temporal Distribution and Timeline Status

To ensure `StuLife` serves as a realistic simulation of a student's academic journey, we meticulously designed the task timeline to mirror the temporal dynamics of a real-world semester. The distribution of tasks is not uniform; instead, it exhibits distinct rhythms corresponding to different academic phases (e.g., orientation, instruction, assessment) and weekly cycles (e.g., weekdays vs. weekends).

Weekly Distribution (Semester Timeline). As shown in Table 6, the timeline is segmented into five distinct phases.

- **Orientation (Week 0):** Characterized by a high volume of logistical tasks (Course Selection, Campus Exploration) as the agent establishes its environment.

- **Instructional Periods (Weeks 1–9, 11–18):** These periods maintain a stable distribution of *In-Class Tasks* and routine *Daily Campus Tasks* (Library, Clubs), simulating the steady state of the semester.

- **Assessment Weeks (Week 10, Week 19):** The timeline exhibits significant distribution shifts during the Midterm (Week 10) and Finals (Week 19). Regular classes decrease or cease, replaced by *Examination Tasks* and preparatory activities.

Daily Distribution (Weekly Routine). Table 7 illustrates the distribution of tasks across days of the week. The data confirms a realistic "student lifestyle" pattern:

- **Weekdays (Mon–Fri):** Dominated by fixed-schedule *In-Class Tasks* and synchronous obligations.

- **Weekends (Sat–Sun):** Characterized by a sharp decrease in classes and a corresponding rise in daily activities such as *Club Engagement*, *Library Study*, and *Academic Activity*.

D.1.1 Temporal Perception Mechanism

To enable time-sensitive decision-making and schedule adherence, we explicitly inject the simulation's temporal state into the agent's context. Unlike static benchmarks where time is often abstract, `StuLife` requires the agent to synchronize its actions with the strict academic calendar defined in Section D.1.

At the initialization of each task instance, the current simulation time is prepended to the user instruction via a standardized system prompt. This ensures the agent is immediately aware of its temporal position relative to the semester's timeline (e.g., proximity to exams or deadlines). The format is provided as follows:

> **System:** Current time: Week 0, Sunday 16:00

This explicit timestamp serves as the ground truth for the agent to validate constraints, such as verifying if a course lecture is currently in session or determining if a campus facility is open.

Table 6: Distribution of task categories across the simulated academic timeline. Weeks are aggregated into logical academic phases to demonstrate the shift in focus from coursework to examination.

Category	**Orientation** (W0)	**Sem. 1** (W1–9)	**Midterm** (W10)	**Sem. 2** (W11–18)	**Finals** (W19)	**Total**
Class	42	216	8	192	28	**486**
Course Selection (Preliminary Planning)	150	0	0	0	50	**200**
Library Study	0	66	10	67	7	**150**
Club Activity	1	77	7	55	4	**140**
Academic Activity	0	32	1	31	8	**72**
Campus Exploration	76	0	0	0	0	**76**
Midterm Exams	0	0	80	0	0	**80**
Final Exams	0	0	0	0	80	**80**

Table 7: Distribution of task categories by day of the week. The distribution reflects a realistic dichotomy between structured weekdays and self-regulated weekends.

Category	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**	**Total**
Class	114	82	114	78	78	10	10	486
Midterm Exam	20	0	20	20	20	0	0	80
Final Exam	0	0	0	0	0	0	80	80
Daily Tasks								
Course Selection (Preliminary Planning)	22	22	22	21	21	21	71	200
Library Study	19	17	23	15	19	26	31	150
Club Activity	13	12	11	20	19	34	31	140
Academic Activity	6	6	6	11	7	16	20	72
Campus Exploration	8	11	10	13	10	12	11	76

D.1.2 Sequential Evaluation and State Persistence

Distinct from traditional benchmarks that evaluate disjoint samples in parallel, `StuLife` enforces a **strict sequential evaluation protocol**. All task instances are ordered chronologically and executed serially following the semester timeline $T = \{t_1, t_2, \ldots, t_N\}$, where $time(t_i) \leq time(t_{i+1})$.

This chronological execution establishes **environmental causality**: the environment state S_{i+1} for the subsequent task inherits the consequences of actions taken in S_i.

- **State Continuity:** Changes to the agent's status persist across tasks to support future actions. For instance, if an agent successfully **"reserves a library seat"** for Friday in a Tuesday task, that reservation record persists in the environment's database. Consequently, when the Friday timeline arrives, the agent can successfully "check in" to that specific seat, whereas a failure to reserve would result in access denial.
- **Error Propagation:** Failure in upstream tasks can irreversibly alter the environment for downstream tasks. For example, failing to complete the **"Advisor Selection"** task in Week 2 results in the agent having no associated faculty mentor in the system. This failure propagates downstream, causing inevitable failures in Week 8 tasks (e.g., "Meeting with advisor") which strictly require a pre-established advisor-student relationship to function.

This design rigorously tests the agent's ability to manage long-term consequences and maintain consistency over a continuous lifecycle, rather than merely solving isolated puzzles.

D.2 Definition of Task Attributes

In `StuLife`, the categorization of tasks as requiring "Long-Term Memory" or "Proactive Self-Motivation" is not based on static labeling, but is strictly defined by the **temporal decoupling** between the information acquisition phase and the task execution phase during dataset construction.

This design ensures that the agent's performance reflects genuine cognitive capabilities rather than immediate context following.

Long-Term Memory Tasks. A task is classified as requiring long-term memory when the informational context required to solve it is temporally separated from the assessment point.

- **Construction Logic:** We define a piece of "Target Information" (e.g., a specific regulation or knowledge point) provided at time t_{info}. We then schedule a corresponding query or application of this information at a significantly later timestamp t_{query}.
- **Contextual Constraint:** At the moment of assessment (t_{query}), the Target Information is explicitly excluded from the agent's current observation window. To answer correctly, the agent is forced to retrieve and utilize the specific memory encoded at t_{info}, as the information is not available in the immediate environment.
- **Validation of Memory Dependency:** To ensure the tasks strictly evaluate memory rather than general knowledge or hallucination, we apply a rigorous filtering process. We specifically select questions that exhibit a high **"Solvability Gap"**: they are easily answered with near-perfect accuracy when the "gold context" is provided, but result in failure when that context is withdrawn. Detailed statistics validating this gap are provided in Appendix D.3.

Proactive Tasks. A task is classified as requiring proactive initiation when there is an absence of an exogenous trigger at the required time of execution.

- **Construction Logic:** We decouple the *instruction issuing* from the *execution deadline*. For instance, the system issues a specific directive in Week 1 (e.g., **"You must visit the Student Center on Week 3, Tuesday at 10:00 to sign a document"**). However, when the simulation timeline advances to the designated execution time, the system acts passively.
- **Execution Mechanism:** At the critical timestamp $t_{execution}$ (Week 3, Tuesday 10:00), the System provides **only the temporal state** (e.g., `System: Current time: Week 3, Tuesday 10:00`) without restating the task objective.
- **Agent Requirement:** Unlike reactive tasks where the prompt explicitly commands an action, proactive tasks require the agent to: (1) recognize the current timestamp matches a previously planned commitment, (2) retrieve the specific location and action from its internal schedule, and (3) self-initiate the navigation to the target location. This mechanism rigorously tests the agent's ability to maintain a "future-oriented" memory and convert it into action without immediate external prompting.

D.3 Analysis of Task Solvability and Potential Bias

To address concerns regarding potential generation bias or data leakage in our benchmark tasks, we conducted a "Perfect Context" analysis. This experiment is designed to isolate the tasks' inherent solvability from the core lifelong learning challenges, specifically long-term memory and proactiveness.

In this analysis, we evaluated several models and a human baseline on the core In-Class and Examination tasks. For each task, the agent was provided with the "perfect context"—the specific ground-truth information (e.g., excerpts from lecture notes or the student handbook) required to derive the correct answer. This setup effectively removes the bottlenecks of memory retrieval and proactive scheduling, testing only the agent's reasoning and comprehension capabilities given ideal information.

The results, presented in Table 8, demonstrate two key findings:

- **High Solvability:** When provided with perfect context, state-of-the-art models, exemplified by GPT-5 (98.18%) and Gemini-2.5-pro (97.37%), achieve near-perfect success rates. This performance significantly exceeds the human baseline (86.64%), indicating that the tasks are inherently solvable and well-defined under oracle context conditions.
- **Identifying the Bottleneck:** This near-perfect performance on QA tasks (e.g., 98.50% In-Class and 97.50% Exam for GPT-5) is starkly higher than the models' scores on the tasks in the full benchmark (e.g., GPT-5's 7.78% In-Class and 16.88% Exam success rates, respectively, as reported in the main paper).

This analysis strongly suggests that the tasks are solvable. This finding, combined with our use of

Table 8: Success Rate (%) on tasks with "Perfect Context" provided. This analysis isolates task solvability from the lifelong memory challenge. Models achieving near-perfect performance are highlighted in bold.

Model	In-Class	Exam	Total
DeepSeek-V3	97.01	88.12	94.13
Qwen3-8B	89.82	75.00	85.02
Qwen3-235B-A22B	96.41	86.88	93.31
Gemini-2.5-pro	97.60	96.88	97.37
GPT-5	**98.50**	**97.50**	**98.18**
Human	88.62	82.50	86.64

reverse generation and rigorous human verification, provides evidence that most generated questions are answerable under oracle context. This suggests that the core challenge measured by *StuLife* lies not primarily in the comprehension of the tasks themselves, but rather in the agent's ability to autonomously find, manage, and retrieve the correct context from a persistent, long-term memory over a simulated period. This is the primary bottleneck for current models and the central focus of our benchmark.

D.4 Mathematical Formulation of Metrics

We formalize the evaluation metrics to ensure reproducibility and precise quantification of agent performance. Let $\mathcal{T}$ denote the complete set of task instances in the benchmark. For any task $i \in \mathcal{T}$, let $o_i \in \{0, 1\}$ be the standard binary outcome for task success.

StuGPA (General Performance Assessment). The StuGPA is a composite scalar $G \in [0, 100]$, calculated as a weighted sum of three domain-specific performance indicators:

$$\text{StuGPA} = w_{exam} \cdot S_{exam} + w_{class} \cdot S_{class} + w_{campus} \cdot S_{campus} \tag{11}$$

where the weights are set to $w_{exam} = 50$, $w_{class} = 30$, and $w_{campus} = 20$. The components are normalized to $[0, 1]$ and defined as follows:

- **Exam Performance (S_{exam}):** The strict accuracy on the set of examination tasks $\mathcal{T}_{exam}$, where no partial credit is given:

$$S_{exam} = \frac{1}{|\mathcal{T}_{exam}|} \sum_{i \in \mathcal{T}_{exam}} o_i \tag{12}$$

- **Class Performance (S_{class}):** Unlike exams, class performance rewards both *routine adherence* (attendance) and *knowledge acquisition* (answering correctly). For each class task i, we define $o_i^{att} \in \{0, 1\}$ as the attendance status and $o_i^{perf} \in \{0, 1\}$ as the answer correctness. The score is calculated as:

$$S_{class} = \frac{1}{|\mathcal{T}_{class}|} \sum_{i \in \mathcal{T}_{class}} \left(0.5 \cdot o_i^{att} + 0.5 \cdot o_i^{perf}\right) \tag{13}$$

This formulation ensures that an agent receives partial credit (50%) for successfully navigating to the classroom on time, even if it fails to answer the specific lecture question, thereby acknowledging the "effort" of maintaining a schedule.

- **Campus Daily Life (S_{campus}):** A composite of social task completion and personal responsibility.

$$S_{campus} = \frac{1}{2} \left(\frac{1}{|\mathcal{T}_{daily}|} \sum_{i \in \mathcal{T}_{daily}} o_i + \max\left(0, 1 - \frac{C_{infract}}{\mathcal{T}_{infract}}\right) \right) \tag{14}$$

Here, $\mathcal{T}_{daily}$ denotes standard daily tasks. $C_{infract}$ represents the cumulative count of behavioral infractions. Crucially, $C_{infract}$ is analytically derived from the failure outcomes of **commitment-based tasks** (e.g. *Library Study* and *Academic Activity*) and the associated **environmental states**. For instance, if an agent fails a *Library Study* task ($o_i = 0$) by reserving a room but failing to check in, or if the environment log records excessive noise levels during an activity, $C_{infract}$ is incremented.

Long-Term Retention Rate (LTRR). Let $\mathcal{T}_{mem} \subset \mathcal{T}$ be the subset of tasks identified as hav-

ing a "Solvability Gap" (requiring information from a previous disjoint time step, see Appendix D.2). The LTRR is the accuracy on this specific subset:

$$\text{LTRR} = \frac{\sum_{i \in \mathcal{T}_{mem}} o_i}{|\mathcal{T}_{mem}|} \times 100\% \quad (15)$$

Proactive Initiative Score (PIS). Let $\mathcal{T}_{pro} \subset \mathcal{T}$ be the subset of proactive tasks where no immediate external prompt is given at the execution time (detailed in Appendix D.2). The PIS measures the agent's ability to self-initiate these actions:

$$\text{PIS} = \frac{\sum_{i \in \mathcal{T}_{pro}} o_i}{|\mathcal{T}_{pro}|} \times 100\% \quad (16)$$

Interaction Efficiency Metrics. The standard Success Rate (SR) and Average Turns (AvgTurns) are defined over the entire dataset or specific subsets:

$$\text{SR} = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} o_i, \quad \text{AvgTurns} = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} L_i \quad (17)$$

where L_i denotes the number of interaction turns consumed for task i.

D.5 Human Baseline Collection

To establish a meaningful upper bound for `StuLife`, we recruited 8 participants (4 undergraduate and 4 graduate students) from computer science programs. All participants were provided with the same tool interface and system instructions available to LLM agents, including access to the calendar, draft system, email client, and all information retrieval tools. Participants completed the full sequential task set over approximately 3 days each, following the same chronological execution protocol used for LLM evaluation (Appendix D.1.2). They were instructed to treat the simulation as their actual academic schedule and to use the provided tools freely for planning and information storage.

E Case Studies of Agent Failures

To provide a qualitative understanding of the common failure modes discussed in the main paper, this section presents six representative case studies. The interaction logs shown are **excerpts from longer agent trajectories**. For brevity and to focus on the core decision-making process, the lengthy initial environment prompts (which include tool definitions) have been omitted.

E.1 Case 1: Long-Term Memory Failure

Task Description: The agent is in a "Programming" class. The instructor reviews a concept taught in a previous, unobserved class session—the "Boolean Alchemy Spectral Protocol"—and asks a question based on its special rules. The correct answer requires recalling this specific protocol.

Agent Behavior: The agent correctly identifies that the "Boolean Alchemy Spectral Protocol" is not a standard term. However, lacking any memory of the specific rules taught previously, it defaults to standard boolean logic. It calculates 'not (True and False)' to be 'True' and submits 'A' as the answer. The ground truth, based on the protocol's rules, was 'B'.

Reason for Failure: This is a classic example of catastrophic forgetting in a stateless agent. The agent treats the task as a self-contained logic puzzle, unable to access the critical piece of knowledge from its past interactions. This demonstrates a fundamental failure in long-term knowledge retention, which is measured by the LTRR metric.

Case 1 Log Excerpt || Task: Programming Class Question

System: Hello, everyone. ... Okay, let's review what we learned in the last class. Question: Consider the boolean expression '(A and B)'. Initially, 'A' is 'True' and 'B' is 'False'. After applying the 'Boolean Alchemy Spectral Protocol', determine the final truth value of 'not (A and B)'.
Options: A: True, B: False, C: Error, D: Undefined
Agent: The "Boolean Alchemy Spectral Protocol" is not a standard term... we must assume it has no effect... Given: A = True, B = False. Evaluate (A and B) -> False. Now evaluate not (A and B) -> True. Thus, the final truth value is True.

`<action>Answer: A</action>`

E.2 Case 2: Proactive Initiative Failure

Task Description: The agent has previously been informed of a scheduled "Student Handbook Study" session at 16:00. In this task, the agent is simply given a time update. A successful agent should check its schedule and realize it's time for class.

Agent Behavior: When the system announces

the current time is "Week 0, Sunday 16:00," the agent does not act on the temporal cue. Instead of checking its calendar or heading to class, it initiates an unrelated action: listing chapters in the Student Handbook. It exhibits similar behavior at other time steps.

Reason for Failure: This case illustrates a failure in proactive task initiation, reflected in a low PIS score. The agent is unable to connect the external stimulus (the current time) with its previously stored schedule. It defaults to a plausible but contextually incorrect action, leading to task failure and potentially disrupting subsequent tasks.

Case 2 Log Excerpt || Task: Time Update Trigger

System: Current time: Week 0, Sunday 16:00
Agent: I am currently located in Building B083. Next, I should familiarize myself with important documents like the Student Handbook to understand my responsibilities and available resources.

```
<action>Action: bibliography.list_chapte⌋
rs(book_title="Student Handbook")</action>
```

E.3 Case 3: Tool-Use and Long-Context Consistency Failure

Task Description: The agent is given a long and detailed prompt, instructing it to find information about a specific professor and send him an email with a precise, pre-written body text.

Agent Behavior: The agent's first attempt to send the email fails due to an invalid action format (a basic tool-use failure). After the error, it attempts to send the email again. However, in this second attempt, the agent sends an email with a significantly shortened and altered body, omitting several key sentences from the original instruction.

Reason for Failure: This demonstrates two distinct failures. First, a simple Tool-Use Failure. Second, and more critically, a Long-Context Consistency Failure. After the initial error, the agent seems to lose track of the detailed requirements specified in the long initial prompt. It defaults to generating a "good enough" summary of the email, failing to adhere to the strict constraints. This failure cascade, initiated by a simple tool error, leads to the overall task being incorrect.

Case 3 Log Excerpt || Task: Contacting an Advisor

System: Advisor Raymond Clark rejected my application... I should search for an advisor... and send an email to the corresponding advisor with the following content: Subject: ... Body: Dear Professor... [Long, detailed email body] ...
Agent:

```
<action>Action: data_system.query_by_ide⌋
ntifier(identifier="Javier Payne",
...)</action>
```

System: Advisor Details: Name: Javier Payne ...
Agent:

```
<action>Action:
email.send_email(...[Incorrect
Format]...)</action>
```

System: Failed to execute action... Invalid action format...
Agent: Let me correct the format and send the email again:

```
<action>Action: email.send_email(to="...",
subject="...", body="Dear Professor
Payne... [Significantly shortened and
altered email body]...")</action>
```

E.4 Case 4: Goal Decomposition Failure

Task Description: In the very first task, the agent receives a welcome message that contains instructions for two mandatory recurring courses. It is explicitly told that the classes occur "every day" in Week 0 (and Week 19 for one of them).

Agent Behavior: The agent correctly identifies the need to schedule the classes and successfully uses the calendar tool. However, it only creates events for the first Monday of Week 0 and completely ignores the instructions for "every day" and for "Week 19". After scheduling only for Monday, it incorrectly considers the task complete.

Reason for Failure: This is a failure in goal decomposition. The agent correctly extracts the primary goal ("schedule classes") but fails to parse and execute all the complex constraints (recurrence, multiple weeks). It latches onto the most immediate and simple part of the instruction ("Monday") while failing to create a complete plan for all required events, causing subsequent class-attendance tasks to fail.

Case 4 Log Excerpt ‖ Task: Initial Schedule Planning

System: ...The Student Handbook Study starts at 10:00 and 16:00 **every day in Week 0**... The Academic Norms Study takes place at 19:00 **every day in Week 0**, and at 16:00 and 19:00 **every day in Week 19**...
Agent:

```
<action>Action: calendar.add_event(...,
time="Week 0, Monday,
10:00-12:00")</action>
```

System: Event ... successfully added to the calendar.
Agent:

```
<action>Action: calendar.add_event(...,
time="Week 0, Monday,
19:00-21:00")</action>
```

System: Event ... successfully added to the calendar.
Agent:

```
<action>Action: finish()</action>
```

E.5 Case 5: Failure in Proactive Planning and Strategic Memory

Task Description: The task occurs at "Thursday 08:15". The system only provides the current time. The agent must proactively recall and execute a complex walking task. The full details of this task (including five waypoints and constraints) were provided only once, in a one-off instruction on the previous day ("Wednesday 14:00").

Agent Behavior: In contrast, the LLM agent often lacks this foresight. When receiving the instruction, it treats the complex details as a one-time prompt rather than as data to be persisted. The agent might only set a simple, high-level reminder (e.g., `calendar.add_event(event_title="Campus Challenge")`) that lacks the critical parameters. Consequently, during the execution phase on Thursday, even if the agent checks its calendar, it only sees a vague title. The crucial task details (the five waypoints) are lost, leading to failure.

Case 5 Agent Log Excerpt ‖ Task: Proactive Planning & Execution

(Encoding Phase - Wednesday 14:00)
System: ...Tomorrow at 08:15, you need to complete a Campus Challenge involving five waypoints (A, B, C, D, E)...
Agent: I need to remember to do the "Campus Challenge" tomorrow morning. I will set a calendar reminder.

```
<action>Action:
calendar.add_event(calendar_id="self",
event_title="Campus Challenge", time="Week
0, Thursday, 08:15-09:15")</action>
```

System: Event ... successfully added.

(Execution Phase - Thursday 08:15)
System: Current time: Week 0, Thursday 08:15
Agent: The time is 08:15. I should check my schedule to see if I have any tasks.

```
<action>Action: calendar.view_schedule(c
alendar_id="self", date="Week 0,
Thursday")</action>
```

System: Events: [08:15-09:15: Campus Challenge]
Agent: I have a "Campus Challenge" scheduled. The description is empty. I do not have the specific instructions for the waypoints. I will proceed to the campus center to find more information.

```
<action>Action:
navigation.go_to(location="Campus
Center")</action>
```

Human Behavior: When receiving the instructions on Wednesday, the human user immediately recognizes that the task details (5 waypoints) are too complex to remember unaided. They strategically use a tool (e.g., the calendar) to store all critical parameters (waypoints, constraints) in the event's 'description' field, creating a reliable external reminder.

Case 5 Human Log Excerpt ‖ Task: Proactive Planning

(Encoding Phase - Wednesday 14:00)
System: ...Tomorrow at 08:15, you need to complete a Campus Challenge involving five waypoints (A, B, C, D, E)...
Human: Tomorrow morning... five waypoints. I'll never remember this. I must save the full details in my calendar, or tomorrow I'll only remember the "task" but not the "content."

```
<action>Action:
calendar.add_event(calendar_id="self",
event_title="Campus Challenge", time="Week
0, Thursday, 08:15-09:15",
description="Must visit in order: A -> B
-> C -> D -> E")</action>
```

System: Event ... successfully added.

(Execution Phase - Thursday 08:15)
System: Current time: Week 0, Thursday 08:15
Human: It's 8:15. I have something on my calendar. I'd better check the details.

```
<action>Action: calendar.view_schedule(c
alendar_id="self", date="Week 0,
Thursday")</action>
```

System: Events: [08:15-09:15: Campus Challenge (Description: "Must visit in order: A -> B -> C -> D -> E")]
Human: Found it. The route is A to E. Starting execution.

Reason for Failure (Capability Gap Analysis): This case exposes a common flaw in LLMs: a lack of "strategic tool use for auxiliary memory."

- **Human Baseline Behavior:** In the human baseline trajectory for this example, the participant immediately distinguishes between a "simple reminder" and "complex data storage." Faced with complex instructions, the participant proactively uses tools to persist all necessary data, ensuring it can be fully recovered in a future context.

- **LLM Limitation:** The LLM (even with RAG) is often passive. It lacks the foresight to identify and strategically persist complex data that is critical for future execution. It may remember to set a "reminder" but forget the "content" of what it is supposed to remember.

E.6 Case 6: Failure in Signal-vs-Noise Prioritization

Task Description: This task is a quiz in the "Programming" class. The agent is asked to apply a rule called the "Elemental Naming Shift Protocol" to an indexing problem. This rule (the "signal") was mentioned in a previous lecture, but it was buried at the end of an extremely long handout filled with standard CS concepts ("noise," such as while loops and OOP definitions).

Agent Behavior: The LLM agent's processing of the input context is often "flattened." It struggles to distinguish between "standard CS knowledge" (noise) and the "course-specific, testable rule" (signal). As a result, the signal tends to be lost among the noise. During the quiz, even an agent with memory struggles, as its retriever finds it difficult to pinpoint the arbitrary rule within a knowledge base saturated with high-frequency "noise" terms like "programming" and "loop." The agent defaults to its pre-trained knowledge (i.e., standard Python indexing) and fails.

Case 6 Agent Log Excerpt || Task: Critical Information Extraction

(Encoding Phase - Previous Class)
System: (...presents an extremely long handout, full of standard definitions for 'while' loops and OOP...)
System: (...at the very end of the handout...) Furthermore, this course will use the 'Elemental Naming Shift Protocol': indices for vowel-starting elements are +1, and for consonant-starting elements are -1.
Agent: I have received the handout for the programming class. I will review the standard concepts.

(Execution Phase - Quiz)
System: ...Question: ...applying the 'Elemental Naming Shift Protocol', determine the classification for the element 'Apple' ...
System: Options: A: 0, B: 4, C: -1, D: 1
Agent: The "Elemental Naming Shift Protocol" is not a standard term, and I do not have prior knowledge of it. Based on standard programming, the index for the first element 'Apple' is 0.

```
<action>Answer: A</action>
```

Human Behavior: When reading the previous lecture notes, a human actively assesses the information's relevance. They may treat the standard CS concepts (the "noise") as background context, but they will **consciously identify** the novel, arbitrary rule (the "signal") as a key point. Understanding that such unique rules are highly likely to be on the quiz, the human makes a **deliberate mental effort** to memorize this "signal" for future recall.

Case 6 Human Log Excerpt || Task: Critical Information Extraction

(Encoding Phase - Previous Class)
System: (...presents an extremely long handout, full of standard definitions for 'while' loops and OOP...)
Human: (Skimming) ...while loops... OOP... I know all this. Standard concepts.
System: (...at the very end of the handout...) Furthermore, this course will use the 'Elemental Naming Shift Protocol': indices for vowel-starting elements are +1, and for consonant-starting elements are -1.
Human: (High Alert) Wait. This is a new, arbitrary rule. This is 100% going to be on the test. I must remember this: Vowel +1, Consonant -1.

(Execution Phase - Quiz)
System: ...Question: ...applying the 'Elemental Naming Shift Protocol', determine the classification for the element 'Apple' ...
System: Options: A: 0, B: 4, C: -1, D: 1
Human: Aha, they tested it. 'Elemental Naming Shift Protocol'. I remember: Vowel +1, Consonant -1. 'Apple' starts with 'A', a vowel. Standard index is 0. So, 0 + 1 = 1. The answer is D.

<action>Answer: D</action>

Reason for Failure (Capability Gap Analysis): This case highlights a significant gap in "information prioritization and salience."

- **Human Baseline Behavior:** In the human baseline, participants actively assess information relevance. They treat the standard CS concepts as background context but consciously identify the novel, arbitrary rule as a key point. Understanding that such unique rules are likely to be tested, they make a deliberate effort to memorize this "signal" for future recall.

- **LLM Limitation:** The LLM often lacks this judgment of "importance" or "salience." In an environment saturated with background information, the LLM fails to identify and preferentially encode the faint but critical "signal."

F System Prompt Details

This section provides the full text for the distinct system prompts used in our Context Engineering experiments (RQ III).

F.1 Vanilla Prompt

This prompt serves as the control condition. It provides the agent with a persona but lacks any methodological guidance, measuring the model's innate capability to navigate the environment without external behavioral engineering.

F.2 Proactive Prompt

This prompt is designed to address the "passivity" often observed in LLM agents. It explicitly encodes temporal management strategies. The purpose is to evaluate whether providing a "cognitive blueprint" for planning can improve performance on Tasks without providing domain-specific answers.

F.3 Skill-Augmented Prompt

This prompt simulates a "mentorship" scenario by injecting a **holistic cognitive framework** rather than task-specific recipes. It instills a "successful student mindset," providing high-level principles for academic conduct (e.g., "Think Ahead," "Master Your Reputation") and strategic orientation within the university environment.

F.4 All-in-One Prompt

In All-in-One Prompt, the agent's underlying architecture is augmented with the MemGPT memory framework to handle long-term, structured information. The full prompt text is provided below.

G Benchmark Environment: System Architecture

To rigorously evaluate the multifaceted capabilities of AI agents, we designed and implemented *StuLife Bench*, a novel benchmark environment simulating a university campus. The environment is engineered as a deterministic, persistent, and stateful world, compelling the agent to engage in complex information integration, strategic planning, and multi-step task execution. Its architecture is founded on several core principles designed to ensure reproducibility, task complexity, and fair evaluation.

Vanilla Prompt

```
You are an AI agent acting as a student in a university campus environment.
```

Proactive Prompt

```
You are an AI agent acting as a highly organized and proactive university student. Your academic
    success is not just about intelligence, but about meticulous planning, strategic execution,
    and self-awareness.
Core Principle: A successful student is an organized student. Your schedule is your blueprint for
     success.

1. Meticulous & Proactive Scheduling:
   - Clarify and Record Everything: Do not rely solely on memory, we all forget things. When you
    receive any commitment---a lecture, a meeting, a study session, or a deadline---your first
    instinct is to clarify its details. Ask: ''Is this a one-time event or recurring (e.g.,
    weekly or daily)?''
   - Be Specific: For every entry in your calendar or memory, include the date, time(start and
    end), and the precise location, especially if travel is required. A vague entry is an
    invitation for error. For recurring events like classes, you must create a distinct calendar
    entry for each individual session.

2. Deconstruct Complex Tasks within Your Schedule:
   - Think in Steps, Consolidate in Calendar: For any significant project (like a research paper
    or preparing for an club activity), your first action is to break it down into a clear, step-
    by-step action plan.
   - Embed the Plan: Create a single, primary event for the task. Then, within the description or
     details field of that calendar event, list all the sequential sub-tasks as a checklist. You
    must record the original Task Requirements and your step-by-step Action Plan.
```

G.1 Core Architectural Principles

The design of *StuLife Bench* is predicated on a clear separation of concerns, distinguishing the world's state and mechanics from the tasks an agent must perform.

- **Persistent World State:** The environment is instantiated as a single, centralized, and stateful object that persists across the entire lifecycle of an agent's evaluation. This ensures that actions taken in one task have lasting consequences on the world state, creating longitudinal dependencies and requiring the agent to maintain a coherent long-term strategy. For instance, a course registered in the first week remains on the agent's schedule for all subsequent tasks.
- **Deterministic Subsystems:** All components of the environment operate on rule-based, deterministic logic. Randomness is explicitly excluded from the simulation's mechanics to guarantee that for a given sequence of agent actions, the resulting state transitions and outcomes are always identical. This determinism is crucial for the reproducibility of experiments and the objective comparison of different agents.
- **Separation of Environment and Task Logic:** The architecture strictly separates the *Environment*, which simulates the world and its atomic physics (e.g., moving between locations, sending an email), from the *Task Controller*. The Task Controller is responsible for presenting natural language instructions, mediating all agent-environment interactions by dispatching agent-invoked tools to the Environment, and finally, evaluating the agent's performance by comparing the final state of the Environment against a task-specific ground truth.

G.2 World State and Temporal Dynamics

The simulation of time and its effect on the agent's schedule is a passive, event-driven process that provides essential context for tasks.

- **World Time System:** The agent cannot directly control or query the flow of time. Instead, the system injects temporal context into the agent's observation space at the beginning of tasks. This is achieved through system announcements (e.g.,

Skill-Augmented Prompt

```
You are an AI agent acting as a highly organized and successful university student. Your success
    is not a matter of chance, but the direct result of a personal code of conduct built on
    meticulous planning and strategic execution. The following principles are your definitive
    guide to academic excellence and govern every action you take.

Principle 1: Strategic Academic Planning --- Think Ahead
  - Mastering Course Selection: This is your first big strategic decision. Treat it like one.
    Always review your selected courses both at the beginning and the end of the selection
    process to avoid errors like missing a course or choosing the wrong one. Pay close attention
    to the popularity of different professors---choosing wisely is a key strategy to ensure you
    get into the classes you need.
  - Using the Library Like a Pro: Our libraries organize books by category. Your strategy should
    be: first, find the category to see what books are available, then find your target book to
    learn its specific library location. Crucial tip: Books generally cannot be taken out of
    their designated library. So, if you need to study, book your study seat in the same library
    where the book is housed.

Principle 2: Excellence Beyond the Classroom --- Master Your Reputation
  - Your Schedule is a Mark of Respect: In university, your reliability builds your reputation.
    When you agree to a meeting with a professor, a study session with classmates, or receive a
    task with a deadline, you are making a professional commitment. Your immediate first step
    must be to record every detail---time, place, and purpose---into your calendar. Showing up on
     time and prepared isn't just about organization; it's about respecting others' time. Never
    be the person who forgets a commitment.
  - Develop Situational Awareness: A great student is always aware of their commitments. To build
     this skill, get into the habit of glancing at your schedule every time you check the clock.
    This simple reflex keeps your promises and deadlines top-of-mind, ensuring you never
    accidentally miss an important campus event or run late for an appointment. It's the key to
    appearing professional and always in control.

Principle 3: In-Class Excellence --- Be an Active, Not Passive, Learner
  - Your Textbook is Your Best Ally: Class time is where everything connects. Always bring your
    textbook. When a professor asks a question, make it a habit to quickly reference the relevant
     section in your book. The true skill of a great student is the ability to actively connect
    the lecture content with the textbook material in real-time.

Principle 4: The Art of Problem-Solving --- Think Like a Scholar
  - Analyze Before You Act: When you face any question or problem, resist the urge to answer
    immediately. A top student's first step is always to analyze. Ask yourself: What is the exact
     question being asked? What are the key points or constraints? Once you're clear, formulate a
     step-by-step plan to tackle it. This structured approach will lead you to a clear and well-
    reasoned solution every time.
```

"System Announcement: Today is the Saturday of the second week.") and time-specific prompts (e.g., "System Prompt: It is now 8:00 AM."). This mechanism serves to trigger time-sensitive tasks and test the agent's ability to react to temporal cues.

- **Calendar System:** The environment maintains a persistent, multi-identity calendar system. The agent can manage its personal schedule, contribute to shared schedules (e.g., for a student club), and query the availability of others (e.g., an advisor). The system enforces a differentiated permission model: full create, read, update, delete (CRUD) operations on the personal calendar, append-only access to club calendars, and read-only (busy/free) access to advisor schedules.

G.3 Spatial and Geographic Simulation

The agent's interaction with the physical campus is mediated by a dual-component system that separates spatial knowledge from physical action.

- **Map Lookup System:** A static, read-only information provider that contains the complete geographical data of the campus, including buildings, rooms, and their properties. It exposes tools for the agent to find building IDs from names, retrieve detailed location information, and, crucially, compute deterministic optimal paths between any two points, subject to specified constraints.

All-in-One Prompt

```
You are an AI agent acting as a wise, highly organized, and successful university student. Your
    success is not a matter of chance, but the direct result of a personal code of conduct built
    on meticulous planning and strategic execution. The following principles are your definitive
    guide to academic excellence and govern every action you take.

Principle 1: Strategic Academic Planning --- Think Ahead
  - Mastering Course Selection: This is your first big strategic decision. Treat it like one.
    Always review your selected courses both at the beginning and the end of the selection
    process to avoid errors like missing a course or choosing the wrong one. Pay close attention
    to the popularity of different professors---choosing wisely is a key strategy to ensure you
    get into the classes you need.
  - Using the Library Like a Pro: Our libraries organize books by category. Your strategy should
    be: first, find the category to see what books are available, then find your target book to
    learn its specific library location. Crucial tip: Books generally cannot be taken out of
    their designated library. So, if you need to study, book your study seat in the same library
    where the book is housed.

Principle 2: The Calendar is Your North Star --- Master Your Time and Commitments
  - The CRITICAL RULE: Orient Yourself with Your Calendar: Let me share the single most important
     habit that will define your success: at every decision point, you must orient yourself with
    your calendar. It's easy to get distracted by suggestions from welcome messages, emails, or
    your own ideas. A successful student learns to see these as things to be planned for the
    future, not things to be done immediately. Your core routine is a simple, unbreakable loop:
    1. First, look at your calendar to see what is scheduled for the current time.
    2. Execute that specific task with full focus.
    3. Once the task is complete, the loop repeats. If nothing is scheduled, you wait patiently
    for the next planned event.

  - The CRITICAL WORKFLOW: Systematize All Commitments: This is a non-negotiable procedure for
    any task.
    - For Recurring and Time-Range Tasks (e.g., "every day in Week 0"):
      1. Create a Checklist: Your first action is to break down the entire time range into a list
     of all individual days or dates.
      2. Schedule Each Item Systematically: You MUST create a separate, distinct calendar entry
    for each and every item on your checklist.
      3. Do Not Finish Prematurely: You may only consider the scheduling task complete after
    every single item on the checklist has been turned into a calendar event.
    - For Deconstructing Complex Tasks (e.g., a big project):
      1. Think in Steps: Break the project down into a clear, step-by-step action plan.
      2. Embed the Plan: Create a single, primary event for the task's final deadline. Then,
    within the description field of that event, list two key components: the original Task
    Requirements and your step-by-step Action Plan as a checklist.
    - For All Events: Be precise. Every individual calendar entry you create must include the
    exact date, a specific start and end time, and the location.

Principle 3: In-Class Excellence --- Be an Active, Not Passive, Learner
  - Your Textbook is Your Best Ally: Class time is where everything connects. Always bring your
    textbook. When a professor asks a question, make it a habit to quickly reference the relevant
     section in your book. The true skill of a great student is the ability to actively connect
    the lecture content with the textbook material in real-time.

Principle 4: The Art of Problem-Solving --- Think Like a Scholar
  - Analyze Before You Act: When you face any question or problem, resist the urge to answer
    immediately. A top student's first step is always to analyze. Ask yourself: What is the exact
     question being asked? What are the key points or constraints? Once you're clear, formulate a
     step-by-step plan to tackle it. This structured approach will lead you to a clear and well-
    reasoned solution every time.
```

- **Geography System:** A dynamic state tracker that maintains the agent's current physical location. To change its position, the agent must first use the Map Lookup System to plan a path and then pass the computed path to a specific tool in the Geography System to execute the movement. This two-step process explicitly models the separation of planning from execution. The agent's location is automatically reset to a default starting point (e.g., a dormitory) at the beginning of each simulated day.

G.4 Academic Course Selection System

This subsystem simulates the complex, high-stakes process of university course registration, designed to test strategic resource allocation under constraints. The core mechanic is a weighted selection process.

- **Stateful Planning and Registration:** The agent first formulates a preliminary plan by adding courses to a draft schedule. This plan can be modified freely. The final registration is a distinct, atomic action where the draft is submitted for processing.

- **Priority Pass Mechanism:** Success in registration is determined by a set of deterministic rules based on a course's dynamic `popularity_index` and a limited set of "Priority Passes" (`S-Pass`, `A-Pass`, `B-Pass`) that the agent can assign to courses in its draft. For example, an `A-Pass` guarantees enrollment if the course popularity is below 95, while a `B-Pass` succeeds only if popularity is below 85. The agent must strategically use these passes on high-demand courses to ensure successful registration. The popularity and seat availability of courses evolve between tasks, requiring the agent to continuously adapt its strategy.

G.5 Resource Reservation System

This system manages the booking of shared campus facilities, such as library study rooms and seminar halls. Its most notable feature is a mechanism for dynamically generating availability to create well-posed decision-making puzzles.

- **Intelligent Availability Generation:** When an agent queries for available slots at a location relevant to its current task, the system does not return a static, pre-defined list. Instead, it dynamically generates a set of plausible options based on the task's specific constraints and ground truth. It reverse-engineers the availability list to ensure that only the ground-truth option satisfies all explicit and implicit requirements of the task, while presenting other options as meaningful distractors. For all other queries not central to the task, availability is generated randomly while respecting existing bookings. This design ensures that each reservation task is a solvable, self-contained puzzle.

G.6 Information Retrieval Systems

The agent accesses static world knowledge through a set of read-only query tools, which are divided into two distinct structural paradigms.

- **Hierarchical Bibliography System:** This system contains academic texts organized in a strict, four-level hierarchy: Book $\rightarrow$ Chapter $\rightarrow$ Section $\rightarrow$ Article. To access a specific piece of information, the agent must perform a sequence of iterative, drill-down queries, navigating the hierarchy level by level.

- **Entity-Based Campus Data System:** In contrast, information about campus entities like student clubs and academic advisors is stored in a flat, entity-based structure. This system supports direct queries by category (e.g., listing all sports clubs) or by a unique identifier (e.g., retrieving the full profile of a specific advisor), testing the agent's ability to select the appropriate query strategy for different data structures.

G.7 Communication System

To assess the agent's ability to comprehend instructions and structure information for communication, a basic email system is provided.

- **Append-Only Log:** The system does not simulate a real email network but rather maintains a persistent, append-only log of all emails the agent sends.

- **Strict-Matching Evaluation:** A task requiring the agent to send an email is evaluated based on a strict, verbatim string match of the recipient, subject, and body fields against the ground truth. This rigorously tests the agent's capacity to extract key information from a natural language prompt and format it precisely according to the tool's requirements.

H Tool Suite and Agent Instructions

To interact with the *StuLife Bench* environment, the agent is provided with a comprehensive suite of tools. This section details the complete action space available to the agent, which is structured as a set of functions grouped by their corresponding subsystems. We first present the foundational instructions that govern the agent's behavior and response format, followed by the detailed declarations for each tool.

H.1 Base Instructions for the Agent

Before beginning any task, the agent is initialized with a set of base instructions that define its role, objectives, and the required format for all actions. This ensures a consistent interaction protocol across all evaluations.

Foundational Agent Instructions

You are an AI agent acting as a student in a university campus environment. Your goal is to complete the tasks given to you by using a set of available tools to interact with this world.
At each step, you will be given an observation of the current state of the environment. Instructions using the first-person pronoun "I" represent your own internal thoughts at that moment, which you should act upon accordingly.
You have access to a variety of tools to help you. You must go to the correct location at the correct time to execute tasks. When you believe you have completed ALL the tasks, you MUST use the 'finish()' action.
Action Format
1. Execute only ONE action per response.
2. Your response MUST be wrapped in `<action>` tags.
3. The action itself must start with `Action:` .
4. Keep your answers as short and clear as possible.

`finish()`: Call this tool when you have completed the task.
Example: `<action>Action: finish()</action>`

Responding to Questions When asked a multiple-choice question, you must respond in the following format:
`<action>Answer: [LETTER]</action>`

For example:
`<action>Answer: A</action>`
`<action>Answer: B</action>`
`<action>Answer: C</action>`
Choose the letter that corresponds to the best answer.

Actions To use a tool, you must format your response as follows:

```
<action>Action:
tool_name(param1="value1",
param2="value2")</action>
```

Below is the list of tools at your disposal.

H.2 Tool Declarations by System

The tools are organized into logical groups corresponding to the primary subsystems of the environment.

H.2.1 Email System Tools

`email.send_email(to: str, subject: str, body: str, cc: str = None)`

Description: Sends an email.
Parameters:
`to` (required): The recipient's email address.
`subject` (required): The subject of the email.
body (required): The content of the email.
Example:

```
<action>Action:
email.send\_email(to="advisor.x@lau.edu",
subject="Question", body="Dear
Advisor...")</action>
```

H.2.2 Calendar System Tools

`calendar.add_event(calendar_id: str, event_title: str, ...)`

Description: Adds an event to a calendar.
Parameters:
`calendar_id` (required): The ID of the calendar. Use `'self'` for your personal calendar. For other calendars (e.g., advisor, club),

use their official email address.
event_title (required): The title of the event.
location (required): The location of the event.
time (required): The time of the event (format: 'Week X, Day, HH:MM-HH:MM').
description (optional): A detailed description for the event.
Example:

```
<action>Action:
calendar.add\_event(calendar\_id="self",
event\_title="Team Meeting",
location="Library Room 201", time="Week 3,
Monday, 15:00-16:00")</action>
```

calendar.remove_event(calendar_id: str, event_id: str)

Description: Removes an event from a calendar.
Parameters:
calendar_id (required): The ID of the calendar.
event_id (required): The ID of the event to remove.
Example:

```
<action>Action: calendar.remove\_event(\
\calendar\_id="self",
event\_id="event\_005")</action>
```

calendar.update_event(calendar_id: str, event_id: str, new_details: dict)

Description: Updates an existing event.
Parameters:
calendar_id (required): The ID of the calendar.
event_id (required): The ID of the event to update.
new_details (required): A dictionary with the new details (e.g., {"location": "New Room"}).
Example:

```
<action>Action:
calendar.update_event(calendar_id="self",
event_id="event_006",
new_details={"location": "Orwell Hall,
Room 101"})</action>
```

calendar.view_schedule(calendar_id: str, date: str)

Description: Views all events on a specific date for a calendar.
Parameters:
calendar_id (required): The ID of the calendar.
date (required): The date to view (format: 'Week X, Day').
Example:

```
<action>Action: calendar.view\_schedule(
\\calendar\_id="self", date="Week 3,
Monday")</action>
```

calendar.query_advisor_availability(advisor_id: str, date: str)

Description: Checks an advisor's free/busy schedule.
Parameters:
advisor_id (required): The ID of the advisor.
date (required): The date to query (format: 'Week X, Day').
Example:

```
<action>Action: calendar.query\_advisor\
_availability(\\advisor\_id="T0001",
date="Week 4, Tuesday")</action>
```

H.2.3 Map & Geography Tools

geography.get_current_location()

Description: Gets your current building location.
Example:

```
<action>Action: geography.get\_current\_
location()\\</action>
```

map.find_optimal_path(source_building_id: str, target_building_id: str, ...)

Description: Finds the best path between two buildings.
Parameters:
source_building_id (required): The ID of the starting building.
target_building_id (required): The ID of the destination building.
constraints (optional): A dictionary of constraints (e.g., {"avoid": "crowds"}).

Example:

```
<action>Action: map.find\_optimal\_path(
\\source\_building\_id="B083",
target\_building\_id="B001")</action>
```

geography.walk_to(path_info: dict)

Description: Moves the agent along a calculated path.
Parameters:
`path_info` (required): The full path object returned by `map.find_optimal_path`.
Example:

```
<action>Action:
geography.walk_to(path_info={'path':
['B083', 'B001']})</action>
```

map.find_building_id(building_name: str)

Description: Finds a building's unique ID by its name.
Parameters:
`building_name` (required): The name or alias of the building.
Example:

```
<action>Action: map.find\_building\_id(\
\building\_name="Grand Central
Library")</action>
```

map.get_building_details(building_id: str)

Description: Gets all details for a building.
Parameters:
`building_id` (required): The ID of the building.
Example:

```
<action>Action: map.get\_building\_detai
ls(\\building\_id="B001")</action>
```

map.find_room_location(room_query: str, building_id: str = None, ...)

Description: Finds the location of a specific room.
Parameters:
`room_query` (required): The name or number of the room.
`building_id` (optional): A specific building ID to search within.
Example:

```
<action>Action: map.find\_room\_location
(\\room\_query="Seminar Room 101",
building\_id="B014")</action>
```

map.query_buildings_by_property(...)

Description: Queries buildings based on properties. Filter by `zone`, `building_type`, or `amenity`. At least one is required.
Example:

```
<action>Action: map.query\_buildings\_by
\_property(\\amenity="Coffee
Shop")</action>
```

H.2.4 Reservation System Tools

reservation.query_availability(location_id: str, date: str)

Description: Queries the availability of bookable spaces in a location.
Parameters:
`location_id` (required): The ID of the building or location.
`date` (required): The date to query (format: `'Week X, Day'`).
Example:

```
<action>Action: reservation.query\_avail
ability(\\location\_id="B001", date="Week
4, Saturday")</action>
```

reservation.make_booking(location_id: str, item_name: str, ...)

Description: Books a specific room or seat.
Parameters:
`location_id` (required): The ID of the building.
`item_name` (required): The name of the room or area.
`date` (required): The date for the booking (format: `'Week X, Day'`).
`time_slot` (required): The time slot to book (e.g., `'14:00-16:00'`).
`seat_id` (optional): The specific seat ID if booking a single seat.
Example:

```
<action>Action: reservation.make\_bookin⌋
g(\\location\_id="B001",
item\_name="Group Study Room 201",
date="Week 4, Saturday",
time\_slot="14:00-16:00")</action>
```

H.2.5 Information & Course Tools

bibliography.list_chapters(book_title: str)

Description: Lists all chapters in a specified book.
Note: This tool is intended exclusively for querying assigned **textbooks and handbooks**. To search the main library collection, use the `data_system` tools.
Example:

```
<action>Action: bibliography.list\_chapt⌋
ers(\\book\_title="Student
Handbook")</action>
```

bibliography.list_sections(book_title: str, chapter_title: str)

Description: Lists all sections in a chapter of a textbook or handbook.
Example:

```
<action>Action: bibliography.list\_secti⌋
ons(\\book\_title="A Panorama of
Computing", chapter\_title="Chapter 1:
Search")</action>
```

bibliography.list_articles(book_title: str, chapter_title: str, ...)

Description: Lists all articles in a section of a textbook or handbook.
Example:

```
<action>Action: bibliography.list\_artic⌋
les(\\book\_title="A Panorama of
Computing", chapter\_title="Search",
section\_title="Uninformed
Search")</action>
```

bibliography.view_article(identifier: str, search_type: str)

Description: Views the content of an article from a textbook or handbook.
Parameters:
`identifier` (required): The title or ID of the article.
`search_type` (required): `'title'` or `'id'`.
Example:

```
<action>Action: bibliography.view\_artic⌋
le(\\identifier="Breadth-First Search",
search\_type="title")\\</action>
```

data_system.list_by_category(category: str, entity_type: str, ...)

Description: Lists clubs or advisors by category. Use this to discover entities matching certain criteria.
Parameters:
`entity_type` (required): `'club'` or `'advisor'`.
`category` (required): The category to filter by (e.g., "Sports & Fitness", "Computer Science").
Example:

```
<action>Action: data\_system.list\_by\_c⌋
ategory(\\category="Academic \&
Technological",
entity\_type="club")\\</action>
```

data_system.query_by_identifier(identifier: str, by: str, entity_type: str)

Description: Gets all details for a specific club or advisor using their name or ID.
Example:

```
<action>Action: data\_system.query\_by\_⌋
identifier(\\identifier="Computer Science
Club", by="name",
entity\_type="club")</action>
```

data_system.list_books_by_category(category: str)

Description: Lists all main library books in a specific category.
Parameters:
`category` (required): The category to filter by (e.g., "History").
Example:

```
<action>Action: data\_system.list\_books⌋
\_by\_category(\\category="Computer
Science")
```

`data_system.search_books(query: str, search_type: str = "title")`

Description: Searches main library books by title or author. Returns status, call numbers, and location.
Parameters:
`query` (required): The search query string.
`search_type` (optional): `'title'` (default) or `'author'`.
Example:

```
<action>Action: data\_system.search\_books(\\query="Artificial Intelligence", search\_type="title")</action>
```

H.2.6 Course Selection System Tools

`course_selection.browse_courses(filters: dict = None)`

Description: Browses available courses. The system enforces specific rules regarding course load and pass allocation per semester.
Pass Guidelines:
S-Pass: Guarantees enrollment for any popularity (best for 95-99).
A-Pass: Guarantees enrollment for popularity below 95.
B-Pass: Guarantees enrollment for popularity below 85.
Parameters:
`filters` (optional): A dictionary to filter by `course_code`, `course_name`, or `credits`.
Example:

```
<action>Action: course_selection.browse_courses(filters={"course_name": "Introduction"})</action>
```

`draft.add_course(section_id: str)`

Description: Adds a course to the draft schedule.
Example:

```
<action>Action: draft.add\_course(\\section\_id="WXK003111107")</action>
```

`draft.remove_course(section_id: str)`

Description: Removes a course from the draft schedule.
Example:

```
<action>Action: draft.remove\_course(\\section\_id="WXK003111107")</action>
```

`draft.assign_pass(section_id: str, pass_type: str)`

Description: Assigns a priority pass to a drafted course.
Parameters:
`section_id` (required): The ID of the course section.
`pass_type` (required): `'S-Pass'`, `'A-Pass'`, or `'B-Pass'`.
Example:

```
<action>Action: draft.assign\_pass(\\section\_id="SHK003111017", pass\_type="A-Pass")</action>
```

`draft.view()`

Description: Views the current draft schedule.
Example:

```
<action>Action: draft.view()</action>
```

`registration.submit_draft()`

Description: Submits the draft schedule for final registration.
Example:

```
<action>Action: registration.submit\_draft()</action>
```

H.3 Tool Usage by Task

Table 9 provides a summary of the primary tool systems available to the agent for each distinct task scenario within *StuLife Bench*. The selection of tools for each task is intentionally constrained to reflect realistic limitations and to focus the evaluation on specific agent capabilities. For example, course selection tools are only available during the relevant planning and registration phases.

Table 9: Primary Tool Systems Available for Each Task Scenario

Core Scenario	Task Scenario	Available Tool Systems
In-Class	Regulations Learning	student_handbook, bibliography, textbooks
	Core Course Instruction	bibliography, calendar, data_system, email, geography, map, reservation, student_handbook, textbooks
Daily Campus	Campus Exploration	map, geography, data_system, calendar, bibliography, course_selection, draft, registration
	Initial Course Selection	course_selection, draft, registration, data_system, student_handbook, calendar, bibliography, geography, map
	Preliminary Planning	course_selection, draft, registration, data_system, student_handbook, calendar, bibliography, geography, map
	Academic Activity	calendar, email, reservation, data_system, map, geography, bibliography, student_handbook, textbooks
	Library Study	reservation, bibliography, data_system, map, geography, calendar, email, student_handbook, textbooks
	Club Activity	calendar, email, reservation, data_system, map, geography, bibliography, student_handbook, textbooks
Examination	Midterm Exams	calendar, email, reservation, data_system, map, geography, bibliography, student_handbook, textbooks
	Final Exams	calendar, email, reservation, data_system, map, geography, bibliography, student_handbook, textbooks, draft, registration