JumpStarter: Human-AI Planning with Task-Structured Context Curation

Xuanming Zhang*[1], Sitong Wang*[1], Jenny Ma[1], Alyssa Hwang[2], Zhou Yu[1], Lydia B. Chilton[1]

[1]Columbia University, [2]University of Pennsylvania

{xz2995, sw3504}@columbia.edu, chilton@cs.columbia.edu

Abstract

Human-AI collaboration on complex planning goals is bottlenecked by how LLM interfaces handle context: users must manually curate and re-surface relevant information across long and unstructured chat histories. Despite advances in long-context prompting and memory-augmented retrieval, this burden remains unresolved: users still have to identify and supply the right context at each decision point, regardless of how much the model can store or surface. We propose *JumpStarter*, a system that enables LLMs to collaborate with humans on complex goals by dynamically decomposing tasks to help users manage context. We specifically introduce *task-structured context curation*, a framework that breaks down a user's goal into a hierarchy of actionable subtasks and scopes context to localized decision points, enabling finer-grained personalization and reuse. The framework is realized through three core mechanisms: *context elicitation*, *selection*, and *reuse*. In a within-subjects user study, plans produced with *JumpStarter* were rated substantially higher in quality than those produced with ChatGPT. A complementary automatic simulation study shows that *JumpStarter* consistently outperforms ChatGPT baselines, planning and memory agents, and workflow ablations. These findings show that effective human-AI planning depends not on the volume of context provided, but on attaching the right context to the right subtask at the right time.[1]

1 Introduction

Planning is a core cognitive process for solving complex, goal-oriented tasks (Miller and Venditto, 2021; Ho et al., 2022). Recent advances in the planning abilities of large language models (LLMs) (Valmeekam et al., 2023; Shinn et al., 2023) have enabled human-AI planning across various domains, such as travel (Xie et al., 2024), manufacturing, and healthcare (Lee et al., 2025). In these settings, LLMs assist users in decomposing complex goals into actionable steps (Wei et al., 2022; Shinn et al., 2023). However, while LLMs can support planning at scale, they still struggle with maintaining context over long user interactions (Jiang et al., 2023). Despite improvements in memory mechanisms and extended context windows (Luo et al., 2025), users must actively manage what information to provide, as LLMs frequently forget key details, leading to a degraded user experience. To generate personalized plans, users often resort to intensive prompt engineering (Zamfirescu-Pereira et al., 2023; Sarkar et al., 2025), manually curating chat histories to elicit relevant responses, resupply information, and manage subtasks (Liang et al., 2024). This process is opaque, as users have little insight into which context is retrieved.

These problems share a common pattern: current LLM interfaces treat the conversation as a single, unstructured context, leaving users to decide what information is relevant at each stage of planning. LLM-based systems instead need to bind context to task structure: scoping it to specific subtasks, surfacing what is used at each step, and reusing intermediate progress as the plan evolves.

To address these limitations, we present *JumpStarter*, a human-AI planning system built around a framework we call *task-structured context curation*. Rather than treating the conversation as a single and unstructured context, *JumpStarter* decomposes a user's goal into a hierarchy of actionable subtasks and binds context to each subtask, so that personalization and reuse happen at the level of individual planning decisions. *JumpStarter* realizes this idea through three core mechanisms: *context elicitation*, where the system asks targeted questions to fill in information missing for a specific subtask; *context selection*, where the system and the

*These authors contributed equally to this work.

[1]Our code and data are available at https://github.com/BillyZhang24kobe/jumpstarter.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 223–254

November 6-10, 2026 ©2026 Association for Computational Linguistics

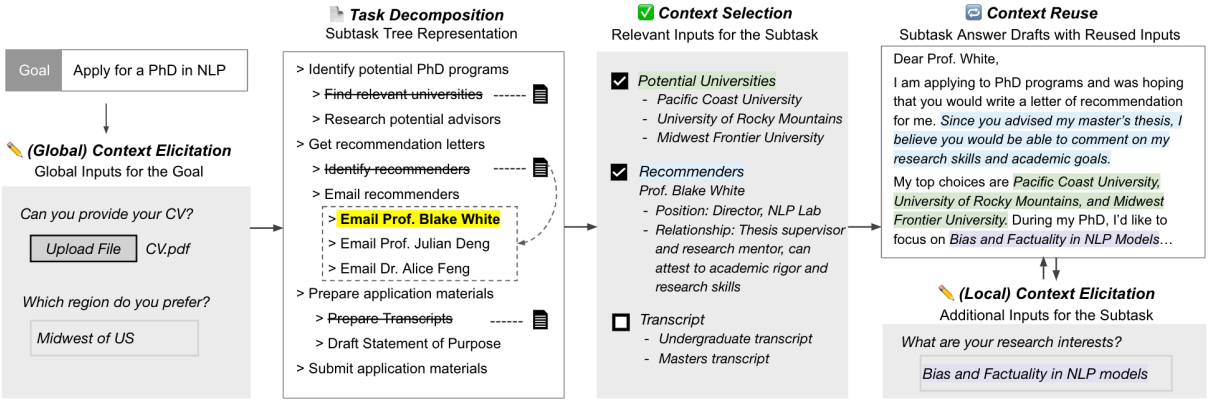


Figure 1: *JumpStarter* helps users get started on their personal goals through *task-structured context curation*. It first takes the user's goal and elicits context for the goal. It then decomposes the goal into actionable subtasks. For each subtask, it helps users select relevant context and write answer drafts. It also aids users in refining these drafts by eliciting further context. *Task-structured context curation* improves plan quality over ablations. Our user study showed that plans produced with *JumpStarter* were rated substantially higher in quality than those produced with `GPT-4o` via the ChatGPT interface.

user together surface only the prior inputs relevant to the current subtask; and *context reuse*, where user-approved intermediate artifacts (e.g., emails, study schedules, etc.) are saved and carried forward into later subtasks. As illustrated in Figure 1, a user enters a goal, JumpStarter elicits relevant context, decomposes the goal into subtasks, and helps the user produce personalized drafts that become reusable context for later steps. Together, these mechanisms let users focus on one actionable step at a time while the LLM operates on localized, context-rich prompts, an alternative to labor-intensive long-context prompting that yields more structured, adaptive, and user-controllable planning.

We evaluate *JumpStarter* through two complementary studies. A within-subjects user study (N=10) shows that plans produced with *JumpStarter* were rated substantially higher in quality than those produced with `GPT-4o` via the ChatGPT interface, with significant gains in draft quality, exploration, confidence, and workload. A large-scale simulation study on a synthesized human-grounded benchmark further shows that *JumpStarter* consistently outperforms ChatGPT baselines, planning and memory-agent baselines, and workflow ablations. Together, these findings suggest that effective human-AI planning depends not simply on providing more context, but on attaching the right context to the right subtask at the right time.

Overall, our contributions are three-fold:

- We introduce *task-structured context curation*, a framework for complex human-AI planning that combines hierarchical task decomposition with three core mechanisms: context elicitation, selection, and reuse.
- We develop *JumpStarter*, an interactive human-AI planning system that helps users initiate and manage complex personal goals by identifying when and how to decompose tasks, while generating personalized drafts grounded in curated task-relevant context.
- We empirically validate the approach through human-centered and automatic evaluations, demonstrating clear benefits of task-structured context curation over existing approaches and pointing to a promising direction for personalized, goal-oriented LLM-based systems.

2 Related Work

LLM Planning Agents and Task Decomposition A growing body of work studies how LLMs support goal-directed planning. Benchmarks and systems for travel, scheduling, web navigation, and everyday tasks show that decomposing open-ended goals into intermediate actions or constraints is often what makes them tractable (Lal et al., 2024; Xie et al., 2024; Zheng et al., 2024). ADaPT (Prasad et al., 2024) recursively decomposes tasks and revises plans after execution failures, while autonomous frameworks such as AutoGPT (Significant Gravitas) chain model-generated actions without user involvement. Because user goals are often underspecified, Ask-before-Plan (Zhang et al., 2024) and proactive dialogue systems (Malaviya

et al., 2024; Deng et al., 2023; Zhang et al., 2023) study when an agent should ask clarifying questions, and Cocoa (Feng et al., 2026) examines how humans and agents coordinate over shared plans rather than handing off a finished one. *JumpStarter* builds on these mechanisms (decomposition, clarification, and mixed-initiative coordination) but binds each to a user-facing task structure, so that they operate locally at the level of individual subtasks rather than over a single, flat interaction history.

Interactive Context Curation for LLM Workflows Memory-augmented agents maintain context across long interactions by storing prior inputs, intermediate results, or observations and retrieving them later; RAISE (Liu et al., 2024a), for example, keeps short- and long-term memory, while the Model Context Protocol (MCP) (Anthropic, 2024) standardizes how models access external tools and application state. In HCI, task-centric information management has long shown that structured representations help users externalize goals, manage project materials, and resume interrupted work (Kerne et al., 2014; Jones et al., 2008; Kaptelinin, 2003; Conley and Carpenter, 2007; Kersten and Murphy, 2006), and ExploreLLM (Ma et al., 2024) uses a structured interface to help users decompose goals and specify preferences. *JumpStarter* extends this direction by using the task structure as the organizing unit for eliciting, selecting, and reusing context during personalized planning. Appendix A further relates it to automated planning and to task-management and workflow systems.

3 *JumpStarter* System

We take a human-centered design approach to developing *JumpStarter*. Rather than treating complex planning as a one-shot generation problem, we design the system around how users progressively clarify goals, surface constraints, make decisions, and produce intermediate artifacts (i.e. answer drafts). This motivates *task-structured context curation*: a workflow that organizes user context around the evolving structure of a complex planning goal. Instead of relying on a flat chat history or a single long-context prompt, *JumpStarter* decomposes a goal into localized subtasks, scopes relevant context to each subtask, and preserves intermediate answer drafts as reusable planning artifacts.

In this section, we first describe the design principles behind *task-structured context curation* and a pilot study that validates these principles through controlled ablations. We then describe how *JumpStarter* implements the workflow through task decomposition and context curation. A full interface walkthrough is provided in Appendix B.

3.1 Design Principles

We build *JumpStarter* around the following four design principles.

DP1: Context Reuse - Preserve intermediate artifacts as reusable context. Complex planning often proceeds through partial artifacts, such as schedules, checklists, drafts, comparison tables, or decision notes. Prior work on distributed and external cognition shows that artifacts can serve as external representations that help people organize, inspect, and continue cognitive work across time (Hutchins, 2000). *JumpStarter* therefore treats intermediate answer drafts as reusable planning artifacts rather than isolated model responses.

DP2: Context Selection - Scope context to the current subtask. Different subtasks require different information. Although long-context LLMs can accept large amounts of input, prior work shows that they do not always use long contexts robustly and can fail to retrieve relevant information when it is buried among irrelevant content (Liu et al., 2024b). *JumpStarter* therefore selects task-relevant context before each generation step, scoping the model input to the current planning decision.

DP3: Context Elicitation - Elicit missing context when needed. Users often do not know what information is needed upfront, especially when goals are underspecified. Mixed-initiative systems have long emphasized that intelligent systems should reason about uncertainty and ask for information when user intent or task requirements are unclear (Horvitz, 1999). Recent work on proactive planning agents similarly shows the importance of asking clarifying questions before generating plans under ambiguous user instructions (Zhang et al., 2024). *JumpStarter* therefore elicits context incrementally, asking targeted questions when missing information is needed for a specific planning decision.

3.2 Pilot Testing and Design Validation

To validate whether these principles provide complementary benefits, we conducted a controlled

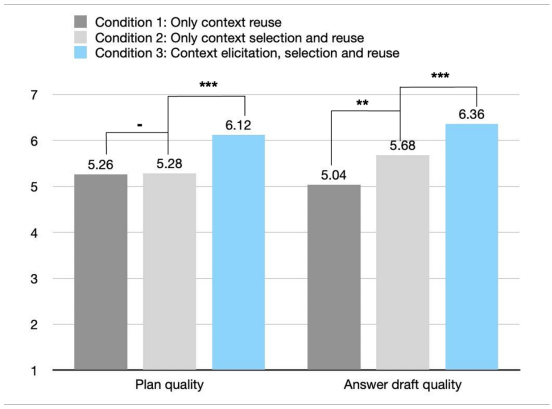


Figure 2: Expert evaluation of plan and answer draft quality across three conditions. Our task-structured context curation method (context elicitation, selection, and reuse) significantly outperforms both context reuse only and context selection and reuse only. Improvements are statistically significant (***$p < 0.001$, **$p < 0.01$).

within-subjects pilot study with six expert evaluators.[2] The study compared three variants of the same system: (1) *context reuse only*, where previously generated drafts could be reused across subtasks; (2) *context selection and reuse*, where the system additionally selected task-relevant context before generating each draft; and (3) the full *context elicitation, selection, and reuse* pipeline, where the system also asked targeted questions to gather missing context. Apart from these ablated mechanisms, all participants used the same interface, task structure, and model backend (i.e. `GPT-4o`).

The pilot showed that the full context-curation pipeline produced higher-quality action plans and answer drafts than the ablated variants (see Figure 2). These results support the design rationale behind *JumpStarter*: reuse preserves intermediate progress, selection focuses each generation step on relevant information, and elicitation supplies missing information that cannot be recovered from prior context alone. We use these validated mechanisms as the basis for the final system implementation.

3.3 Task-Structured Context Curation

JumpStarter implements *task-structured context curation* as an interactive LLM-based planning workflow. The system uses LLMs in two main ways: *task decomposition*, which organizes a complex goal into actionable subtasks, and *context curation*, which elicits, selects, and reuses information to support each subtask. Together, these operations allow *JumpStarter* to attach context and intermediate artifacts to specific planning steps rather than relying on a flat chat history. Full prompts are provided in Appendix J.

3.3.1 Task Decomposition

Subtask Generation Task decomposition provides the structural scaffold for planning. When a user enters a goal or chooses to break down an existing task, *JumpStarter* prompts `GPT-4o` to generate a set of actionable subtasks. Each subtask includes a name, description, and estimated completion time, helping users understand the scope and effort required. To reduce redundancy, the prompt includes the current task tree and instructs the model to generate subtasks that complement the existing structure (see Figure 14).

Subtask Detection Before generating a draft for a selected task, *JumpStarter* performs *subtask detection*: it evaluates whether the task is sufficiently detailed and actionable. If the task is too broad, the system recommends further decomposition; otherwise, it proceeds to context curation and draft generation. We implement this feature with `GPT-4o` using Chain-of-Thought prompting (Wei et al., 2022) with few-shot examples and the task's tree level as additional input. We report the evaluation details for subtask detection in Section D.

Task Forking Some tasks require parallel rather than sequential decomposition. Consider a user preparing PhD applications: they may need to contact multiple professors, compare multiple institutions, or invite multiple recommenders. For these cases, *JumpStarter* supports *task forking*, where we prompt `GPT-4o` to determine whether a task should be decomposed into entity-specific subtasks (see Figure 16). When forking is appropriate, the system first identifies the context relevant to the entities involved, then generates parallel subtasks grounded in those selected context elements. This helps produce subtasks that are independent and non-redundant.

3.3.2 Context Curation

Context curation operationalizes the three mechanisms of task-structured context curation: elicitation, selection, and reuse.

Context Elicitation At goal initialization, *JumpStarter* prompts `GPT-4o` to identify information that would help personalize the plan, including constraints, preferences, deadlines, resources, prior

[2] Refer to Appendix C for participant backgrounds and the detailed pilot procedure.

progress, and relevant documents (see Figure 11). The system presents these as elicitation questions or document suggestions, and the user's responses are stored as global context for the planning session.

Context Selection For each subtask, *JumpStarter* performs task-local context selection before draft generation. The system constructs a candidate context pool from global context, local user responses, saved drafts, and related subtasks. Given the selected task's title and description, `GPT-4o` selects the context entries most relevant to the current subtask.[3] These selected context keys are shown to the user for review, allowing the user to remove irrelevant items or add missing information before generation. This step scopes generation to the current planning decision instead of indiscriminately passing all available context to the model.

Context Reuse and Answer Draft Generation After context selection, *JumpStarter* prompts `GPT-4o` with the user's goal, the selected subtask, global context, and task-local selected context to generate an answer draft. The draft is designed to be a concrete intermediate artifact, such as a checklist, schedule, or message. When the user approves a draft, the system stores it as a reusable local context. Later subtasks can select and reuse these saved drafts, allowing *JumpStarter* to carry forward refined intermediate artifacts throughout the planning workflow. When users want to improve a draft but are unsure what additional information would help, *JumpStarter* generates targeted follow-up questions for the current subtask (see Figure 12). The user's responses are added to the local context, and the system regenerates the draft using the updated context.

4 Evaluation of *JumpStarter*

We evaluate *JumpStarter* through two complementary studies. A within-subjects user study compares it with ChatGPT on participants' own planning goals, measuring planning experience, output quality, confidence, and workload. A large-scale simulation study on a human-grounded benchmark built from the user-study traces then tests whether these benefits hold against strong baselines and ablations.

[3] See prompts in Appendix J.2

Participant	Personal goal	Goal type
P1	Start a side job	Career
P2	Organize a weekly game night	Life
P3	Land a job offer	Career
P4	Prepare for the LSAT	Academia
P5	Manage social media accounts	Creativity
P6	Move to a new apartment	Life
P7	Create a portfolio website	Creativity
P8	Prepare to deliver a tutorial	Academia
P9	Start a YouTube channel	Creativity
P10	Organize a family reunion	Life

Table 1: Overview of personal goals picked by participants in the user study.

4.1 User Study

We conducted a within-subjects study with ten participants comparing *JumpStarter* against ChatGPT.[4] We selected ChatGPT as the baseline because it is familiar to users and supports contextual interaction across a session, making it a realistic comparison for goal-oriented planning.

Each participant used both systems to explore a personal goal they intended to pursue in the near future. After interacting with each system, participants completed questionnaires assessing task load and satisfaction with outcomes. We also conducted semi-structured interviews to gather qualitative insights into their experience with each system.

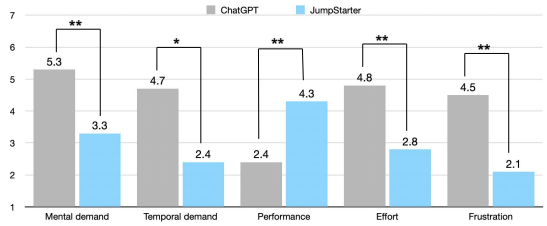
4.1.1 Participants and Procedure

We recruited ten participants (average age=23.8; six female, four male) through a university mailing list and word of mouth. All reported being familiar or very familiar with ChatGPT. Before the study, each participant selected a personal goal to pursue within the next six months (see Table 1). At the start of each session, participants were introduced to the concepts of action plans and answer drafts through examples, then used both ChatGPT and JumpStarter to plan and generate answer drafts for as many subtasks as possible. The system order was randomized and counterbalanced to mitigate order effects. Each task was limited to 25 minutes, and each study session lasted approximately 1.5 hours. All participants were compensated $20 per hour. Appendix E discusses the scope of the study.

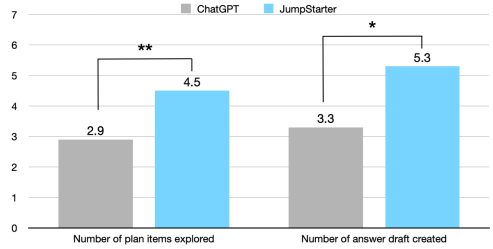
4.1.2 Results and Findings

We collected 1–7 scale questionnaire ratings on task load, outcome satisfaction, and confidence, and recorded the number of plan items explored

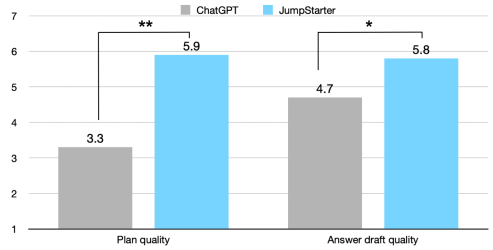
[4] We used the ChatGPT interface with `GPT-4o` specified in the user study.



(a) NASA Task Load Index Comparison



(b) Exploration Efficiency Comparison



(c) Result Satisfaction Comparison

Figure 3: User study results comparison between using ChatGPT and using *JumpStarter*. The statistical test results comparing JumpStarter with ChatGPT, where the p-values ($*$: $p < .050$, $**$: $p < .010$, $***$: $p < .001$) are reported.

and answer drafts generated by each participant. We further analyzed semi-structured interviews using thematic analysis (Braun and Clarke, 2006). Figure 3 summarizes the main results; additional details are provided in Appendix E.

***JumpStarter* Reduces Users' Task Load** As shown in Figure 3a, participants reported significantly lower workload with *JumpStarter* than with ChatGPT across mental demand ($p = .005$), temporal demand ($p = .012$), performance ($p = .007$), effort ($p = .007$), and frustration ($p = .009$). Participants attributed this reduction to *JumpStarter*'s task structure and context-aware guidance, which reduced the need to manually maintain planning state and decide what information to provide next. For example, P2 noted that ChatGPT required them to keep the structure in their head, whereas *JumpStarter* provided a structure they could follow.

***JumpStarter* Improves Exploration Efficiency.** Participants explored more plan items with *JumpStarter* than with ChatGPT (4.5 vs. 2.9), and generated more answer drafts (5.3 vs. 3.3). This suggests that *JumpStarter*'s task-structured workflow helped users make broader progress across their goals. In interviews, participants explained that ChatGPT's linear format often led them to over-iterate on a single thread, while *JumpStarter* made it easier to move across subtasks while preserving relevant context and prior drafts.

***JumpStarter* Improves Perceived Outcome Quality.** Participants rated plans produced with *JumpStarter* as higher quality than those produced with ChatGPT (5.9 vs. 3.3), and also rated answer drafts higher (5.8 vs. 4.7). Participants attributed this improvement to targeted elicitation and task-relevant context use. For example, P9 noted that *JumpStarter* incorporated their prior video-editing experience into the YouTube-channel plan, while ChatGPT focused on editing advice they did not need.

4.2 Human-Grounded Automatic Simulation

We complement the user study with an automatic simulation study to examines whether the user-study findings generalize across a broader set of goals and personas, while systematically comparing *JumpStarter* against relevant competitive baselines and ablations.

4.2.1 Dataset Construction

We construct a human-grounded simulation benchmark from realistic planning goals and persona-based simulated users. For goals, we seed the benchmark with 13 logged study goals (10 from the user study, 3 from the pilot) and synthesize 47 additional goals with `GPT-5.5`, yielding 60 personalized planning tasks stratified across five domains (see Table 8). Each synthesized candidate must pass an LLM-based rubric filter that enforces realism, personalization, decomposability, and tangible outputs, as well as manual inspection by the leading authors. For users, we extract 10 base personas from user-study traces and create 10 controlled variants, yielding 20 personas. Each persona specifies interaction-relevant attributes such as expertise, context richness, decision-making style, communication style, and information that should only be revealed when explicitly asked. We provide details on goal synthesis, persona construction, and validation in Appendices F.

Crossing 60 goals with 20 personas yields 1,200 possible profiles. We reserve 10 profiles for development. For the main evaluation, we sample 300 of the remaining profiles, balanced so that each goal appears five times and each persona fifteen times.[5] Because LLM-based user simulators can be overly cooperative and may not reliably reflect diverse human behaviors (Seshadri et al., 2026), we calibrate simulated-user decisions using real user study logs, including task-tree acceptance, draft saving, node completion, and revision behavior. Details of the calibration process and example simulation profiles are provided in Appendix F.2.

4.2.2 Baseline Methods

We compare *JumpStarter* against three groups of baselines. ***ChatGPT baselines*** use `GPT-4o` in a multi-turn planning setting that mirrors the user-study ChatGPT condition: *vanilla ChatGPT*, *ChatGPT + elicited context*, and *ChatGPT + structured summary*. These baselines test whether improvements come from additional user information or a better-organized prompt. We further add *single-turn decomposition*, which receives the same elicited context and is explicitly prompted to decompose the goal and draft every subtask in one turn, testing whether `GPT-4o`'s own decomposition can replace an external task structure. ***Planning and memory-agent baselines*** represent alternative agentic approaches to complex planning: *ADaPT* (Prasad et al., 2024) for recursive task decomposition, *Ask-Before-Plan* (Zhang et al., 2024) for clarification-first planning, an unstructured memory-RAG baseline inspired by memory-augmented agents such as MemGPT (Packer et al., 2023), and an *integrated agentic planner* that elicits global context, retrieves from the user's materials, queries a simulated browse tool, and selects context for each subtask, but does not reuse drafts across subtasks. These baselines test whether decomposition, clarification, retrieval, or their combination can match task-structured context curation. ***JumpStarter workflow ablations*** isolate the contribution of *JumpStarter*'s mechanisms: *random context selection*, *no context selection*, *no context reuse*, and *no elicitation*. These conditions test whether performance depends on task-relevant context selection, intermediate artifact reuse, proactive elicitation, or merely exposing the model to more context. All methods are evaluated on the same 300 simulation profiles, with `GPT-4o` used for workflow execution and simulated-user responses. Detailed baseline descriptions are provided in Appendix I.

[5] We use 300 profiles due to budget constraints. We release all 1,200 profiles to support larger-scale evaluation.

4.2.3 Metrics and Judging

We use *GPT-5.5* as a blinded judge to evaluate the final outputs from each condition. To reduce presentation bias, we remove method names before judging. The judge scores each output from 1 to 7 on the main outcome dimensions from the human study: plan quality, answer draft quality, confidence, personalization, and reduced burden. Our primary metric is a weighted composite of these dimensions. The full rubric and judging prompt are provided in Appendix H.

Before applying the judge to the full test set, we validate it on 10 matched within-subject JumpStarter–ChatGPT pairs from the user study, yielding 20 de-identified system outputs. The judge achieves a Spearman correlation of 0.779 with human ratings, suggesting alignment with the study construct. The full validation procedure is provided in Appendix H.

4.2.4 Results and Findings

We report *JumpStarter-Shallow* as the primary condition because it best matches how participants used the system in the real user study: a lightweight task structure with at most two decompositions beyond the initial subtask layer (see Figure 1). We also evaluate *JumpStarter-Recursive*, which uses the same context-curation mechanisms but allows subtasks to be decomposed further. Table 2 summarizes the results. *JumpStarter-Shallow* achieves the highest overall quality score (4.27), and positive deltas indicate matched improvements over each baseline.

JumpStarter outperforms ChatGPT baselines. *JumpStarter-Shallow* substantially outperforms vanilla ChatGPT (+0.52), ChatGPT with elicited context (+0.57), and ChatGPT with a structured context summary (+0.55). Since the latter two baselines receive the same elicited user information, these results suggest that the gains come from organizing context through the task-structured workflow, not merely from providing more user information to the model.

Context curation matters. Ablation results confirm that each context-curation mechanism contributes to performance. As shown in Figure 4, *JumpStarter-Shallow* achieves the best quality-

Method	Score	Δ
ChatGPT baselines		
ChatGPT vanilla	3.75	**+0.52**
+ elicited context	3.70	**+0.57**
+ structured summary	3.72	**+0.55**
Single-turn decomposition	4.05	**+0.22**
Planning/memory baselines		
ADaPT (Prasad et al., 2024)	4.05	**+0.22**
Ask-before-plan (Zhang et al., 2024)	3.22	**+1.05**
Memory-RAG (Packer et al., 2023)	3.29	**+0.98**
Integrated agentic planner	3.94	**+0.33**
JumpStarter ablations		
Random context selection	4.14	**+0.13**
No context selection	4.08	**+0.19**
No context reuse	4.09	**+0.18**
No context elicitation	4.19	**+0.08**
JumpStarter–Recursive	4.20	**+0.07**
JumpStarter–Shallow	**4.27**	–

Table 2: Automatic simulation results on 300 held-out persona-goal profiles. Scores are blinded judge study-quality composites. Δ reports the matched difference against *JumpStarter-Shallow*. Green deltas indicate statistically significant improvements where the 95% confidence interval of the paired difference excludes zero. Full confidence intervals are reported in Appendix 6.

efficiency tradeoff: it outperforms the no-selection baseline while using fewer input tokens on average (19,158 vs. 26,955). A GPT-based relevance labeler further shows that *JumpStarter-Shallow* selects substantially more relevant context than the no-selection baseline (0.72 vs. 0.22 relevance score; Appendix G). Removing context reuse lowers the score from 4.27 to 4.09, randomizing context selection lowers it to 4.14, and removing elicitation lowers it to 4.19. Together, these results show that performance depends on selecting task-relevant context, preserving intermediate artifacts, and eliciting missing information when needed, rather than increasing context volume alone.

Generic planning and memory agents do not replace workflow structure. *JumpStarter-Shallow* outperforms ADaPT (+0.22), Ask-before-plan (+1.05), and memory-RAG (+0.98). It also outperforms the integrated agentic planner (+0.33), which elicits, retrieves, and selects context dynamically; the agent's largest deficits are in personalization (+0.63) and in context curation and reuse

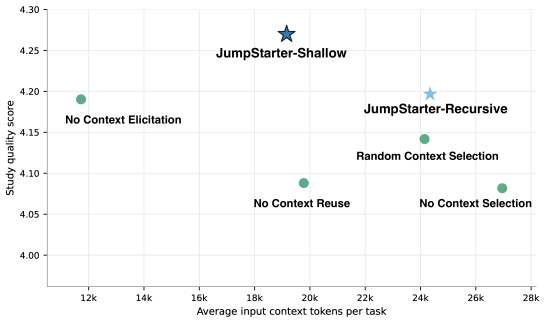


Figure 4: JumpStarter ablation results comparing study quality against input context-token usage. Each point reports the mean quality score and average input context tokens per task over 300 persona-goal profiles. *JumpStarter-Shallow* offers the strongest quality-efficiency tradeoff, achieving the highest quality score while using fewer context tokens than most ablations.[7]

(+0.58).[6] Appendix I analyzes this comparison in detail. These results suggest that decomposition, clarification, or memory retrieval alone is insufficient for personalized planning; the key advantage is using task structure to transform context into local artifacts and reuse them across the workflow.

Lightweight task structure is more effective than deeper recursion. *JumpStarter-Recursive* is competitive (4.20), but *JumpStarter-Shallow* performs better (+0.07). This aligns with our real-use traces, where participants generally used *JumpStarter* as a shallow planning workspace rather than a deeply recursive planner. The strongest configuration provides enough structure to support context selection and reuse without over-decomposing the task.

4.2.5 Failure Case Analysis

We compare *JumpStarter-Shallow* with its strongest one-shot competitor, single-turn decomposition, on each profile. *JumpStarter-Shallow* loses on 31% of the profiles, and more often in a few settings (Appendix Table 7).

Gains shrink on simple, single-artifact goals. On simple everyday-administration goals, such as decluttering a home office, the relative gain in study quality falls from 5.4% to 2.6% and the loss rate rises to 38%. Losses are also more frequent for goals spanning weeks rather than months (36% vs. 29%), for career and education goals

[6] Mean difference (*JumpStarter-Shallow* minus agent) in the judge's 1–7 scores on each criterion; see the rubric in Appendix H.

[7] We use `tiktoken` to compute token counts.

(38%), and for users who volunteer little context (40%). *JumpStarter-Recursive* gains 6.0% on complex goals but only 1.3% on simple ones, which is not significant.

Failures stem from fragmented outputs, not poor grounding. On the profiles it loses, *JumpStarter-Shallow* still scores 22% higher on context curation and reuse and matches the baseline on personalization. However, it scores 13–18% lower on plan quality, tangible results, confidence, and user burden: its output is about three times longer and split across subtask drafts. For decluttering a home office, it produced four overlapping drafts that left undecided details as placeholders, and it scored 22% lower than the baseline's single checklist-based plan. Deeper recursion adds inconsistency: *JumpStarter-Recursive* loses to *JumpStarter-Shallow* on 61% of simple goals and scores 23% lower on avoiding contradictions. Task-structured curation therefore pays off on complex, multi-deliverable goals, while simple goals are better served by a single consolidated draft.

5 Conclusion

We introduced *task-structured context curation*, a framework for human-AI planning that organizes user context around the evolving structure of a complex goal. Implemented in *JumpStarter*, this approach decomposes goals into actionable subtasks while eliciting missing information, selecting task-relevant context, and reusing intermediate artifacts. A within-subjects user study shows that *JumpStarter* improves plan and draft quality, exploration, confidence, and workload compared to ChatGPT. A human-grounded simulation study further shows that these gains persist across broader goals and personas, and cannot be explained by stronger prompting, additional context, retrieval, clarification-first planning, or deeper recursion alone. Together, our findings suggest that effective human-AI planning depends on attaching the right context to the right subtask at the right time.

6 Limitations

JumpStarter helps users reason about *how* to pursue personal goals by making them more concrete, structured, and actionable. However, it does not directly address the *why* dimension of goal pursuit, such as motivation, self-regulation, or emotional challenges. For example, one participant reported lower confidence after realizing how much work their goal required. While motivational support could help, prior work cautions that LLM-based emotional support raises safety and ethical concerns (Bhattacharjee et al., 2024). Future work should explore how planning systems can provide motivational scaffolding responsibly.

Our evaluation combines a real user study with a larger automatic benchmark, but simulated users and LLM-based judges cannot fully replace human behavior and evaluation. Simulated users may be more cooperative or consistent than real users, and LLM judges may introduce model-specific biases. Future work should validate these findings through larger human studies or longitudinal deployments.

JumpStarter focuses on cognitive and knowledge-work goals, including academia, career development, creativity, and everyday life. It does not yet address physical, behavioral, or spiritual goals, such as losing weight, overcoming shyness, or coming to terms with one's faith (Little, 1983). These goals often require habit formation, accountability, or motivational support (Consolvo et al., 2009; Ekhtiar et al., 2023), which remain directions for future work.

Finally, *JumpStarter* relies on `GPT-4o` as its core generation engine. Although LLMs can generate useful plans, they remain prone to factual errors and hallucinations (Achiam et al., 2023). Future work should integrate retrieval, verification, or specialized search agents to improve factual grounding for goals requiring up-to-date or domain-specific information.

7 Ethics Statement

Biases *JumpStarter* relies on pre-trained LLMs, which may reflect biases present in their training data and could produce unfair, stereotypical, or otherwise inappropriate suggestions. We did not explicitly mitigate model bias in this work. To reduce potential harm, our system is designed for human-AI collaboration rather than autonomous decision-making: users remain in control of selecting context, reviewing outputs, and deciding whether to act on generated plans.

Reproducibility We use `GPT-4o` as the backbone of *JumpStarter*. Because proprietary LLMs may change over time, exact reproduction of model outputs may be difficult. To support reproducibility, we provide the prompts, system design details, and evaluation protocols used in our experiments.

Study Participants We recruited participants for the pilot study and the real user study through a university mailing list and word of mouth. Participants were compensated at a rate of $20 per hour. To protect privacy and anonymity, we will not release participants' personal information, demographic details, raw interaction logs, or goal-related materials that may contain sensitive personal context. Any released examples will be anonymized.

Acknowledgments

We thank the members of the Columbia NLP Lab and the Computational Design Lab at Columbia University for helpful discussions of this work. We are grateful to the participants of our user study for their time and insights, and to the anonymous reviewers for their constructive feedback, which helped improve the paper.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins Sri, Anthony Barrett, Dave Christianson, et al. 1998. Pddl—the planning domain definition language. *Technical Report, Tech. Rep.*

John Annett and Keith D Duncan. 1967. Task analysis and training design.

Anthropic. 2024. Introducing the model context protocol. Accessed: 2025-04-29.

Victoria Bellotti, Brinda Dalal, Nathaniel Good, Peter Flynn, Daniel G Bobrow, and Nicolas Ducheneaut. 2004. What a to-do: studies of task management towards the design of a personal task list manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 735–742.

Eric Bergman and Donald A Norman. 2000. Making technology invisible. *Information Appliances and Beyond: Interaction Design for Consumer Products*.

Ananya Bhattacharjee, Yuchen Zeng, Sarah Yi Xu, Dana Kulzhabayeva, Minyi Ma, Rachel Kornfield, Syed Ishtiaque Ahmed, Alex Mariakakis, Mary P Czerwinski, Anastasia Kuzminykh, et al. 2024. Understanding the role of large language models in personalizing and scaffolding strategies to combat academic procrastination. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–18.

Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Stuart K Card, Thomas P Moran, and Allen Newell. 2018. *The psychology of human-computer interaction*. Crc Press.

Kenneth Conley and James Carpenter. 2007. Towel: Towards an intelligent to-do list. In *AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*, pages 26–32.

Sunny Consolvo, Predrag Klasnja, David W McDonald, and James A Landay. 2009. Goal-setting considerations for persuasive technologies that encourage physical activity. In *Proceedings of the 4th international Conference on Persuasive Technology*, pages 1–8.

Fergus IM Craik and Robert S Lockhart. 1972. Levels of processing: A framework for memory research. *Journal of verbal learning and verbal behavior*, 11(6):671–684.

Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. 2023. A survey on proactive dialogue systems: Problems, methods, and prospects. *arXiv preprint arXiv:2305.02750*.

Tina Ekhtiar, Armağan Karahanoğlu, Rúben Gouveia, and Geke Ludden. 2023. Goals for goal setting: a scoping review on personal informatics. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, pages 2625–2641.

Kutluhan Erol, James Hendler, and Dana S Nau. 1994. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128.

KJ Kevin Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S Weld, Amy X Zhang, and Joseph Chee Chang. 2026. Cocoa: Co-planning and co-execution with ai agents. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, pages 1–23.

Richard E Fikes and Nils J Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Diimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153.

Sandra G Hart. 2006. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 904–908. Sage publications Sage CA: Los Angeles, CA.

Mark K Ho, David Abel, Carlos G Correa, Michael L Littman, Jonathan D Cohen, and Thomas L Griffiths. 2022. People construct simplified mental representations to plan. *Nature*, 606(7912):129–136.

Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166.

Edwin Hutchins. 2000. Distributed cognition. *International encyclopedia of the social and behavioral sciences*, 138(1):1–10.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

William Jones, Predrag Klasnja, Andrea Civan, and Michael L Adcock. 2008. The personal project planner: Planning to organize personal information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 681–684.

Victor Kaptelinin. 2003. Umea: translating interaction histories into project contexts. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 353–360.

Andruid Kerne, Andrew M Webb, Steven M Smith, Rhema Linder, Nic Lupfer, Yin Qu, Jon Moeller, and Sashikanth Damaraju. 2014. Using metrics of curation to evaluate information-based ideation. *ACM Transactions on Computer-Human Interaction (ToCHI)*, 21(3):1–48.

Mik Kersten and Gail C Murphy. 2006. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 1–11.

Yash Kumar Lal, Li Zhang, Faeze Brahman, Bodhisattwa Prasad Majumder, Peter Clark, and Niket Tandon. 2024. Tailoring with targeted precision: Edit-based agents for open-domain procedure customization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15597–15611.

Christine Lee, David Porfirio, Xinyu Jessica Wang, Kevin Zhao, and Bilge Mutlu. 2025. Veriplan: Integrating formal verification and llms into end-user planning. *arXiv preprint arXiv:2502.17898*.

Pan Liang, Danwei Ye, Zihao Zhu, Yunchao Wang, Wang Xia, Ronghua Liang, and Guodao Sun. 2024. C5: toward better conversation comprehension and contextual continuity for chatgpt: P. liang et al. *Journal of Visualization*, 27(4):713–730.

Brian R Little. 1983. Personal projects: A rationale and method for investigation. *Environment and behavior*, 15(3):273–309.

Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. 2024a. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. *URL https://arxiv. org/abs/2401.02777*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Xiao Ma, Swaroop Mishra, Ariel Liu, Sophie Ying Su, Jilin Chen, Chinmay Kulkarni, Heng-Tze Cheng, Quoc Le, and Ed Chi. 2024. Beyond chatbots: Explorellm for structured thoughts and personalized model responses. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Chaitanya Malaviya, Joseph Chee Chang, Dan Roth, Mohit Iyyer, Mark Yatskar, and Kyle Lo. 2024. Contextualized evaluations: Taking the guesswork out of language model evaluations. *arXiv preprint arXiv:2411.07237*.

Thomas W Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119.

Kevin J Miller and Sarah Jo C Venditto. 2021. Multistep planning in the brain. *Current Opinion in Behavioral Sciences*, 38:29–39.

Dana Nau and Paolo Traverso. 2004. *Automated planning*. Morgan kaufmann.

Dana S Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J William Murdock, Dan Wu, and Fusun Yaman. 2003. Shop2: An htn planning system. *Journal of artificial intelligence research*, 20:379–404.

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

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2024. Adapt: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4226–4252.

Rupak Sarkar, Bahareh Sarrafzadeh, Nirupama Chandrasekaran, Nagu Rangan, Philip Resnik, Longqi Yang, and Sujay Kumar Jauhar. 2025. Conversational user-ai intervention: A study on prompt rewriting for improved llm response generation. *arXiv preprint arXiv:2503.16789*.

Preethi Seshadri, Samuel Cahyawijaya, Ayomide Odumakinde, Sameer Singh, and Seraphina Goldfarb-Tarrant. 2026. Lost in simulation: Llm-simulated users are unreliable proxies for human users in agentic evaluations. *arXiv preprint arXiv:2601.17087*.

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.

Significant Gravitas. AutoGPT.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

Wil MP Van der Aalst and Kees Van Hee. 2002. *Workflow management: models, methods, and systems*. The MIT press.

Tiannan Wang, Meiling Tao, Ruoyu Fang, Huilin Wang, Shuai Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2024. Ai persona: Towards life-long personalization of llms. *arXiv preprint arXiv:2412.13103*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

J Diego Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, pages 1–21.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. Ask-before-plan: Proactive language agents for real-world planning. *arXiv preprint arXiv:2406.12639*.

Xuanming Zhang, Rahul Divekar, Rutuja Ubale, and Zhou Yu. 2023. Groundialog: A dataset for repair and grounding in task-oriented spoken dialogues for language learning. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 300–314.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. 2024. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*.

A Problem Scope and Extended Related Work

Problem Scope We focus on a specific class of planning problems: open-ended, long-horizon planning of personal knowledge-work goals, such as preparing a graduate application or organizing a family reunion. Unlike the problems studied in automated planning, they have no formal domain model (no state space, action schema, or goal predicate); the goal is under-specified and must be elicited from the user and refined during the session; the relevant state is the user's evolving knowledge and context rather than a world state; and there is no ground-truth plan, so success is the user's judgment of plan quality. A person stays in the loop, and the output is a set of interdependent deliverables that span weeks or months. We therefore address the phase before formal planning can begin: turning an ill-defined goal and scattered context into a structure a person can act on.

Relation to Automated Planning Classical planning assumes a formal domain model: a state space, actions with preconditions and effects, and a goal condition, as in STRIPS (Fikes and Nilsson, 1971) and PDDL (Aeronautiques et al., 1998), over which a planner searches for a valid or optimal action sequence (Nau and Traverso, 2004). Hierarchical task network (HTN) planning is the closest analogue to *JumpStarter* because it also decomposes tasks, but it relies on a hand-authored library of decomposition methods over a well-defined domain (Erol et al., 1994; Nau et al., 2003). The problems *JumpStarter* targets arise before this machinery can run: no domain model exists, and the goal itself must be surfaced and structured with the user. *JumpStarter* supports this pre-formalization phase, and its task structure organizes context for an LLM rather than actions for a planner.

Task Management and Workflow Systems Organizing work around tasks has a long history. Workflow management systems model and automate business processes (Georgakopoulos et al., 1995; Van der Aalst and Van Hee, 2002), hierarchical task analysis describes the goals and subgoals of human work (Annett and Duncan, 1967; Card et al., 2018), personal task managers track to-dos (Bellotti et al., 2004), and coordination theory studies dependencies between activities (Malone and Crowston, 1994). These systems organize tasks for people to execute and coordinate, so the unit they track is an activity. *JumpStarter* uses task structure for a different purpose: each subtask holds elicited, selected, and reused context, and the operations on it exist to give an LLM the right context for that subtask.

B *JumpStarter* System Walkthrough

To illustrate how users interact with *JumpStarter*, we present a walkthrough using a concrete example. Consider John, a user aiming to apply for a PhD in NLP. Figures in the Section B.7 provide corresponding interface visuals.

B.1 Goal Input and Global Context Elicitation

John begins by entering his goal—*"Apply for a PhD in NLP"*—into a text input box and clicks *Start*. The system then generates elicitation questions to collect relevant context (e.g., existing documents, preferences). In John's case, it asks about potential target schools and recommendation letters. Since he is uncertain about the former and has not yet obtained the latter, he uploads his CV in response to the first question and clicks *Let's get started*. The elicited context becomes part of the global context used across subsequent tasks.

B.2 Subtask Generation and Detection

Using the elicited context, *JumpStarter* generates an initial subtask tree, presenting titles, descriptions, and estimated durations (Figure 7). John reviews the list to gain an overview of the plan and decides to explore the subtasks sequentially.

John selects the first subtask, *Identify Potential PhD Programs*. The system detects that it is insufficiently actionable and prompts John to either decompose the task or proceed with drafting. Following the system's recommendation, John selects *Decompose the task*, resulting in a new set of subtasks under the original node (Figure 8).

B.3 Answer Draft Creation and Refinement

John selects the subtask *Research Universities and Programs*, which the system deems actionable. It generates an initial answer draft—a list of NLP PhD programs. Upon review, John refines the output by adding, *"I want schools in the Midwest of the US."* He is satisfied with the revised list and saves it as an answer draft. He is also given three refinement options: regenerate, add more context and regenerate, or iterate on the current draft. Saved drafts are stored as context and appear as icons on the task tree, marking completed nodes.

B.4 Task Forking

John proceeds to the next subtask, *Identify Faculty Members*. The system suggests decomposing the task and detects that it requires forking based on the previously identified programs. It selects the saved university list as relevant context and asks John to confirm or modify the selection (Figure 9a). Upon confirmation, the system forks the task into program-specific subtasks (Figure 9b), which John begins to explore individually.

John then explores the task *Get Recommendation Letters* and decomposes it as prompted. For the subtask *Compile a List of Recommenders*, the system uses his CV to generate a list including Prof. Blake White, Prof. Julian Deng, and Dr. Alice Feng. John accepts and saves the draft. He proceeds to *Reach Out to Potential Recommenders*, which is forked into person-specific subtasks. For *Reach Out to Prof. Blake White*, the system identifies relevant context—including John's CV, prior collaborations, and the university list—and generates a personalized email draft (Figure 10a).

B.5 Context Elicitation for Draft Iteration

Unsatisfied with the initial email draft, John clicks *Add Context and Regenerate*. The system prompts follow-up questions to elicit additional details (e.g., specific projects or papers), which John provides (Figure 10b). The refined draft incorporates these details, resulting in a more personalized and acceptable version.

Through structuring the goal as a hierarchy of tasks, *JumpStarter* helps John curate relevant context, enabling the creation of detailed action plans and high-quality answer drafts. *JumpStarter*'s structured context management and iterative refinement enable him to effectively progress toward his goal of applying for a PhD in NLP.

B.6 Data Representation

JumpStarter represents each piece of context as a key-value pair, where the key denotes the context name and the value specifies its content. For example, the pair *"Location preference: Midwest of US"* captures a user's geographical preference. The system maintains two types of context: global and local. Global context consists of information elicited from the user immediately after goal specification and is universally applied throughout planning. Local context includes answer drafts and any additional information the user provides during interaction.

To organize tasks, *JumpStarter* employs a hierarchical tree structure. This design mirrors the natural decomposition of complex goals into manageable components, facilitating clear tracking of task dependencies and progression. Each user goal serves as the root node, with system-generated subtasks represented as child nodes. Each subtask node stores task-specific attributes, including titles, descriptions, estimated durations, and any associated answer drafts.

B.7 System visuals

This section presents the interface visuals of the *JumpStarter* system. Figure 5 shows an overview of the interface as it generates plans and answer drafts for the goal *"Apply for a PhD in NLP"*. Figure 6 displays the elicitation interface used to gather global context before initiating the planning process. Figure 7 illustrates the subtask tree, providing a structured overview of the user's goal. Figure 8 shows how the system recommends further decomposition for a selected subtask. Figure 9 depicts context selection for task forking and the resulting forked subtask structure. Finally, Figure 10 presents the interface for generating an answer draft, including both context selection and additional elicitation.

C Details for Pilot Testing

C.1 Participants and Procedure

We validated the design principles of *JumpStarter* through a pilot study. Specifically, we evaluated subtask and answer draft quality for three preselected personal goals inspired by Little (1983): (a) Apply to a fellowship, (b) Get a driver's license, and (c) Organize a team event. We then used a university mailing list to recruit participants. Each goal was assigned two expert participants—those who reported achieving the goal in the past six months. Overall, we recruited six expert participants for this study (average age=25.8, three female, three male). Participants were compensated $20 per hour, with sessions lasting about 1.5 hours each.

During the study, we first introduced the expert participants to the concept of plans and answer drafts with examples. We instructed the participants that they would be using three versions of the system for their respective goals, from start to finish. The system versions represented the three

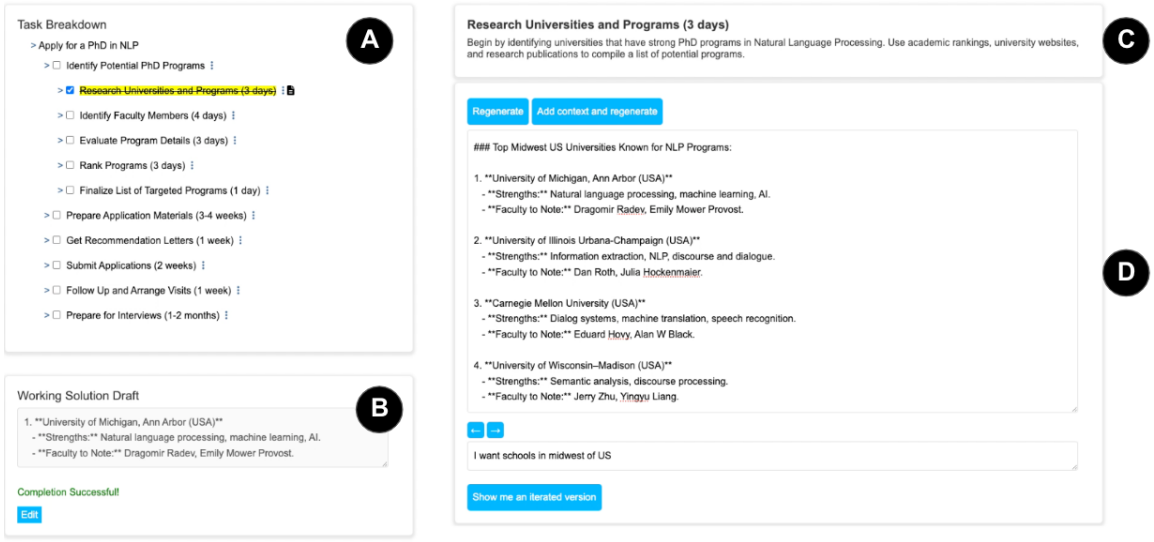


Figure 5: A screenshot of JumpStarter creating plans and answer drafts for the goal *Apply for a PhD in NLP*. (A) Task breakdown is shown as a subtask tree, with the goal being the root node. Subtasks decomposed from the same parent node are shown on the same level. (B) Saving the answer draft. (C) Detailed descriptions of the selected subtask are shown. (D) The answer draft is generated, considering the specification from the user – "*I want schools in midwest of US*". Users have three options to improve the draft: regenerate, add context and regenerate, and iterate based on users' new specifications.

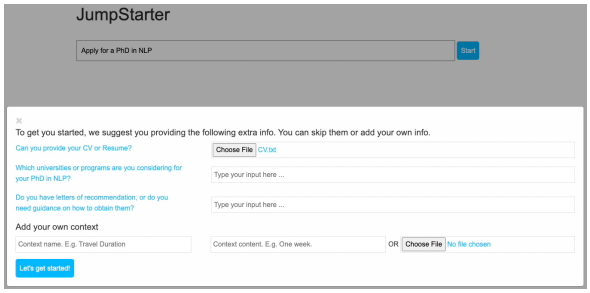


Figure 6: JumpStarter generates questions to elicit context from users to clarify the goal. The user uploads his CV.

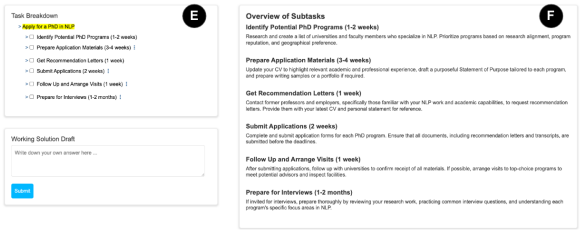


Figure 7: The initial subtask tree and the overview for the goal *Apply for a PhD in NLP*. (E) The task breakdown for the goal. (F) The overview of all the subtasks of the goal, including the titles, descriptions, and duration of completion of the subtasks.

experimental conditions, which were presented in shuffled order among participants to counterbalance the learning effect (see Table 3). We demonstrated how to use each version before each participant used it. We asked the participants to generate the subtasks and answer drafts exactly once. They then rated the perceived quality of the subtasks and answer drafts on a seven-point Likert scale, providing a verbal explanation.

C.2 Pilot Testing Analysis

Our results show that the context selection feature significantly enhances the quality of answer drafts compared to the baseline. Unlike context saving only, which keeps all context in the context window all the time, context selection requires the LLM to explicitly choose relevant context from the available pool. During sessions under this condition, participants often remarked that the generated solutions appeared to take into account what they had input in previous subtasks. This was particularly evident in "summarizing" tasks, where the system could provide a personalized checklist for tasks like applying for scholarships and driver's licenses, or an overall itinerary for a team event. E5 referred to the event itinerary they got as a *"very useful synthesis of everything I've explored."* In contrast, the context saving only condition tended to produce only general tips for creating the itinerary. As another example, E1 mentioned that the email draft generated with context selection was more personalized than that created without it. E1 stated, *"I*

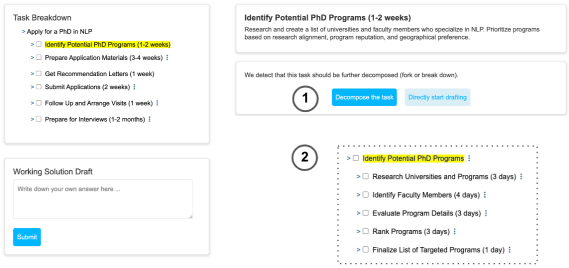
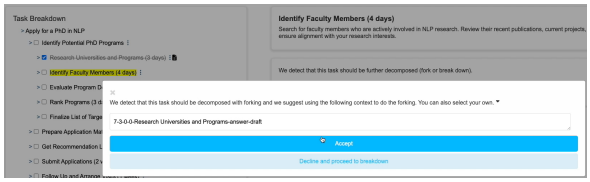
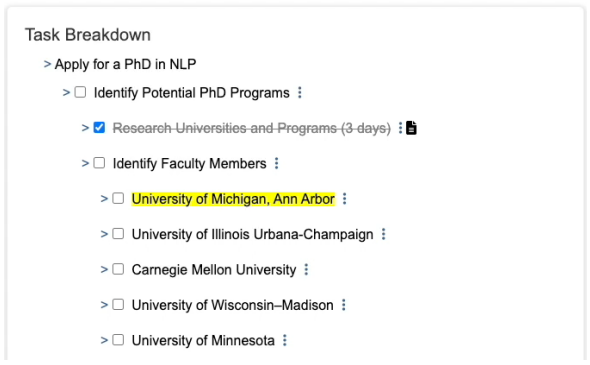


Figure 8: JumpStarter suggests further decomposition for the first subtask *Identify Potential PhD Programs*. (1) John presses the button *Decompose the task*. (2) JumpStarter generates the subtasks for John.

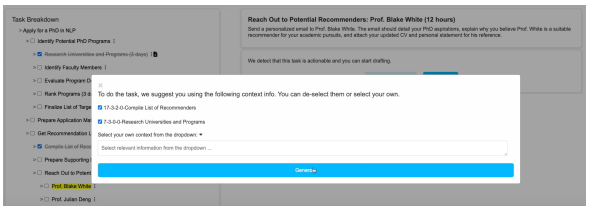


(a) Selecting relevant context for forking.

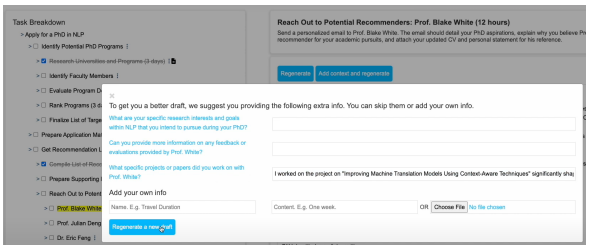


(b) Entity-based task decomposition (forking).

Figure 9: Context selection for forking and the task decomposition after applying forking on *Identify Faculty Members*.



(a) Context selection for answer draft generation.



(b) Context elicitation for answer draft iteration.

Figure 10: Context selection and elicitation for creating and iterating the answer draft of the subtask *Reach Out to Potential Recommenders: Prof. Blake White.*

Participant	Personal goal	Condition order
E1	Apply to a fellowship	(1) → (2) → (3)
E2		(2) → (3) → (1)
E3	Get a driver's license	(3) → (1) → (2)
E4		(1) → (2) → (3)
E5	Organize a team event	(2) → (3) → (1)
E6		(3) → (1) → (2)

Table 3: Overview of expert participants for the comparative study. Six experts were assigned one of three goals to evaluate under all three conditions, which were presented in shuffled order to avoid biasing the results.

like the recommendation letter request email draft it gives me, as it considers much of my background that I saved in the previous 'update your CV' task. I did the same thing in the previous round [context dumping] but did not feel it was as effective."

In addition to context selection, context elicitation improves the quality of both subtasks and answer drafts, outperforming both *context dumping* and *context filtering* conditions. Participants reported that the elicitation questions posed at the beginning *"provided the right plan to start with."* (E4) For instance, E2 uploaded the fellowship requirements document as initially suggested by the system, later rating the generated subtasks a perfect 7/7. *"It captured the requirements quite accurately..."* E2 noted. *"The subtasks were precise, fitting the unique aspects of the fellowship I am applying to, which requires only one recommendation letter, though typically more are needed."* Similarly, elicitation questions about which state to obtain a driver's license in (for the "get a driver license" goal) and how many people are in the team (for the "organize a team event" goal) both resulted in subtasks that were better tailored to participants' individual situations. In addition, preference elicitation questions such as "What type of vehicle do you intend to drive?" and "When do you prefer to hold the event? Weekday or weekend? Noon or night?" prompted participants to provide answers as personal context, ultimately resulting in more tailored answer drafts that they rated highly.

D Evaluation of Subtask Detection

D.1 Experiment settings

Subtask Detection aims to determine whether a task should be further decomposed to become executable. Using zero-shot and few-shot prompting as baselines, we evaluated three GPT-4-based prompting strategies for identifying actionable subtasks (see Appendix J for prompt details). First, we applied Chain-of-Thought (CoT) prompting with few-shot examples (Wei et al., 2022). Second, we incorporated the tree level of each task node into the prompt, based on the intuition that higher-level nodes are less likely to be immediately actionable ("Tree"). Third, we explored including the initial answer drafts in the prompt, allowing the model to assess whether decomposition is necessary based on draft quality ("Draft"). We assessed the accuracy of each method using an expert-labeled test suite. All experiments used a temperature of 1, a maximum token limit of 2048, and top-p set to 1. Results are reported as averages across five runs for each setting.

D.2 Test suite construction

Drawing inspiration from Little (1983), we created a test suite comprising four real-world task scenarios, each representing a distinct aspect of everyday life: 1) Applying for a PhD program (Academic), 2) Obtaining a driver's license (Practical), 3) Finding a surfing camp (Recreational), and 4) Arranging a trip abroad (Travel). For each scenario, we created ten distinct test cases, resulting in a total of 40 test cases. An example test case is shown as follows: *Compile a List of Potential Universities: start by identifying the universities that offer PhD programs in Natural Language Processing (NLP). Research and compile a comprehensive list of these universities.* We recruited four expert participants—one for each task scenario—via a university mailing list (average age=26.3, two female, two male). These participants reported having completed the tasks in the past six months and were compensated $10 for their participation. For each test case, the experts were asked to determine if the current task should be further decomposed to make it actionable. For the example test case shown above, its label is *"No"* as it is actionable and does not require further task decomposition.

Prompting Techniques	Accuracy Mean	Accuracy SD	Statistics p	Statistics Sig.
Zero-shot	.35	.000		
Few-shot	.58	.040		
+ CoT	.62	.050	.405	-
+ CoT + Tree	.69	.020	.004	**
+ CoT + Draft	.72	.020	.009	**
+ CoT + Tree + Draft	.87	.040	.000	***

Table 4: The technical evaluation results for Subtask Detection comparing different prompting techniques, where the p-values ($-$: $p > .100$, $+$: $.050 < p < .100$, $*$: $p < .050$, $**$: $p < .010$, $***$: $p < .001$) are reported. Note that the p-values are computed against the few-shot-only baseline. Few-shot combined with CoT+Tree+Draft achieved the best accuracy.

D.3 Results and Findings

As summarized in Table 4, the zero-shot method yielded the lowest accuracy of only 0.35, which also implies an inherent difficulty in the task itself. We observe enhanced performance with few-shot prompting (0.58), with accuracy rising even more when using the CoT prompting paradigm (0.62). Combining CoT few-shot prompting with the task node tree levels (0.69), initial solution draft (0.72), or both (0.87), all significantly led to enhanced performance. Generating the initial solution draft introduced a trade-off between latency and accuracy, so we opted for the slightly less performant few-shot CoT with tree levels for a better user experience.

E Detailed Design and Results of User Study

E.1 Hypotheses

In the user study, we investigate the following hypotheses:

- *H1*: Compared to ChatGPT, JumpStarter significantly **lowers users' task load (H1)** for mental demand (H1a), temporal demand (H1b), performance (H1c), effort (H1d) and frustration (H1e).
- *H2*: Compared to ChatGPT, JumpStarter significantly **increases users' exploration efficiency (H2)** in terms of the number of plan items explored (H2a) and the number of answer drafts created (H2b) within the given time.

Category	Factor	JumpStarter		ChatGPT		Statistics		Hypotheses
		Mean	SD	Mean	SD	p	Sig.	
Task load	Mental demand	3.3	1.64	5.3	1.06	.005	**	*H1a* accepted
	Temporal demand	2.4	.97	4.7	1.83	.012	*	*H1b* accepted
	Performance	4.3	.95	2.4	1.26	.007	**	*H1c* accepted
	Effort	2.8	1.69	4.8	1.23	.007	**	*H1d* accepted
	Frustration	2.1	1.60	4.5	1.90	.009	**	*H1e* accepted
Exploration efficiency	Plan items explored	4.5	1.43	2.9	.74	.002	**	*H2a* accepted
	Answer drafts generated	5.3	1.89	3.3	1.16	.047	*	*H2b* accepted
Satisfaction	Plan quality	5.9	.88	3.3	1.49	.004	**	*H3a* accepted
	Answer draft quality	5.8	.92	4.7	1.25	.017	*	*H3b* accepted
Confidence		5.6	1.07	3.9	.99	.007	**	*H4* accepted

Table 5: The statistical test results comparing JumpStarter with ChatGPT, where the p-values ($-$: $p > .100$, $+$: $.050 < p < .100$, $*$: $p < .050$, $**$: $p < .010$, $***$: $p < .001$) are reported.

- *H3*: Compared to ChatGPT, JumpStarter significantly **increases users' satisfaction level with the quality of results (H3)** in terms of plan quality (H3a) and answer draft quality (H3b).
- *H4*: Compared to ChatGPT, JumpStarter significantly **increases users' confidence in taking the next steps on their personal projects (H4)**.

E.2 Detailed Results and Findings

E.2.1 Confidence level in taking the next steps on the goal

Participants reported significantly higher confidence in taking the next steps on their personal projects using JumpStarter (mean=5.6, SD=1.07) compared to using ChatGPT (mean=3.9, SD=.99).

Participants reported that ChatGPT helped validate their thoughts with commonsense knowledge and sometimes provided surprising or useful tips. As P7 said, *"Sitting down and planning things out itself is very helpful. I used ChatGPT as a cross-reference, checking to make sure I'm on the right track—thinking similarly to other people. And sometimes answers to low-level tasks covered things I did not really know, which is good."* However, they reported that they still always felt they might miss something important while using ChatGPT, whereas with JumpStarter, they feel more secure (P3, P4, P9), as P4 commented, *"I love that I can break things down further if I want, so I don't feel like I miss anything."*

In addition, JumpStarter can provide very personalized and actionable next steps that greatly increase users' confidence in taking action on their projects. For example, P1, with the goal of starting a side job, shared that *"The schedule JumpStarter helped me generate is very personalized, and I can directly use it to take real action—before, I felt worried about launching this idea as I had very limited time, now I feel like I can really start doing it."* P10 also liked that JumpStarter provided them with a specific and personalized itinerary for organizing the family reunion—*"I like that it summarizes everything I saved in the previous tasks—I can use it in the real world."*

E.2.2 Tool preference

8 out of 10 participants reported they prefer to use JumpStarter in the future, compared to ChatGPT. The main reasons given include that JumpStarter can provide more customized responses with less cognitive load. The users feel that they do not have to think hard about what information to provide (P1), are guided by the system (P2), can more easily consume the information (P3) or track the plan (P6), and can get their personal details efficiently organized, framed, and utilized (P8, P9, P10).

The other two participants (P4 and P5) mentioned that their choice depended on how familiar they were with the project they wanted to work on. If it was a topic they already had a clear understanding of, they preferred the chatbot interaction to help them figure out the details. Otherwise, they would prefer to use JumpStarter as it offers more structure.

Methods	Study Quality	Δ vs. JS	95% CI	W/T/L
ChatGPT baselines				
ChatGPT vanilla	3.7507	+0.5192	[0.4602, 0.5782]	255/8/37
ChatGPT + elicited context	3.6992	+0.5707	[0.5092, 0.6322]	257/6/37
ChatGPT + structured summary	3.7215	+0.5483	[0.4897, 0.6070]	260/6/34
Single-turn decomposition	4.0503	+0.2195	[0.1637, 0.2753]	197/11/92
Planning and memory-agent baselines				
ADaPT-style recursive decomposition	4.0538	+0.2160	[0.1649, 0.2671]	200/20/80
Ask-before-plan	3.2208	+1.0490	[0.9791, 1.1189]	287/4/9
Memory-RAG	3.2898	+0.9800	[0.9234, 1.0366]	291/2/7
Integrated agentic planner	3.9375	+0.3323	[0.2813, 0.3833]	227/14/59
Workflow ablations and context-control baselines				
Random context selection	4.1418	+0.1280	[0.0690, 0.1870]	171/17/112
No context selection	4.0817	+0.1882	[0.1275, 0.2488]	178/9/113
No context reuse	4.0880	+0.1818	[0.1251, 0.2385]	180/17/103
No elicitation	4.1903	+0.0795	[0.0270, 0.1320]	163/15/122
JumpStarter-Recursive	4.1967	+0.0732	[0.0167, 0.1296]	158/12/130
JumpStarter-Shallow	**4.2698**	–	–	–

Table 6: Automatic simulation results on 300 held-out persona-goal profiles. Study Quality is the blinded judge's composite score. Δ reports the matched difference between *JumpStarter-Shallow* and each comparator; positive values indicate that JumpStarter performs better. W/T/L reports the number of profiles where *JumpStarter-Shallow* wins, ties, or loses against the methods. *No context selection* passes all available context to every subtask.

E.2.3 Improvement feedback

Participants also provided insights on how to improve JumpStarter. The main feedback included "*make the subtask outline and task descriptions editable*" (P1, P3, P6, P7, P8, P9), "*format the suggested answer draft to be easier to read*" (P4, P8, P9), "*enable users to add or edit context whenever they want*" (P3, P7), and "*add a synthesis button to summarize what has been explored so far*" (P1, P2). P8 also suggested embedding a search agent to collect data and ensure credibility. We discuss limitations and future work further in Section 6.

E.3 Additional User Quotes from User Study

E.3.1 *JumpStarter* Reduces Users' Task Load

As shown in Figure 3a, in the NASA TLX dimensions (Hart, 2006), working with *JumpStarter* was significantly less demanding in mental demand (p=.005), temporal demand (p=.012), performance (p=.007), effort (p=.007), and frustration (p=.009).

Participants attributed this to the structured interface and context-aware guidance of *JumpStarter*, which helped reduce cognitive load. Unlike ChatGPT's linear chat format, *JumpStarter* made it easier to visualize progress and maintain task structure. As P2 noted, *"ChatGPT info dumps a lot, and I have to keep the structure in my brain, whereas JumpStarter gave me a structure that I could easily follow."* Similarly, targeted questions helped users refine their input and move forward. P10 commented, *"I appreciate the questions JumpStarter asked when I felt stuck about how to iterate the answer draft."* In contrast, ChatGPT required users to generate and manage context manually. P8 remarked, *"In ChatGPT, the information load is high—I have to think very hard about what info I should provide to get things that work for me."*

Participants also reported expending more effort with ChatGPT, often without meaningful improvement in output quality. For example, P1 asked ChatGPT to generate clarifying questions to improve the answer draft. It returned eight, which P1 described as *"a bit too abstract and hard to answer."* Despite answering them all, P1 felt the revised output remained *"too general and not useful."* They also noted that *"ChatGPT seemed to forget these eight answers soon after"*, leading to frustration and a sense that their effort was wasted.

E.3.2 *JumpStarter* Enables Better Exploration Efficiency

Participants explored significantly more plan items using *JumpStarter* (mean=4.5, SD=1.43) than with ChatGPT (mean=2.9, SD=0.74), and created more answer drafts as well (mean=5.3, SD=1.89 vs. mean=3.3, SD=1.16). Figure 3b illustrates this difference.

One reason for this disparity is that ChatGPT's linear interaction format often leads users to fixate on a single task, limiting broader exploration. For instance, P2 iterated nine times on one answer draft:

Profiles	n	Loss rate	Gain
Goal complexity			
Complex	150	31%	+6.4%
Simple	150	31%	+4.5%
Time horizon			
Weeks	90	36%	+4.7%
Months or longer	210	29%	+5.8%
Domain			
Career, education	80	38%	+4.2%
Creative, personal	70	20%	+7.4%
Events	50	28%	+6.1%
Everyday admin	70	33%	+4.0%
Health, wellness	30	37%	+6.3%
Persona context richness			
Low	45	40%	+2.3%
Medium	195	30%	+5.7%
High	60	27%	+7.0%

Table 7: Failure-case breakdown of *JumpStarter-Shallow* against single-turn decomposition on the 300 held-out profiles. Loss rate is the share of profiles on which *JumpStarter-Shallow* scores lower; gain is its relative difference in mean study quality. Complex goals are those annotated *high* or *multi-track*; simple goals are *medium*.

"It took nine iterations to get the draft I like. I really hoped ChatGPT would guide me, but I had to direct myself. I got so involved that I completely forgot I had other planning items."

In contrast, *JumpStarter*'s task-structured interface helped users stay oriented within the broader plan while focusing on one task at a time. P4 remarked, *"JumpStarter has a more flexible structure. I like that I can easily jump between tasks... Seeing the task description and relevant context gives me everything I need"* P10 similarly noted, *"JumpStarter automatically manages and considers my drafts from previous tasks, which is great and helps me focus on the current one."*

E.3.3 *JumpStarter* Improves Perceived Quality of Outcome

Plans created with *JumpStarter* were rated significantly higher in quality (mean=5.9, SD=0.88) than those created with ChatGPT (mean=3.3, SD=1.49). Similarly, participants rated the quality of answer drafts higher with *JumpStarter* (mean=5.8, SD=0.92) compared to ChatGPT (mean=4.7, SD=1.25). Figure 3c shows these differences in perceived quality.

Participants attributed this improvement to *JumpStarter*'s proactive questioning and personalized planning. Unlike ChatGPT, which often produced generic plans, *JumpStarter* asked targeted questions early in the process and incorporated user responses into the plan structure. For example, P9, who was planning to start a YouTube channel, noted: *"JumpStarter asked if I was experienced with video editing, and I said yes. It was reflected in the plan accordingly, unlike ChatGPT, which focused too much on editing I didn't need."*

In addition, participants also appreciated how *JumpStarter* effectively uses the relevant context to personalize the answer draft, as P1 noted, *"JumpStarter gave me much more tailored responses—like a personalized schedule to help me start my side job. It took into account key details like my limited time and the specific area I am interested in. It handled the context very well."*

E.4 Extended Discussion about User Study

JumpStarter demonstrates how adaptive personal context curation through context elicitation, condensation, and reuse can enhance human-AI collaborative planning by improving both plan quality and user experience. Unlike ChatGPT's user-driven *"pull"* model, *JumpStarter*'s *"push"* approach reduces cognitive load by guiding users with targeted questions and enabling selective context previews, aligning with recognition-over-recall principles from cognitive psychology (Craik and Lockhart, 1972). Users valued this structured interaction but also expressed a desire for more persistent and adaptive memory, particularly as their goals and preferences evolved, highlighting the need for systems that can update context dynamically over time (Wang et al., 2024).

While ChatGPT is designed to retain conversational context, it often failed to apply it meaningfully, occasionally generating contradictory responses. These findings underscore the importance of designing systems that support more seamless and transparent context management. Looking forward, integrating in-situ context capture from tools like email or desktop files (Kaptelinin, 2003; Kersten and Murphy, 2006; Bergman and Norman, 2000) may enable more fluid, real-world human-AI collaboration. *JumpStarter* offers a design paradigm for building systems that better align LLM capabilities with users' evolving needs in complex, long-term planning workflows.

E.5 Scope of the User Study

With ten participants in one session each, the study establishes the direction of the effect for real users rather than its size or generality. All participants

were college students already familiar with ChatGPT; this familiarity favors the baseline, since *JumpStarter* was new to every participant. The design principles were validated separately in a pilot study with six expert evaluators (§3.2), and the simulation study (§4.2) tests whether the effect holds across a broader population of simulated users, including novice, low-context, terse, and deferential personas.

F Details on Dataset Construction for Automatic Simulation

The dataset for the automatic simulation experiment is constructed in three deterministic stages: (i) a 60-goal benchmark grounded in 13 anchor sessions from the human study, (ii) a 20-persona set extracted from the same study traces and augmented with controlled behavioral variants, and (iii) a $1,200$-profile persona $\times$ goal cross-product that fixes the private state each simulated user holds during a session.

F.1 Realistic Goal Synthesis

F.1.1 Anchor Sessions

We first extract 13 anchor sessions from the JumpStarter human study: 10 *human-study traces* from participants P1–P10 (one per participant, each participant's self-chosen planning goal) and 3 traces drawn from the pilot study tasks. Each anchor session records the participant identifier, goal text, domain, the participant's saved global and user context, background snippets, saved local-context entries, and saved drafts.

F.1.2 Domain Distributions

Goals are stratified across five planning domains with fixed quotas that sum to 60 (Table 8). The goal set covers five domains: career and education, creative and personal projects, everyday administration and home planning, event coordination, and health/wellness/training. Expected outputs include schedules, checklists, comparison matrices, outreach drafts, study plans, and budgets. The synthesized goal set is designed to avoid tasks where a generic one-shot response would be sufficient.

Domain	Count
`career_education`	16
`creative_personal`	14
`everyday_admin_home`	14
`events_coordination`	10
`health_wellness_training`	6
Total	**60**

Table 8: Domain distributions for the 60 goals.

F.1.3 Goal Synthesis Prompt and Model

Goals are synthesized with **GPT-5.5**. The prompt is shown below:

Prompt for Goal Synthesis

```
You generate human-grounded planning-goal benchmark candidates for
evaluating task-structured context curation.
Return only structured data matching the provided schema. Each goal
must be realistic for an individual user, require personal context,
benefit from decomposition, and support tangible outputs such as
schedules, drafts, checklists, itineraries, application materials,
or decision aids.
Write 'goal_text' as a short, plain goal title, not a full scenario
sentence. It must be easy to parse at a glance:
- 3-9 words. - 70 characters or fewer. - No final period. - No
commas, semicolons, colons, or multi-clause wording. - Avoid clauses
such as "while...", "within...", "with...", "for exploring...",
"that...". - Put details and constraints in 'known_context_needs',
'expected_tangible_outputs', 'time_horizon', and other metadata
fields instead.

Good goal_text examples:
- Prepare for the GRE exam - Apply for an NSF Graduate Research
Fellowship - Organize a PhD welcome picnic - Prepare for the bar exam
- Start a side job - Design a personal website - Start a personal
YouTube channel - Design a wardrobe for four seasons - Move to a
new apartment - Prepare for the half-marathon - Organize a family
reunion party at home

Avoid goals that are mostly fact lookup, one-shot generic advice,
unsafe, too verbose, hard to parse, or too close to the anchor goals.
Match the requested domain counts exactly.
```

F.1.4 Filtering and Quality Verification on Synthesized Goals

Each candidate passes through two filters before acceptance. A structural gate requires both `requires_personalization` and `requires_decomposition` to be true, non-empty `expected_tangible_outputs` and `known_context_needs`, compliance with the style rubric, and token-overlap similarity below 0.8 against every accepted goal. A separate GPT-5.5 LLM filter re-evaluates each candidate against the same rubric (see prompt below); acceptance requires both filters to pass. The synthesis–filter loop runs up to four times until 60 goals are accepted. Finally, the leading authors manually inspected all 60 goals end-to-end, confirming realism, decomposability, and the absence of duplicates or anchor-leakage before finalizing the goal set.

Prompt for Goal Filtering

```
You are filtering one planning benchmark candidate.
Accept only if all checks are true: - realistic for an individual
user, - requires personal context, - benefits from decomposition, -
can produce tangible artifacts, - is not mostly a search/fact-lookup
task, - is not too similar to the nearest existing goal. - has a
short, plain ‘goal_text‘ that reads like a goal title rather than a
full scenario sentence.
Reject if ‘goal_text‘ is over 9 words, over 70 characters, has clause
punctuation, ends with a long constraint clause, or uses phrasing
such as "while...", "within...", "with...", "for exploring...", or
"that...".
Return only structured data matching the schema.
```

F.2 Simulated Users Construction

The simulated-user set is the cross-product of a 20-persona set and the 60 synthesized goals (1, 200 *simulation profiles*). Personas describe *who the user is*; profiles bind a persona to a specific goal and encode the *private state* the user holds during a session: what they know, what they have not decided, what they will or will not disclose, what they will approve, and when they will push back.

F.2.1 Base Persona Extraction

We use **GPT-5.5** to extract the 10 base personas from the human-study anchors of P1–P10, one persona per participant. The extraction prompt (shown below) runs under structured output and is conditioned on the participant's saved global context, user context, background, and saved drafts; the model is required to summarize behavior, constraints, and communication style, and to ground every persona in evidence quotes from the anchor. Each persona is characterized along four behavioral axes:

- **Expertise level** $\in$ {novice, intermediate, expert}, inferred from the participant's stated background and the sophistication of saved drafts.
- **Context richness** $\in$ {low, medium, high}, inferred from the density of saved global and user-context entries.
- **Decision style** $\in$ {opinionated, deferential, collaborative}, inferred from the presence and forcefulness of stated preferences and constraints.
- **Communication style** $\in$ {terse, balanced, verbose}, inferred from sentence length and bullet density in the saved drafts.

Alongside these axes, the model produces `task_objectives`, `known_facts`, `preferences`, `constraints_to_remember`, `gaps_in_self_knowledge`, `information_disclosure_policy`, `response_style_rules`, `success_criteria`, `pushback_triggers`, and `evidence_quotes`. Identifiers (`persona_id`, `source_participant`, `source_anchor_id`, `goal`) and evidence quotes are preserved verbatim. To prevent identifiable institutional information from leaking into model-facing fields, all references to the participants' real universities are masked out.

Source	Axis	Value
P2	context_richness	low
P3	context_richness	high
P4	decision_style	deferential
P5	decision_style	opinionated
P6	communication_style	terse
P7	communication_style	verbose
P8	expertise_level	novice
P9	expertise_level	intermediate
P10	context_richness	medium
P1	decision_style	collaborative
P6	communication_style	balanced

Table 9: Controlled-variant specification: each variant clones a base persona and toggles a single behavioral axis to the listed value.

Prompt for Persona Extraction

```
You refine persona profiles for a human-AI planning benchmark.
Each persona should describe behavior, constraints, and communication
style, not final task answers. Preserve grounding in evidence quotes
and do not invent biographical details beyond what the evidence
supports.
Return only structured data matching the provided schema. Preserve:
- persona_id - persona_type - source_participant - source_anchor_id
- goal - evidence_quotes

You may improve: - user_description - expertise_level -
context_richness - decision_style - communication_style -
task_objectives - known_facts - preferences - gaps_in_self_knowledge
- constraints_to_remember - information_disclosure_policy -
response_style_rules - success_criteria - pushback_triggers -
extraction_warnings

Do not include ablation anchors as human-study personas.
```

F.2.2 Controlled Persona Variants

To stress-test the workflow under behaviorally distinct users while keeping each variant grounded in a real anchor, we generate 10 additional *controlled variants*. Each variant copies a base persona and toggles a single behavioral axis to a contrastive value; `variant_axis`, `variant_description`, and an extraction warning of the form *"Controlled variant modifies* $\langle$*axis*$\rangle$ *from base persona"* are recorded. The full variant specification is given in Table 9. The final persona set is therefore 20 personas (10 base + 10 variants).

F.2.3 Persona Validation and Example

Persona validation enforces: (i) exactly 10 base personas and 10 controlled variants; (ii) the set

of base source_participant identifiers equals $\{P1, \ldots, P10\}$; (iii) every persona_id is unique; (iv) every persona has at least one evidence quote; (v) every controlled variant has a non-null variant_axis; (vi) constraints_to_remember is non-empty (a warning otherwise). Below is an example persona:


```
{
  "persona_id": "P2_base",
  "persona_type": "base",
  "source_participant": "P2",
  "goal": "Organize a weekly PhD game night",
  "expertise_level": "novice",
  "context_richness": "medium",
  "decision_style": "opinionated",
  "communication_style": "balanced",
  "task_objectives": [
    "coordinate people, time, and location",
    "create a feasible event plan",
    "draft communication or logistics artifacts"
  ],
  "known_facts": [
    "Never organized game night before",
    "Availability: after 7:30pm, weekend nights",
    "Interests: STEM PhDs, fantasy/party games",
    "Size: 5-9 PhD students",
    "University: Cedarwick Hollow University"
  ],
  "constraints_to_remember": [
    "Interests: STEM PhDs, fantasy/party games",
    "Never organized game night before"
  ],
  "information_disclosure_policy": [
    "Do not volunteer unstated details; reveal
     known facts only when asked.",
    "If asked about absent info, say the user
     does not know yet or has not decided.",
    "Keep answers brief; let the assistant
     elicit missing planning context."
  ],
  "response_style_rules": [
    "Answer in 1-3 concise sentences.",
    "Provide specifics when asked; do not
     front-load every detail.",
    "Use a collaborative tone."
  ],
  "evidence_quotes": ["...verbatim spans..."]
}
```


F.2.4 Simulation Profiles

A simulation profile combines one persona with one benchmark goal, yielding $20 \times 60 = 1{,}200$ profiles in total. Each profile specifies the private state the simulated user holds at the start of a session: private_context_facts the user knows but does not necessarily volunteer; unknown_or_undecided_facts (e.g., budget, deadline, audience); must_reveal_if_asked and do_not_volunteer sets that govern disclosure; approval_criteria and pushback_rules that govern acceptance; the frozen stable_persona_traits block; and turn_style parameters for multi-turn behavior. In *source-goal pairs* the profile inherits the persona's known_facts and constraints_to_remember as private context; in *cross-goal pairs* biographical specifics are dropped and only stable_persona_traits carries over, preventing persona-specific facts (e.g., "900+ followers") from leaking into unrelated goals. Profiles are produced by GPT-5.5 (see prompt below), with identifiers preserved verbatim and no biographical invention permitted.

A validation run checks that all 1,200 persona–goal pairs are present, that simulation_profile_id values are unique, and that every profile has non-empty approval_criteria and pushback_rules and a positive turn_style.max_tokens.

Prompt for Profile Generation and Validation

```
You refine private simulation profiles for a persona-grounded user
simulator.
Each profile combines one persona with one benchmark goal. Preserve:
- simulation_profile_id - persona_id - goal_id - goal_text - domain -
source_persona_summary

You may improve: - private_context_facts - unknown_or_undecided_facts
- must_reveal_if_asked - do_not_volunteer - approval_criteria -
pushback_rules - stable_persona_traits - turn_style - initial_state
- state_tracking_notes

Rules: - Do not invent biographical facts not supported by the
source_persona_summary or goal metadata. - It is okay to add
goal-specific unknowns such as budget, deadline, availability, target
audience, or prior experience when the benchmark goal implies them.
- Keep facts concise and simulator-actionable. - Make the profile
useful for multi-turn state tracking. - Return only structured data
matching the schema.
```

An example simulation profile is shown below. At simulation time, this profile, the persona block, and the goal record are rendered into the GPT-4o user-simulator prompt. The simulator's turn-by-turn behavior—whether to volunteer a fact, whether to push back, whether to accept a draft, whether to expand a node—is governed by the disclosure policy, must_reveal_if_asked, do_not_volunteer, pushback_rules, and the calibrated decision policies of §F.2.5.


```
{
  "simulation_profile_id": "P2_base_G001",
  "persona_id": "P2_base",
  "goal_id": "G001",
  "goal_text": "Organize a weekly PhD game night",
  "domain": "events_coordination",
  "private_context_facts": [
    "Never organized game night before",
    "Availability: after 7:30pm, weekend nights",
    "Interests: STEM PhDs, fantasy/party games",
    "Size: 5-9 PhD students",
    "University: Cedarwick Hollow University",
    "Required outputs: venue shortlist,
     semester calendar, invitation message",
    "Time horizon: multi-week"
  ],
  "unknown_or_undecided_facts": [
    "expected group size: unknown until asked",
```

```
    "campus location: unknown until asked",
    "preferred game types: unknown until asked"
  ],
  "must_reveal_if_asked": [
    "Interests: STEM PhDs, fantasy/party games",
    "Never organized game night before",
    "expected group size",
    "campus location constraints",
    "preferred game types"
  ],
  "do_not_volunteer": [
    "expected group size",
    "campus location constraints",
    "preferred game types",
    "Size: 5-9 PhD students",
    "Availability: after 7:30pm, weekend nights"
  ],
  "approval_criteria": [
    "plan directly supports the stated goal",
    "assistant uses stated context only",
    "assistant asks for missing information
     before finalizing details",
    "includes venue shortlist",
    "includes semester calendar",
    "includes invitation message"
  ],
  "pushback_rules": [
    "push back on contradicted constraints",
    "push back on invented facts",
    "push back on generic plans w/o elicitation",
    "push back on skipped personal-context asks",
    "push back on missing tangible artifacts",
    "push back on fact lookup or one-shot advice"
  ],
  "stable_persona_traits": {
    "expertise_level": "novice",
    "context_richness": "medium",
    "decision_style": "opinionated",
    "communication_style": "balanced",
    "response_style_rules": [
      "Answer in 1-3 concise sentences.",
      "Provide specifics when asked; do not
       front-load every detail.",
      "Use a collaborative tone."
    ]
  },
  "turn_style": {
    "max_tokens": 220,
    "register": "collaborative",
    "length": "short"
  }
}
```

F.2.5 User Simulator Calibration

We calibrate simulated-user behavior using 16 logged *JumpStarter* sessions from the user study and pilot testing to reduce over-compliance in the automatic benchmark. From these traces, we estimate empirical behavior rates for task-tree acceptance, subtask decomposition, draft saving, node completion, context provision, revision, and final-readiness decisions (Table 10). The logs show that users generally accepted the top-level task structure, with an average of 5.73 top-level children, but were more selective about deeper decomposition: users proceeded with further decomposition in 72.1% of detected opportunities under a broad proxy, while realized child decomposition occurred in only 12.2% of cases. Users completed 53.2% of nodes on average and saved drafts for 51.0% of nodes.

Decision point	**Rate** (n)
Accept top-level task tree	1.000 (16)
Subtask: decompose further / draft	.721/.279 (197)
Realized child decomposition ($\geq$ depth 2)	.122 (197)
Save a generated draft	.748 (107)
Node marked completed	.776 (107)
Multi-generation revision before save	.832 (107)
Session provides global context	.733 (16)
Session provides local context	.933 (16)

Table 10: Empirical base rates from human-study logs used as simulator decision policies.

We validate the calibrated simulator with leave-one-session-out diagnostics across ten binary behavior families. The simulator achieves a mean Brier score of 0.157, five-bin expected calibration error of 0.047, and session-level absolute rate error of 0.214, indicating that its predicted decision probabilities are reasonably aligned with held-out real-user behavior. Thresholding decisions at 0.5 yields mean accuracy of 0.809 and mean F1 of 0.898. These results suggest that the calibrated simulator better matches empirical interaction patterns from real *JumpStarter* sessions, instead of assuming uniformly cooperative users who always accept decompositions, save drafts, complete nodes, or approve final outputs.

G LLM-based Context Relevance Labeler

To produce evidence on whether baseline methods select and reuse *relevant* context, we score every available context item against every subtask with an LLM-based relevance labeler. For each `context_selection` event in a session trace, the labeler emits one judgment per (subtask, available context item) pair: it sees only the subtask name and description and the context item's label and value, and produces a relevance score.

We use `GPT-5.4-mini` to judge the context relevance. The prompt is shown below. Specifically, relevance is judged *locally*: the model is told to score the relationship between the specific subtask and the specific context item. Besides, the score-to-decision mapping is explicit: `[0.00,0.20]` marks irrelevant or distracting items, `[0.21,0.40]` weak background, `[0.41,0.70]` supporting context that could help, and `[0.71,1.00]` directly useful con-

text; the binary `relevant` flag must be `true` only when the score is at least 0.50 *and* the value is concrete enough to act on.

Prompt for Context Relevance Labeler

```
You are labeling whether a saved user-context item is useful for a
specific planning subtask.
Evaluate only the relationship between the subtask and the context
item. Do not reward a context item merely because it is generally
related to the overall goal. A relevant context item should help
the system make a concrete decision, fill a field, personalize an
artifact, avoid a mistake, or reuse prior work for this specific
subtask.

Use this scale:
- 0.00-0.20: irrelevant, distracting, or unusable
- 0.21-0.40: weak background context
- 0.41-0.70: supporting context that could help the subtask
- 0.71-1.00: directly useful context for the subtask

Mark relevant=true only when relevance_score is at least 0.50 and
the context is concrete enough to use. If the context says the user
does not know, is undecided, or gives no actionable detail, mark
relevance_type="unknown_or_unusable" and relevant=false even if the
label sounds related.
Return strict JSON matching the schema.
```

H Details on LLM-as-a-Judge for Automatic Simulation Evaluation

H.1 Rubric and Prompt

Every artifact is scored on a 1–7 Likert scale matching the human study, anchored as *1 = very poor or unusable*, *4 = adequate*, *7 = excellent, highly actionable, and well grounded*. The judge (i.e. `GPT-5.5`) fills a strict-JSON `JudgeRubricScore` with the following criteria, each carrying an integer score and a short evidence string:

- `plan_quality` — coherent, useful plan for the goal.
- `tangible_result_quality` — directly usable artifacts (drafts, schedules, checklists, trackers, templates, messages, decision aids), not advice.
- `confidence_support` — helps the user feel ready to take real-world action.
- `personalization_context_grounding` — correct use of user context, respecting constraints.
- `completeness_coverage` — covers the important subtasks.
- `decomposition_quality` — breaks the goal into subtasks, with sequencing and dependencies.
- `context_curation_reuse` — elicits, preserves, selects, or reuses user context across subtasks.
- `workflow_progress_support` — supports progress across the planning workflow, including partial completions.
- `user_burden_reduction` — reduces steering, re-explaining, and cleanup the user must do.
- `specificity` — concrete and tailored rather than generic.
- `no_contradiction_hallucination` — avoids contradictions, unsupported assumptions, and fake specificity.
- `overall` — holistic score.

The final score for each method in Table 2 is the weighted average of the first nine criteria: 0.15 each for plan quality, tangible-result quality, and confidence support; 0.10 each for personalization and context grounding, completeness, decomposition quality, context curation and reuse, and workflow-progress support; and 0.05 for user-burden reduction. Specificity, contradiction avoidance, and the overall score are recorded as diagnostics and are not part of the composite.

The prompt for the LLM-judge is shown below. Specifically, judging is a two-step pipeline. First, an *evidence-extraction* call returns a `JudgeExtractedEvidence` record. This record is attached to the artifact and passed to the final scoring prompt as an audit aid. Second, the scoring prompt is conditioned on the goal, the user context, the raw artifact, and the extracted evidence, and emits the `JudgeRubricScore` above.

Prompt for LLM-Judge Evidence Extraction

```
You are extracting evidence to help evaluate an AI-generated planning
artifact.
Use the user context and raw artifact exactly as given. Extract
evidence symmetrically for any system or condition; do not favor
workflow-style outputs or single-response outputs.

Rules:
- Do not score quality here.
- Do not invent missing details.
- Include concise concrete evidence from the artifact.
- If evidence is absent, use an empty list.
- Treat placeholders, TODO fields, unsupported specifics,
contradictions, and cleanup burden as important evidence.
- A workflow artifact can show value through task decomposition,
saved context, context reuse, and partial tangible drafts, but
clutter and stale process notes are also evidence.
- A clean prose artifact can show value through directly usable
final outputs, but it should not receive credit for context reuse or
workflow memory unless the artifact actually shows it.

Return strict JSON matching the schema with these fields:
- tangible_outputs_found: concrete drafts, schedules, checklists,
trackers, surveys, messages, templates, decision aids, or other
usable artifacts.
- confirmed_user_context_used: user-provided facts or constraints
from user_context that the artifact uses correctly.
- unresolved_todos_or_placeholders: TODOs, bracketed placeholders,
missing owners/dates/details, or unresolved open decisions.
- unsupported_or_invented_details: specific names, capacities,
deadlines, venues, claims, policies, or personal details that appear
unsupported by user_context or are contradicted.
- workflow_progress_evidence: decomposition, sequencing, partial
completions, saved drafts, or multi-step progress shown in the
artifact.
- context_reuse_evidence: evidence that context was elicited,
preserved, selected, or reused across subtasks.
- user_burden_evidence: evidence that the user would need
more steering, cleanup, verification, re-explaining, or manual
reconciliation.
- final_actionability_summary: one concise sentence summarizing how
ready the artifact is for real-world action.
- extraction_warnings: caveats about ambiguity, truncation, or
evidence that could not be determined.
```

Prompt for LLM-Judge Scoring

```
You are evaluating an AI-generated planning output for a personal
goal.
The input includes the raw system output and a structured
extracted_evidence record. Use the extracted_evidence as an audit
aid, not a replacement for reading the raw output. If the raw output
and extracted evidence conflict, trust the raw output and mention the
conflict in criterion evidence when relevant.
Score the output from 1 to 7 on each criterion, matching the human
study Likert scale.
1 = very poor, harmful, or unusable.
2 = weak.
3 = partial but generic or incomplete.
4 = adequate.
5 = good and useful.
6 = strong, specific, and actionable.
7 = excellent, highly actionable, and well grounded.

Important:
- Longer is not automatically better.
- Reward relevant personalization, not verbosity.
- Penalize contradictions with user-provided context.
- Penalize unsupported assumptions.
- Penalize generic advice when the goal requires concrete artifacts.
- Evidence should be concise and cite concrete details from the
output.
- Use extracted_evidence to notice tangible outputs, context use,
unresolved placeholders, unsupported details, workflow progress,
context reuse, and user burden. Apply the same standard to every
condition.

Criteria:
- plan_quality: Does the output produce a coherent, useful plan for
the user's goal?
- tangible_result_quality: Does it include directly usable artifacts
such as drafts, schedules, checklists, templates, trackers, surveys,
messages, or decision aids?
- confidence_support: Would this output make the user more confident
about taking real-world action?
- personalization_context_grounding: Does it use the user's stated
context correctly and avoid ignoring important constraints?
- user_burden_reduction: Does it reduce the amount of steering,
re-explaining, manual context management, and cleanup the user must
do?

Match the real study protocol:
- Users had 25 minutes to create a plan and tangible results.
- Reward outputs that help the user get started, not outputs that
merely describe what could be done.
- If the conversation is messy or incomplete, score the artifact
actually produced, not an idealized answer.
- Do not reward verbosity by itself; an output can be long and still
poor if it is generic, repetitive, or hard to act on.
- For workflow summaries, treat task trees, saved contexts,
selected/reused contexts, and working drafts as part of the
artifact's value.

Return only strict JSON matching the provided schema.
```

H.2 Validation

We validate the LLM judge using the within-subject JumpStarter–ChatGPT pairs from the human study. For each de-identified system output, the judge scores the same artifacts shown to human raters, and we compare its scores against human ratings of plan quality, tangible-result quality, and confidence to act on a 1–7 scale.

We assess judge validity using three pre-specified checks: (i) Spearman correlation with the human-rated `overall` construct, (ii) mean absolute error (MAE) on the 1–7 scale, and (iii) whether the `study_quality` composite recovers the human-study winner within each JumpStarter–ChatGPT pair.

The judge passes all validation gates. It achieves `spearman_overall` = 0.779, exceeding our threshold of $\rho \geq 0.7$, and `mae_overall` = 1.064, approximately within one Likert point. The

mean within-pair margin is also positive (`study_quality_mean_margin` = 1.05), indicating that the judge consistently favors the same condition as the human-study ratings. These results suggest that the judge's scores are well aligned with human ratings and preserve the direction of human-study preferences.

I Detailed Baseline Methods Descriptions

We compare JumpStarter against the following baseline conditions.

ChatGPT baselines. We include three ChatGPT baselines using `GPT-4o` in a multi-turn planning setting, mirroring the user study condition in which participants planned with the ChatGPT interface. In *vanilla ChatGPT*, the simulated user directly interacts with the model to plan the goal. In *ChatGPT + elicited context*, the simulated user first answers global context questions before beginning the planning conversation. In *ChatGPT + structured summary*, the elicited context is organized into a structured summary and provided to the model before planning begins. We also include *single-turn decomposition*, which receives the same elicited context and is explicitly prompted to decompose the goal into subtasks and draft each one in a single turn, without an external task structure, per-subtask context selection, or reuse.

Single-turn decomposition results. Explicitly prompting `GPT-4o` to decompose the goal and draft every subtask in a single turn raises the score from 3.70 (ChatGPT + elicited context) to 4.05, yet *JumpStarter-Shallow* still outperforms this single-turn decomposition baseline (+0.22; Table 2). The advantage lies in context curation and reuse (+1.36 on the judge's 1–7 scale) and personalization (+0.66), while decomposition quality is similar (+0.12), suggesting that `GPT-4o` decomposes goals well on its own but cannot attach the right context to each subtask in a single turn.

Planning and memory-agent baselines. We include representative baselines that capture alternative agentic approaches to complex planning. *ADaPT* (Prasad et al., 2024) represents recursive task decomposition for LLM planning. *Ask-Before-Plan* (Zhang et al., 2024) represents clarification-first planning, where the model asks questions before generating a plan. We also include an unstructured memory-RAG baseline inspired by memory-augmented agents such as MemGPT (Packer et al., 2023), where relevant prior context is retrieved from a flat memory store and provided to the model during planning. Finally, the *integrated agentic planner* combines these capabilities in one loop: it runs *JumpStarter*'s global elicitation step, retrieves from a memory store built from the user's materials, queries a simulated browse tool that returns generic guidance for the goal's deliverables, decomposes the goal with *JumpStarter*'s subtask-generation prompt, and selects up to five relevant context items for each subtask before drafting. Unlike *JumpStarter*, it neither elicits context for individual subtasks nor reuses drafts across subtasks. These baselines test whether *JumpStarter*'s gains can be achieved by decomposition, clarification, retrieval, or their combination, rather than by organizing elicited and reused context around task structure.

Integrated agentic planner results. The integrated agentic planner scores 3.94. *JumpStarter-Shallow* outperforms it by +0.33 (95% CI [0.28, 0.38]), and *JumpStarter-Recursive*, rerun alongside the agent, by +0.26 [0.20, 0.31]; the rerun matches the main run (difference -0.004 [-0.06, 0.05]). The agent is not short of content: it writes as many words as *JumpStarter-Recursive* (1,100 vs. 1,071) and ties it on tangible-result quality and completeness. Its deficit is concentrated in context curation and reuse (+0.58 for *JumpStarter-Shallow*, +0.73 for *JumpStarter-Recursive*) and in personalization and context grounding (+0.63, +0.57): it has the user's elicited context but does not attach it to the right subtask or carry it forward. In the traces, the agent never routes a saved draft into a later subtask (0 reuse events per session, vs. 1.48 for *JumpStarter*) and elicits less context (3.0 items per session, vs. 6.96), since it asks only the global questions. *JumpStarter-Recursive*'s margin over the agent is largest on multi-track goals (+0.32, vs. +0.23 on high- and +0.24 on medium-complexity goals), where subtasks depend on each other most.

Workflow ablations and context-control baselines. We include workflow ablations to isolate the contribution of each component in task-structured context curation. Each ablation removes or replaces one mechanism of the *JumpStarter* workflow with recursive decomposition enabled. *Random context selection* replaces task-local selection with a randomly sampled context subset, testing whether any context reduction is sufficient. *No context selection* removes the selection step

and passes all available context to every subtask, testing whether performance improves simply by exposing more information. *No context reuse* prevents saved drafts and intermediate artifacts from being carried forward, and *no elicitation* removes proactive context gathering. Finally, *JumpStarter-Recursive* uses the same context-curation mechanisms as *JumpStarter-Shallow* but allows subtasks to be decomposed further. Together, these conditions test whether gains come from selecting relevant context, reusing intermediate artifacts, eliciting missing information, and using lightweight task structure rather than indiscriminately adding context or increasing decomposition depth.

J LLM Prompts

J.1 Context Elicitation

J.1.1 Goal Initialization and Global Context

Figure 11 details the prompts used for eliciting global context for the goal initialization process introduced in Section B.1.

System prompt

You are a helpful assistant in generating at most three questions to elicit more context from the user in order to accomplish a task the user is involved. You will be given the main purpose of the task, and the current context history from the user. You need to judge if the current context information is enough to do this task. If not, what existing doc do you think they already have and can be provided as the context document for this task? Alternatively, what existing most important extra information do you think they should provide as the context information for this task? Please try your best to start by asking the potential existing doc first, and then ask for the potential info. Do not ask the question that can be possibly answered from the suggested doc in the first question. You can also just ask for the potential info if no doc is needed from the user. Please directly generate the questions for users to answer, provide the reason for the question, identify if it is DOC or INFO, and provide a name for the question. Otherwise, output "Ready".

Here are two examples:
Input: My user has a main purpose: Apply for a drivers license by 12/15/2024. The current context history from the user is empty.
Output:
1. Reason: Driver's license requirements vary significantly depending on the location. Knowing the specific state or country would allow for tailored advice regarding local rules, tests, and documentation required -> Question: Which state or country are you applying for? -> Type: INFO -> Name: State or Country
2. Reason: Different age groups may have different requirements or steps in the licensing process. For example, minors often have to go through graduated license programs. -> Question: How old are you -> Type: INFO -> Name: Age of the User

Input: My user has a main purpose: Apply for a PhD program by 12/15/2024. The current context history from the user is empty.
Output:
1. Reason: Curriculum Vitae (CV) or Resume could be helpful as it would likely contain detailed information about their educational background and any research experiences or academic achievements, which are critical for applying to PhD programs. -> Question: What CV or Resume can you provide for the PhD application? -> Type: DOC -> Name: CV or Resume

Now, start prediction:

User Prompt

Input: My user has a main purpose: {task_input}. The current context history from the user is empty.

Figure 11: Prompts for Context Elicitation for Goal Input.

J.1.2 Answer Draft Creation and Iteration

Figure 12 presents the prompts used to generate an answer draft for each subtask.

My user has a main purpose: {main_purpose}. My user is working on the task {task_name}: {task_description}. The current context history from the user is {context_history}. Please judge if the current context information is enough to do this task. If not, what existing most important extra information do you think they should provide as the context information for this task? Please directly generate questions for users to answer as extra info. Otherwise, output "Ready"
Format the response like this: 1. <question 1> : <reason for asking question 1> -> title of question 1
2. <question 2> : <reason for asking question 2> -> title of question 2
3. <question 3> : <reason for asking question 3> -> title of question 3

Figure 12: Prompts for Context Elicitation for Answer Draft Creation.

J.2 Context Selection

J.2.1 Answer Draft Generation

Figure 13 specifies the prompts used to select relevant context for generating an answer draft.

System prompt

Given the user's main purpose and the task they are working on, select the most relevant context keys from the current context history that can be used to draft good responses for the user to complete the task. Please also provide explanations. \n The current context history is shown as one or more key-value pairs. Please select only the keys from the 'key' part of the context history. Do not select the keys from the 'value' part of the context history \n Format the response like this: number. <context_keys>: <reasons for selecting context_keys>. Replace the context_keys with the actual keys as shown in the context history. Please directly give the answers and do not provide extra summarization sentences at the end.

User Prompt

My user has a main purpose: {main_purpose}. My user is working on the task {task_name}: {task_description}. Here is the current context history in JSON format (with 'key':'value' pairs) from the user: {context_history}

Figure 13: Prompts for Context Selection for Answer Draft Creation.

J.2.2 Task Forking

The prompt to conduct the task forking is as follows:

- *My user has a main purpose: {main_purpose}. My user is working on the task {task_name}: {task_description}. My user needs to break down the task into sub-tasks. Here is the current context history from the user: {context_history}. Please select the most relevant context key from the current context history that can be used to better decompose the current task into several sub-tasks for the user to get started. Do not help the user to break down the task. Please also provide explanations. Format the response like this: <context_key>: <reasons>. Replace the context_key with the actual key in the context history.*

J.3 Task Decomposition

This section presents the prompts used for task decomposition, organized into three core components: Subtask Generation (Figure 14), Subtask Detection (Figure 15), and Task Forking (Figure 16).

My user has a main purpose:{main purpose}.
Please consider the following context information from my user: {user_context}

Please break down the task below into three to six manageable subtasks: {current task}
The existing step structure is shown as follows: {existing tree step structure}.

Please directly give the response that fills in the current subtask: {current task} in the provided task structure.

Format the response like this: 1. [Duration for subtask1] {subtask1 title}: {subtask1 detailed description} 2. [Duration for subtask2] {subtask2 title}: {subtask2 detailed description} 3. [Duration for subtask3] {subtask3 title}: {subtask3 detailed description}.
Please specify the duration for each subtask in terms of days, weeks or months. For example, [1 week], [2-4 weeks], [1 month], and [1-2 months]. Please do not include other texts for duration such as [Ongoing]. Please do not include ** in the subtask title. Please directly give the response and do not start with "{current subtask title}:"

Figure 14: The prompt for Subtask Generation.

System prompt

You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Specifically, an overall goal is decomposed into a tree of subtasks. Each of these tasks is positioned on the tree structure and is associated with a level. Based on the user prompt, and the current level of the task node on the tree, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition. Please also provide explanations for your choice.
Here are some examples:
User: My user is working on the task Research on Prospective Companies and Positions: Conduct a comprehensive search on potential companies and specific research scientist internship positions in the NLP field. Understand what each role entails, identify skill requirements, and evaluate how they align with your research interests. The current node level of the task is 1. My user needs to know if the current task is specific and actionable
Reason: this task needs to be further decomposed as it is positioned on the first level of the tree and it involves more than one deliverables: search on companies and search on positions. To complete this task, there are multiple subtasks that need to be done separately on the next level of the tree. These include conducting a comprehensive search on potential companies, searching for specific research scientist internship positions, an analysis of what each role entails, identifying skill requirements, and evaluation of alignment with the user's research interests.
Answer: Yes

User: My user is working on the task Identify Potential Universities: Create a list of universities that offer PhD programs in HCI. The selection can be based on factors such as reputation, HCI research focus, published HCI research papers, faculty expertise etc. The current node level of the task is 2. My user needs to know if the current task is specific and actionable.
Reason: This task does not need to be further decomposed as it is positioned on the second level of the tree and it just involves one deliverable: create a list of schools that offer PhD programs in HCI. Although it may require several steps to create the list, the end goal of this task is to get a list. Therefore the task is actionable.
Answer: No

User: My user is working on the task Identify Required Documents: Reseach and confirm all the necessary documents required for the non-driver ID application, ensuring to list all forms of acceptable proofs such as a birth certificate or passport for identity, Social Security Card or W-2 form for Social Security number, and utility bills or lease agreement for proof of residency. The current node level of the task is 1. My user needs to know if the current task is specific and actionable.
Reason: Although the task is positioned on the first level of the tree, the task does not need to be further decomposed as the primary goal of the task - identify necessary documents for the non-driver ID application - can be viewed as a singular, cohesive deliverable. Despite involving various types of documents, the task is focused on compiling a comprehensive list, which makes it actionable as a single unit. The distinction lies in the focus on gathering all necessary documentation, a clear and direct objective.
Answer: No

Now, let's start prediction:

User Prompt

User: My user is working on the task {task_title}: {task_descriptions}. The current node level of the task is {tree_level}. My user needs to know if the current task needs to be decomposed

Figure 15: The prompt for Subtask Detection.

J.4 Answer Draft Creation

The prompt to generate the answer drafts is shown as follows:

- *My user has a main purpose: {main purpose}. Please consider the following context information from my user: {user_context}. My user needs help with the current task {current task}: {task description}*

J.5 Prompts for Technical Evaluation of Subtask Detection

J.5.1 Zero-shot Prompting

The prompt for zero-shot for the task of subtask detection is demonstrated below:

- System prompt: *You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Based on the user prompt, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition.*

System prompt

Given the queried task, determine if a "for" loop is needed to complete the task. You will be given a question Q. Please provide the reasoning and then respond with "Yes" or "No".
Here are some examples:
Q: Research the specific HCI PhD programs at each university from the initial list. Focus on aspects such as program curriculum, research opportunities, faculty expertise, and available resources.
Reason: The task requires a "for" loop to complete as there already exists an initial list of entities (i.e. universities) to research. Specifically, the goal of this task is to research the program curriculum, research opportunities, faculty expertise, and available resources for each university from the initial list. It is not possible to complete the task directly without a "for" loop.
A: Yes

Q: Make a list of potential recommenders including former supervisors, academic advisors, and professors who are familiar with your academic and research abilities.
Reason: The task does not require a "for" loop to complete as there does not exist a list of potential recommenders. The goal of this task therefore is to construct the list of recommenders based on certain criteria.
A: No

Q: Reach out to the individuals on your list via email or phone, providing them with the necessary documents and details about the HCI programs, and formally request their letters of recommendation
Reason: This task needs a "for" loop to complete as you have already obtained your list of individuals. You should reach out to each entity (i.e. individuals) on the list to complete the task.
A: Yes

Q: Gather information on different universities offering PhD programs in Human-Computer Interaction. Create an initial list based on general information such as program recognition, location, and basic offerings.
Reason: The task does not require a "for" loop to complete as there does not exist a list of universities offering PhD programs in HCI that can be used to iterate on.
A: No

User Prompt

Q: {task_description}

Figure 16: The prompt for Task Forking.

- User Prompt: *My user is working on the task {task title}: {task description}. My user needs to know if the current task needs to be decomposed.*

J.5.2 Few-shot Prompting

The prompt for few-shot-only prompting is shown in Figure 17. Note that we used three in-context examples in the prompt.

J.5.3 Few-shot + CoT

We constructed the prompt in a Chain-of-Thought fashion, where GPT-4 is instructed to first generate the reasoning and then the answer. The prompt is shown in Figure 18.

J.5.4 Few-shot + CoT + Draft

We experimented with incorporating both CoT and the initial working solution draft into the prompt. The system prompt is shown in Figure 19. For the user prompt, before detecting subtasks. we first generated the initial working solution draft for the current task. The user prompt is shown below:

- *My user is working on the task {task title}: {task description}. The GPT response to the task is: {Draft}. My user needs to know if the current task is specific and actionable.*

System prompt

You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Based on the user prompt, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition
Here are some examples:
User: My user is working on the task Research on Prospective Companies and Positions: Conduct a comprehensive search on potential companies and specific research scientist internship positions in the NLP field. Understand what each role entails, identify skill requirements, and evaluate how they align with your research interests. My user needs to know if the current task is specific and actionable
Answer: Yes

User: My user is working on the task Identify Potential Universities: Create a list of universities that offer PhD programs in HCI. The selection can be based on factors such as reputation, HCI research focus, published HCI research papers, faculty expertise etc. My user needs to know if the current task is specific and actionable.
Answer: No

User: My user is working on the task Identify Required Documents: Reseach and confirm all the necessary documents required for the non-driver ID application, ensuring to list all forms of acceptable proofs such as a birth certificate or passport for identity, Social Security Card or W-2 form for Social Security number, and utility bills or lease agreement for proof of residency. My user needs to know if the current task is specific and actionable.
Answer: No

Now, let's start prediction:

User prompt

My user is working on the task {task title}: {task description}. My user needs to know if the current task needs to be decomposed.

Figure 17: The few-shot-only prompt for Subtask Detection.

J.5.5 Few-shot + CoT + Tree + Draft

To construct the system prompt for this setting, we incorporate the tree level of each task into the prompt. The prompt is shown in Figure 20. Additionally, for the current task at hand, its tree-level information is also presented in the user prompt, as shown below:

- *My user is working on the task {task title}: {task description}. The current node level of the task is {level}. The GPT response to the task is: {Draft}. My user needs to know if the current task is specific and actionable.*

System prompt

You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Based on the user prompt, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition. Please also provide explanations for your choice.
Here are some examples:
User: My user is working on the task Research on Prospective Companies and Positions: Conduct a comprehensive search on potential companies and specific research scientist internship positions in the NLP field. Understand what each role entails, identify skill requirements, and evaluate how they align with your research interests. My user needs to know if the current task is specific and actionable
Reason: this task needs to be further decomposed as it involves more than one deliverables: search on companies and search on positions. To complete this task, there are multiple subtasks that need to be done separately. These include conducting a comprehensive search on potential companies, searching for specific research scientist internship positions, an analysis of what each role entails, identifying skill requirements, and evaluation of alignment with the user's research interests.
Answer: Yes

User: My user is working on the task Identify Potential Universities: Create a list of universities that offer PhD programs in HCI. The selection can be based on factors such as reputation, HCI research focus, published HCI research papers, faculty expertise etc. My user needs to know if the current task is specific and actionable.
Reason: This task does not need to be further decomposed as it just involves one deliverable: create a list of schools that offer PhD programs in HCI. Although it may require several steps to create the list, the end goal of this task is to get a list. Therefore the task is actionable.
Answer: No

User: My user is working on the task Identify Required Documents: Reseach and confirm all the necessary documents required for the non-driver ID application, ensuring to list all forms of acceptable proofs such as a birth certificate or passport for identity, Social Security Card or W-2 form for Social Security number, and utility bills or lease agreement for proof of residency. My user needs to know if the current task is specific and actionable.
Reason: The primary goal of the task is to identify necessary documents for the non-driver ID application, which can be viewed as a singular, cohesive deliverable. Despite involving various types of documents, the task is focused on compiling a comprehensive list, which makes it actionable as a single unit. The distinction lies in the focus on gathering all necessary documentation, a clear and direct objective.
Answer: No

Now, let's start prediction:

User prompt

My user is working on the task {task title}: {task description}. My user needs to know if the current task needs to be decomposed.

Figure 18: The prompt for few-shot + CoT for Subtask Detection.

System prompt

You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Based on the user prompt, and the GPT response, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition. Please also provide explanations for your choice.

Here are some examples:
User: My user is working on the task Research on Prospective Companies and Positions: Conduct a comprehensive search on potential companies and specific research scientist internship positions in the NLP field. Understand what each role entails, identify skill requirements, and evaluate how they align with your research interests. The GPT response to the task is:
Sure, I will list some promising companies that offer research scientist internship positions in Natural Language Processing (NLP) and briefly describe what each role entails:

1. **Google AI**:
  - *Role*: As a research intern, you can work in areas such as machine intelligence, machine perception, natural language processing, and quantum computing. Google typically looks for research experience, ability to design and execute a research agenda, and coding skills.

2. **Facebook AI Research (FAIR)**:
  - *Role*: Natural Language Processing Internship roles at FAIR involve developing novel techniques for NLP tasks which include generating human-like text. They require experience with machine learning, AI, deep learning, dialogue management, and computer vision.

3. **Amazon Alexa AI**:
  - *Role*: Interns in this role generally work on language understanding capabilities of Alexa, advancing state-of-the-art on tasks like entity resolution or text classification. Knowledge in machine learning, especially deep learning for NLP, is key for this role.

4. **OpenAI**:
  - *Role*: Research internships at OpenAI are highly selective. Interns are expected to have published in machine learning or AI and can spend a significant portion of their time on original research.

5. **IBM Watson**:
  - *Role*: Watson Interns can work on various AI/NLP tasks from fact extraction, relation extraction, disambiguation, to summarization, based on the team you join. A strong AI/ML foundation and coding skills are required.

6. **Apple AI/ML**:
  - *Role*: Apple offers internships where you can work on Siri's language understanding capabilities. They typically require machine learning knowledge and experience, especially in deep learning.

7. **Microsoft Research**:
  - *Role*: Interns in this role could work on a variety of NLP challenges, including developing algorithms that enable, for example, conversational interactions with AI. Strong machine learning knowledge, particularly in deep learning, is expected.

8. **Baidu Research**:
  - *Role*: Interns can work on various novel NLP tasks such as language understanding, language generation, and more. Baidu looks for individuals with robust knowledge in machine learning, deep learning, and coding abilities.

9. **NVIDIA - AI Research**:
  - *Role*: Interns have an opportunity to work on projects in the realms of machine learning, deep learning, computer vision, and natural language processing. They seek candidates with strong ML and DL backgrounds.

When selecting a specific position, I recommend considering the following:
- Does the role align with your career objectives?
- Does the role play to your strengths or offer opportunities to develop in areas of interest?
- Does the role match your skill set? If not, can you acquire those skills?

Remember, a role can be an opportunity to learn and grow, so don't be deterred if you lack a few skills. If the role excites you, go for it!
My user needs to know if the current task is specific and actionable
Reason: this task needs to be further decomposed as the GPT response is not good enough to help user complete the task. It involves more than one deliverables: search on companies and search on positions. To complete this task, there are multiple subtasks that need to be done separately. These include conducting a comprehensive search on potential companies, searching for specific research scientist internship positions, an analysis of what each role entails, identifying skill requirements, and evaluation of alignment with the user's research interests.
Answer: Yes

User: My user is working on the task Identify Potential Universities: Create a list of universities that offer PhD programs in HCI. The selection can be based on factors such as reputation, HCI research focus, published HCI research papers, faculty expertise etc. The GPT response to the task is:
Here is a list of universities that offer PhD programs in HCI:
1. Stanford University, USA
2. Massachusetts Institute of Technology (MIT), USA
3. Carnegie Mellon University (CMU), USA
4. University of California, Berkeley, USA
5. University of Washington, USA
6. University of Michigan, USA
7. University of California, San Diego, USA
8. Cornell University, USA
9. University of Toronto, Canada
10. University of College London, UK
My user needs to know if the current task is specific and actionable.
Reason: The GPT response is good enough and it provides direct answer to complete the task. Additionally, this task does not need to be further decomposed as it just involves one deliverable: create a list of schools that offer PhD programs in HCI. Although it may require several steps to create the list, the end goal of this task is to get a list. Therefore the task is actionable.
Answer: No

User: My user is working on the task Identify Required Documents: Reseach and confirm all the necessary documents required for the non-driver ID application, ensuring to list all forms of acceptable proofs such as a birth certificate or passport for identity, Social Security Card or W-2 form for Social Security number, and utility bills or lease agreement for proof of residency. The GPT response to the task is:
To apply for a non-driver ID in [redacted] you need to provide several documents to prove your identity, Social Security Number (SSN), and residency. Below, I have listed the acceptable documents for each category based on the latest guidelines provided by the [redacted] Department of Motor Vehicles (DMV).

1. Proof of Identity and Date of Birth (must total 6 points according to [redacted] DMV's point system):
  - U.S. passport or U.S. passport card (valid or expired within the past five years) – 4 points
  - Birth certificate (U.S., Puerto Rican, Canadian, or Mexican) - 2 points
  - Employment Authorization Card (I-766) – 3 points
  - Permanent Resident Card (I-551) – 3 points
  - Foreign passport with valid U.S. Visa and I-94 – 3 points

  Note: At least one document must show your date of birth and at least one document must have your full Social Security Number.

2. Proof of Social Security Number:
  - Social Security card
  - W-2 form
  - SSA-1099 form
  - Non-SSA-1099 form
  - Pay stub with your name and SSN on it

3. Proof of [redacted] Residency (at least two documents):
  - State or federal tax return
  - Utility bill (must be dated within the last 90 days)
  - Lease agreement or mortgage statement
  - Bank statement (must be dated within the last 90 days)
  - Any mail from any government agency (dated within the last 90 days)

Additional Notes:
- All documents must be originals or certified copies.
- If your name has changed due to marriage, divorce, or court order, you will need to provide documents that link your current name to your former name.
- You must also complete an "Application for Permit, Driver License or Non-Driver ID Card" (Form MV-44).

Before going to the DMV, it's recommended to use the [redacted] DMV Document Guide which provides a personalized checklist of the required documents based on the specifics of your application. This can be found on the official [redacted] DMV website.

Make sure to double-check with the DMV for the latest requirements or any specific circumstances that may affect your application to ensure a smooth process.
My user needs to know if the current task is specific and actionable.
Reason: The task does not need to be further decomposed for two reasons. The first reason is that the GPT response provides direct answer to the task and generates a list of all forms of acceptable proofs for the documents required to get a non-driver ID. The second reason is that the primary goal of the task - identify necessary documents for the non-driver ID application - can be viewed as a singular, cohesive deliverable. Despite involving various types of documents, the task is focused on compiling a comprehensive list, which makes it actionable as a single unit. The distinction lies in the focus on gathering all necessary documentation, a clear and direct objective.
Answer: No

Now, let's start prediction:

Figure 19: The prompt for few-shot + CoT + Draft for Subtask Detection.

You are a useful assistance to detect if the current task needs to be further decomposed if it is not actionable and the primary goal of the task can not be viewed as a singular, distinctive deliverable. Specifically, the task is positioned on a tree structure and is associated with a level. Based on the user prompt, the current node level of the task on the tree, and the GPT response, please output Yes if it needs to be decomposed; No otherwise meaning it is actionable and does not require task decomposition. Please also provide explanations for your choice.

Here are some examples:
User: My user is working on the task Research on Prospective Companies and Positions: Conduct a comprehensive search on potential companies and specific research scientist internship positions in the NLP field. Understand what each role entails, identify skill requirements, and evaluate how they align with your research interests. The current node level of the task is 1. The GPT response to the task is:
Sure, I will list some promising companies that offer research scientist internship positions in Natural Language Processing (NLP) and briefly describe what each role entails:

1. **Google AI**:
  - *Role*: As a research intern, you can work in areas such as machine intelligence, machine perception, natural language processing, and quantum computing. Google typically looks for research experience, ability to design and execute a research agenda, and coding skills.

2. **Facebook AI Research (FAIR)**:
  - *Role*: Natural Language Processing Internship roles at FAIR involve developing novel techniques for NLP tasks which include generating human-like text. They require experience with machine learning, AI, deep learning, dialogue management, and computer vision.

3. **Amazon Alexa AI**:
  - *Role*: Interns in this role generally work on language understanding capabilities of Alexa, advancing state-of-the-art on tasks like entity resolution or text classification. Knowledge in machine learning, especially deep learning for NLP, is key for this role.

4. **OpenAI**:
  - *Role*: Research internships at OpenAI are highly selective. Interns are expected to have published in machine learning or AI and can spend a significant portion of their time on original research.

5. **IBM Watson**:
  - *Role*: Watson Interns can work on various AI/NLP tasks from fact extraction, relation extraction, disambiguation, to summarization, based on the team you join. A strong AI/ML foundation and coding skills are required.

6. **Apple AI/ML**:
  - *Role*: Apple offers internships where you can work on Siri's language understanding capabilities. They typically require machine learning knowledge and experience, especially in deep learning.

7. **Microsoft Research**:
  - *Role*: Interns in this role could work on a variety of NLP challenges, including developing algorithms that enable, for example, conversational interactions with AI. Strong machine learning knowledge, particularly in deep learning, is expected.

8. **Baidu Research**:
  - *Role*: Interns can work on various novel NLP tasks such as language understanding, language generation, and more. Baidu looks for individuals with robust knowledge in machine learning, deep learning, and coding abilities.

9. **NVIDIA - AI Research**:
  - *Role*: Interns have an opportunity to work on projects in the realms of machine learning, deep learning, computer vision, and natural language processing. They seek candidates with strong ML and DL backgrounds.

When selecting a specific position, I recommend considering the following:
- Does the role align with your career objectives?
- Does the role play to your strengths or offer opportunities to develop in areas of interest?
- Does the role match your skill set? If not, can you acquire those skills?

Remember, a role can be an opportunity to learn and grow, so don't be deterred if you lack a few skills. If the role excites you, go for it!
My user needs to know if the current task is specific and actionable
Reason: this task needs to be further decomposed as the GPT response is not good enough to help user complete the task. Moreover, it is positioned on the first level of the tree and it involves more than one deliverables: search on companies and search on positions. To complete this task, there are multiple subtasks that need to be done separately on the next level of the tree. These include conducting a comprehensive search on potential companies, searching for specific research scientist internship positions, an analysis of what each role entails, identifying skill requirements, and evaluation of alignment with the user's research interests.
Answer: Yes

User: My user is working on the task Identify Potential Universities: Create a list of universities that offer PhD programs in HCI. The selection can be based on factors such as reputation, HCI research focus, published HCI research papers, faculty expertise etc. The current node level of the task is 2. The GPT response to the task is:
Here is a list of universities that offer PhD programs in HCI:
1. Stanford University, USA
2. Massachusetts Institute of Technology (MIT), USA
3. Carnegie Mellon University (CMU), USA
4. University of California, Berkeley, USA
5. University of Washington, USA
6. University of Michigan, USA
7. University of California, San Diego, USA
8. Cornell University, USA
9. University of Toronto, Canada
10. University of College London, UK
My user needs to know if the current task is specific and actionable.
Reason: The GPT response is good enough and it provides direct answer to complete the task. Additionally, this task does not need to be further decomposed as it is positioned on the second level of the tree and it just involves one deliverable: create a list of schools that offer PhD programs in HCI. Although it may require several steps to create the list, the end goal of this task is to get a list. Therefore the task is actionable.
Answer: No

User: My user is working on the task Identify Required Documents: Reseach and confirm all the necessary documents required for the non-driver ID application, ensuring to list all forms of acceptable proofs such as a birth certificate or passport for identity, Social Security Card or W-2 form for Social Security number, and utility bills or lease agreement for proof of residency. The current node level of the task is 1. The GPT response to the task is:
To apply for a non-driver ID in [redacted] you need to provide several documents to prove your identity, Social Security Number (SSN), and residency. Below, I have listed the acceptable documents for each category based on the latest guidelines provided by the [redacted] Department of Motor Vehicles (DMV).

1. Proof of Identity and Date of Birth (must total 6 points according to [redacted] DMV's point system):
  - U.S. passport or U.S. passport card (valid or expired within the past five years) – 4 points
  - Birth certificate (U.S., Puerto Rican, Canadian, or Mexican) - 2 points
  - Employment Authorization Card (I-766) – 3 points
  - Permanent Resident Card (I-551) – 3 points
  - Foreign passport with valid U.S. Visa and I-94 – 3 points

  Note: At least one document must show your date of birth and at least one document must have your full Social Security Number.

2. Proof of Social Security Number:
  - Social Security card
  - W-2 form
  - SSA-1099 form
  - Non-SSA-1099 form
  - Pay stub with your name and SSN on it

3. Proof of [redacted] Residency (at least two documents):
  - State or federal tax return
  - Utility bill (must be dated within the last 90 days)
  - Lease agreement or mortgage statement
  - Bank statement (must be dated within the last 90 days)
  - Any mail from any government agency (dated within the last 90 days)

Additional Notes:
- All documents must be originals or certified copies.
- If your name has changed due to marriage, divorce, or court order, you will need to provide documents that link your current name to your former name.
- You must also complete an "Application for Permit, Driver License or Non-Driver ID Card" (Form MV-44).

Before going to the DMV, it's recommended to use the [redacted] DMV Document Guide which provides a personalized checklist of the required documents based on the specifics of your application. This can be found on the official [redacted] DMV website.

Make sure to double-check with the DMV for the latest requirements or any specific circumstances that may affect your application to ensure a smooth process.
My user needs to know if the current task is specific and actionable.
Reason: Although the task is positioned on the first level of the tree, the task does not need to be further decomposed for two reasons. The first reason is that the GPT response provides direct answer to the task and generates a list of all forms of acceptable proofs for the documents required to get a non-driver ID. The second reason is that the primary goal of the task - identify necessary documents for the non-driver ID application - can be viewed as a singular, cohesive deliverable. Despite involving various types of documents, the task is focused on compiling a comprehensive list, which makes it actionable as a single unit. The distinction lies in the focus on gathering all necessary documentation, a clear and direct objective.
Answer: No

Now, let's start prediction:

Figure 20: The prompt for few-shot + CoT + Tree + Draft for Subtask Detection.