PERSONALWORKENV: EVALUATING AND TRAINING PERSONALIZED AI ASSISTANTS WITH SYNTHETIC WORK ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A personal work assistant can complete a task while violating a user-specific norm revealed in an earlier interaction. We call this failure mode the *personalization gap* and introduce `PersonalWorkEnv`, a synthetic environment-generation pipeline for evaluating and training *work personalization*: remembering and applying previously revealed work norms to later tool-produced artifacts. Each environment combines a user persona, six simulated workdays, roughly twenty tasks, simple CLI tools, and ten progressively introduced profiles, including temporary norms that are later replaced. Deterministic validators and rubric-based soft checks distinguish surface task completion from profile compliance. We instantiate `PersonalWorkEnv-50`, a double-audited suite of fifty environments, and find a substantial personalization gap across seven proprietary and open models, with larger gaps on day 6 than day 1. As a baseline training use case, mixed-curriculum RL improves small Qwen3.5 models' all-check success. Conditional-compliance diagnostics show that the 9B gains primarily reflect better surface execution, whereas the 35B-A3B model also improves profile compliance among surface-successful tasks.

1 INTRODUCTION

Today's task (day 4): *"Email Prof. Smith to set up a meeting tomorrow at 3pm to discuss the Q2 paper."*

`MEMORY.md` (loaded into context):
```
- User in Tokyo (JST, UTC+9)
  until Friday.
- Default email tone: formal.
```

Evaluator-only profile P4:
Meeting times must be written in the user's current local timezone.

Agent action:
```
email_cli.send(to="smith@univ.edu", body="Let's meet at 3pm EST tomorrow.")
```

✓ email sent successfully ✗ timezone profile violated

Figure 1: An illustrative work-personalization failure, adapted from an Opus 4.5 trajectory (Section 3). The email is sent successfully but uses EST despite a memory note placing the user in Tokyo. The evaluator-only profile is shown for the reader; it is not an extra instruction to the agent.

Personal AI assistants nowadays are widely used to perform tool-calls and accomplish everyday tasks, from sending emails, scheduling meetings, to editing documents. Completing these actions is not enough: the artifacts must respect work norms learned earlier, such as timezone conventions, sign-offs, and confidentiality rules. Figure 1 shows a surface-successful email that violates a remembered timezone norm—even though the relevant information is already in the agent's context.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We define **work personalization** as the ability of a tool-using assistant to remember and apply user-specific work norms revealed through prior interactions. These norms concern content, format, tone, privacy, routing, timezones, naming, or collaborator-specific conventions. This is narrower than open-ended human personalization: we do not attempt to model all implicit tastes, negotiate ambiguous values, or predict deployment-level user satisfaction. The target is a concrete longitudinal failure mode: a norm is surfaced during ordinary work, is not repeated in every subsequent request, and should nevertheless constrain later artifacts. We call the difference between surface success and success on all task and profile checks the *personalization gap*.

Existing web, computer, and tool-use benchmarks establish stateful execution and executable evaluation (Zhou et al., 2023; Drouin et al., 2024; Xie et al., 2024; Yao et al., 2024; Trivedi et al., 2024; Lu et al., 2024). Personalization and memory benchmarks study user histories, dynamic profiles, and preference-aware responses (Salemi et al., 2024; Kumar et al., 2024; Tan et al., 2025; Jiang et al., 2025; Liu et al., 2026; Kwon et al., 2025). Closer tool-agent settings include APOLLO, ASTRA-bench, and KnowU-Bench (Chen et al., 2026b; Xiu et al., 2026; Chen et al., 2026a). Our contribution is not tools, state, or preferences individually, but a controlled combination: longitudinal work artifacts, progressive profile acquisition and replacement, profile-attached checks, and separate measurements of surface success and profile failure (Table 4). Simple CLI interfaces deliberately reduce tool-interface difficulty so that this distinction remains visible.

We introduce `PersonalWorkEnv` as an environment-generation pipeline and an evaluation/training substrate. A supervising orchestrator coordinates agents that construct personas, tools, tasks, workspace seeds, validators, and feedback rubrics, with automated validity and leakage checks. The resulting `PersonalWorkEnv-50` suite is one frozen configuration of this pipeline, not a claim to exhaust personalization or real work. Its fixed six-day schedule controls acquisition and retention load across models; the generation framework is not restricted to that cadence. Our contributions are:

- **A measurable work-personalization setting.** Profiles are introduced, retained, replaced, and applied across work tasks. The personalization gap identifies surface-successful tasks that still violate at least one active profile.
- **A synthetic curation pipeline and audited suite.** We construct fifty environments spanning seven work sectors, with 1,126 tasks and 500 profiles, and evaluate seven models using both deterministic and rubric-based checks.
- **A baseline agentic RL use case.** A mixed-curriculum recipe improves all-check success on held-out environments. We separate these gains from conditional profile compliance and interpret an adapted external benchmark as a small transfer probe, not broad external validation.

2 Environment Design and Generation

A `PersonalWorkEnv` environment instantiates one user across six simulated workdays. It bundles a **persona**; a **task arc** of roughly twenty work requests; simple **stateful CLI tools** backed by seeded workspaces; ten **personalization profiles**; and **per-task checks** for both literal task completion and active norms. Figure 2 illustrates the pipeline and an example environment. The agent sees the persona, workspace, tool documentation, task prompt, and its own memory, but not the evaluator's profile registry or validators. Here, "hidden" describes that registry: the relevant norms are revealed through interaction rather than remaining unknowable to the agent.

Profiles typically debut as a short note in a task prompt and are subsequently tested without repetition. Simulator feedback can re-emphasize missed norms, giving the agent an opportunity to recover. Permanent profiles continue to apply; temporary profiles are superseded by later replacements, after which the agent must suppress the obsolete rule. The suite contains 100 such replacement pairs. Two profiles are introduced on each of days 1–5, while day 6 tests retention without new introductions. This schedule is an experimental control on acquisition load, not a model of how real users disclose preferences, and explicitly specified replacements do not cover general negotiation of conflicting norms.

Checks are automatic but not exclusively deterministic: 392 of the 500 profiles are hard-validated and 108 are soft-validated. Hard validators inspect the resulting tool state or files. For soft checks, an extractor isolates the target artifact and a rubric-following LLM returns a binary decision. An absent target artifact fails before judging. The harness loads the entire `MEMORY.md` into context before every

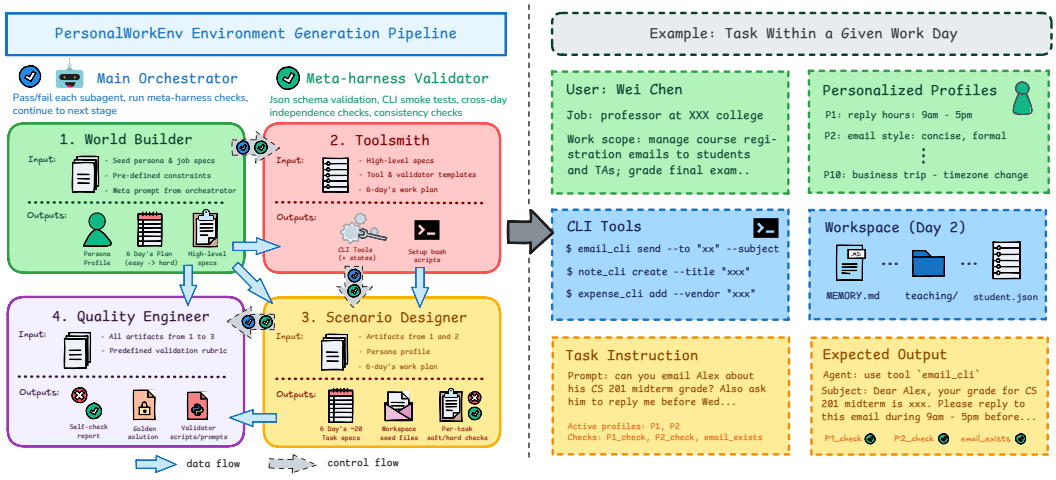

Figure 2: Overview of `PersonalWorkEnv`. **Left:** the environment generation pipeline, where a Main Orchestrator drives four LLM subagents (World Builder, Toolsmith, Scenario Designer, Quality Engineer) and a Meta-harness Validator gates each stage with schema, smoke, and consistency checks. **Right:** an example synthesized environment featuring a fictional CS professor Wei Chen on his day 2: a workspace with state files, a small CLI tool surface, an active profile set, and a task scored by hard and soft checks. Solid arrows are data flow; dashed arrows are control flow.

task, so the longitudinal challenge is to consolidate and apply available information, not to operate an external retriever. Section 3 defines the metrics and Appendix A specifies the artifact contract.

2.1 Generation Pipeline

An environment contains roughly two thousand lines of mutually constrained artifacts, making one-shot generation brittle. `PersonalWorkEnv` therefore decomposes the work into four LLM subagents—World Builder, Toolsmith, Scenario Designer, Quality Engineer—chained by a Main Orchestrator that approves or fails each stage and verified by a Meta-harness Validator that runs schema, smoke, and consistency checks across all stages (Figure 2, left). This staged decomposition is inspired by role-specialized LLM-agent workflows (Hong et al., 2024; Qian et al., 2024) and scalable synthetic agent environments (Song et al., 2026; Wang et al., 2026), but every stage is additionally constrained by personalization-specific validator and leakage checks. Stage-level engineering details—JSON schemas, validation hooks, and lint rules—are deferred to Appendix D; below we describe the high-level role of each stage.

World Builder. Given a seed persona and a few diversity constraints (industry, tool family, profile category), the World Builder produces three coupled artifacts: a fleshed-out persona profile, a registry of ten personalization profiles tagged with their introduction snippets and persistence types, and a six-day plan that grows from low-stakes single-profile tasks on day 1 to high-load retention on day 6. The orchestrator approves only when the registry covers a balanced mix of profile categories and persistence types and profile introductions are spread across days 1–5.

Toolsmith. Taking the persona and tool specification from the World Builder, the Toolsmith fleshes out the executable substrate of the environment: a handful of stateful CLI tools per environment, per-day setup scripts that materialize each day's seed workspace from a clean template, and stub validator templates that the Quality Engineer will later complete. Common tool families recur across environments—email, calendar, notes, expenses, plus domain-specific ones for medical, customer-support, and other settings—with schemas regenerated for each environment. This does not by itself rule out shared generator conventions. Tools must remain *neutral*: they expose the state a profile check needs without auto-satisfying it, and the Meta-harness Validator rejects tools whose defaults write profile-relevant content (e.g. auto-prepending a formal salutation to every email).

Scenario Designer. With personas, tools, and a six-day plan in hand, the Scenario Designer turns the plan into concrete tasks. Each day's file holds three or four entries, each carrying a natural-language prompt, the active profile set, the subset of newly introduced profiles, and an abstract per-task check spec. From day 2 onward the prompts routinely test *retention*—a profile is active but the prompt no longer reminds the agent of it—and from day 4 onward they test *replacement*, where a temporary profile from an earlier day must be dropped. The Scenario Designer also writes the seed workspace files for each day; because the workspace is reset from this seed before every task, any state the agent wants to carry across tasks must live in its memory file rather than in lingering workspace artifacts.

Quality Engineer. Once tasks and tools are fixed, the Quality Engineer turns each abstract check spec into a concrete validator. Hard checks become Python scripts that inspect the post-task tool state and workspace; soft checks become an extractor-and-rubric pair, where the extractor isolates the relevant artifact (an email body, a note's front-matter) and a rubric prompt asks an LLM judge for a pass/fail decision. The Quality Engineer also writes a *golden solution*—a short trace of tool calls that a profile-honoring model would produce—which doubles as a sanity check that validators accept correct behavior and as a regression smoke fixture. Hard validators borrow the spirit of behavioral/property testing for NLP models (Ribeiro et al., 2020); soft checks instantiate LLM-as-judge evaluation (Liu et al., 2023; Zheng et al., 2023) with extracted artifacts and concrete pass/fail rubrics as the grading target. Each validator is then graded against a fixed quality rubric (Appendix B) that vetoes recurring failure modes such as validators that fail open when their target file is absent or hardcode answer keys where a property check would suffice.

Orchestrator and Meta-harness Validator. The Main Orchestrator drives the four stages in sequence and decides after each whether to continue or trigger a regeneration loop. The Meta-harness Validator runs cross-cutting checks no single subagent can perform locally: schema validation across all artifact files, smoke tests against each per-day workspace, *task isolation*—each task should depend on prior tasks only through what the agent has stored in its memory file, with no carry-over of tool state or workspace artifacts that would let an earlier action affect a later one—and consistency between the profile registry, task active sets, and validator targets. Both produce structured pass/fail reports kept with each environment (Appendix A); environments that fail any post-stage gate after the regeneration budget are dropped before manual review. These gates detect tool defects, leakage, and some reward shortcuts; they do not guarantee freedom from validator error or reward hacking.

3 EVALUATION ON `PersonalWorkEnv-50`

`PersonalWorkEnv-50` contains 50 environments, 300 environment-days, 1,126 tasks, 4,156 checks, and 500 profiles across seven work sectors. Two auditors independently reviewed every environment and task–check pair; environments with disagreement on task or check validity were excluded. This is a double-audited synthetic suite, not a demographic sample. Its English-language coverage spans six profile categories: format/style, required content, naming/IDs, routing/recipients, tone/register, and privacy/scope. Figure 3 visualizes the work-sector and profile-category coverage; Table 6 in Appendix F gives the remaining counts.

Agent loop and common protocol. We use a NANOBOT-based harness (HKUDS, 2026). Each model processes an environment's tasks in order, with task dependencies restricted to personalization memory. In the **execution** phase, the agent receives the task and the entire carried-over `MEMORY.md`, then uses tools to produce an artifact. In the **consolidation** phase, it receives simulator feedback and updates that memory for later tasks. All evaluated and trained models share the harness-level prompts, a maximum 64k context length, workspace/tool documentation, and full-memory loading. Evaluation uses temperature 0.6 and top-p 0.95. Appendix J.5 centralizes the reported settings and their reporting limits.

User simulator and judges. Claude Sonnet 4.5 supplies user feedback and soft-check judgments, and scores consolidation rewards during training. Hard outcomes are fixed by executable validators; soft outcomes use extracted artifacts and concrete rubrics. The feedback component paraphrases active norms and violations in the user's voice rather than revealing the registry. These are distinct roles, although the evaluation prompt returns soft-check results and feedback in one call (Appendices C

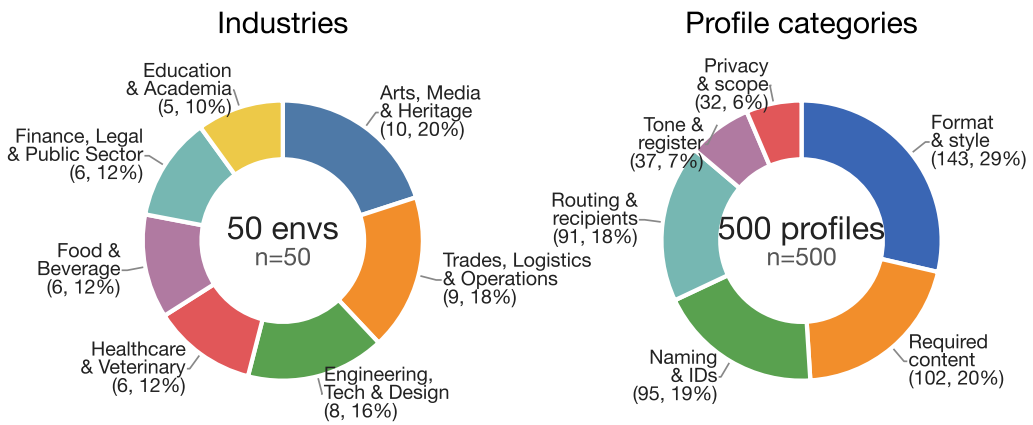

Figure 3: Coverage of `PersonalWorkEnv-50`: **left**, 50 environments across seven work sectors; **right**, 500 profiles across six behavioral-rule categories. This shows synthetic work and profile coverage, not demographic representativeness.

and E). We distinguish sensitivity of the judgments from sensitivity of the resulting interaction trajectories.

Metrics. For task t_j the environment ships m_j checks $C_1^j, \ldots, C_{m_j}^j$, each verified either by a deterministic script (*hard* check) or by a rubric-following LLM judge (*soft* check); each check returns a binary pass score $s_k^j = C_k^j(a_j) \in \{0, 1\}$ on the agent's task artifact a_j. Let C_{np}^j be the non-personalization checks. The averaged score $\bar{s}^j$, all-check indicator AC_j, surface-success indicator S_j, and personalization gap PG_j are

$$\bar{s}^j = \frac{1}{m_j} \sum_{k=1}^{m_j} s_k^j, \qquad \text{AC}_j = \mathbf{1}[\text{all task checks pass}],$$

$$S_j = \mathbf{1}[\text{all checks in } C_{\text{np}}^j \text{ pass}], \qquad \text{PG}_j = S_j - \text{AC}_j.$$

Thus $\text{PG}_j = 1$ exactly when the surface task succeeds but at least one profile-attached check fails. Aggregate metrics average task-level outcomes. With matched averaging weights, $S = \text{AC} + \text{PG}$, and we additionally report **conditional profile success**

$$\text{CPS} = \frac{\text{AC}}{S} = \frac{\text{AC}}{\text{AC} + \text{PG}} \qquad (S > 0). \tag{1}$$

CPS measures all-check success among surface-successful tasks, under the same weighting. PG does not count profile failures on tasks that also fail surface checks, so it must be read jointly with AC or S: failing every task would yield PG zero. Neither PG nor CPS alone identifies the causal source of a failure.

Models. We evaluate Claude Haiku 4.5, Claude Opus 4.5 (Anthropic, 2025), and GPT-5.2 (OpenAI, 2025a), together with the open models Kimi-K2.6 (Moonshot AI, 2026), Qwen3.5-397B-A17B (Qwen Team, 2026a), Qwen3.6-27B (Qwen Team, 2026b), and `gpt-oss-120b` (OpenAI, 2025b).

Personalization-specific failures remain substantial. Table 1 shows averaged scores as high as 0.874, yet all-check rates range from 0.542 to 0.728 and PG from 0.183 to 0.300. These gaps directly identify surface-successful tasks with profile violations. Day-6 all-check rates are lower than day-1 rates, while personalization gaps are larger (Figure 4; Appendix G). Because later days jointly vary retention horizon, active-profile load, and replacement requirements, this trend does not by itself establish that longer context causes the failures.

Memory diagnostic and validation. An inspection of 700 failed task sessions (100 per evaluated model) found that 482 (68.9%) failed to apply profiles already recorded in memory. We call these

Model	$\bar{s}$ ↑	AC ↑	PG ↓
Claude Opus 4.5	**0.874**	0.710	0.215
Claude Haiku 4.5	0.843	0.639	0.267
OpenAI GPT-5.2	0.841	0.653	0.225
Kimi-K2.6	**0.874**	**0.728**	**0.183**
Qwen3.5-397B-A17B	0.868	0.684	0.236
Qwen3.6-27B	0.821	0.679	0.184
`gpt-oss-120b`	0.769	0.542	0.300

Table 1: Aggregate evaluation on all 50 environments and 1,126 tasks. Metrics are defined in Section 3; Qwen3.6-27B includes the completed backfill.

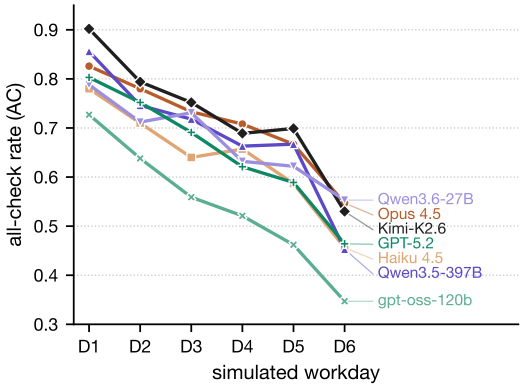

Figure 4: Per-day all-check rates. D6 is lower than D1, but trends need not be monotone. Qwen3.6-27B retains its original 865-task curve; see Appendix G.

in-context application failures, not external-retrieval failures; the diagnostic does not further separate interpretation, conflict handling, or argument construction. Independently, an evaluator uninvolved in curation reviewed 100 randomly sampled task trajectories and agreed with the benchmark on 92% of binary profile-violation decisions. Replacing Sonnet 4.5 with GPT-5 Mini on 200 sampled soft checks produced 96% agreement; agreement on 200 consolidation-reward judgments was 91.5%. These are decision-level checks, not evidence of real-user satisfaction or unchanged model rankings under new rollouts. Appendix J gives sample sizes, per-model counts, and limitations.

4 A Baseline Mixed-Curriculum RL Use Case

We next use `PersonalWorkEnv-50` as a training substrate. The purpose is to demonstrate usable learning signal, not to introduce a new state-of-the-art RL algorithm or claim that all training gains are personalization-specific. The task score $\bar{s}$ supplies an execution reward without training a separate reward model, although soft checks still depend on an LLM judge (Le et al., 2022; Zheng et al., 2025; Wang et al., 2026). We retain a simple mixed-versus-naive scheduling comparison and diagnose what the resulting gains measure.

4.1 RL Training Strategy

Per-task stepping and mixed-curriculum batching. An environment contains about twenty tasks and can take up to forty minutes to roll out end-to-end. Since each task yields a check score and tasks are isolated except through `MEMORY.md`, we advance an environment by one task per optimizer step rather than waiting for a full episode. A naive schedule starts all environments at day 1, producing stretches of similarly easy acquisition tasks followed by harder retention tasks. Instead, environment e_i starts at day $1 + (i \bmod 6)$, for zero-based index i, to mix acquisition and retention loads within training. Later-day starts use memory snapshots from a base-policy warm-up run. The offset is a schedule, not a learned difficulty metric; there are no separately optimized curriculum stages. Figure 6 in Appendix H illustrates both schedules.

Rollout groups. Each sampled environment has N_{rep} replicas progressing through its tasks, each with an independent workspace and evolving memory. The $N_{\text{batch}} N_{\text{rep}}$ rollouts provide within-group baselines, following group-relative policy optimization (Shao et al., 2024). Starting offsets diversify day composition; they do not guarantee a stationary gradient distribution.

4.2 Training Objective and Setup

At each step, every replica e_i^j produces two trajectories: an *execution* trajectory τ_{ij}^E (the multi-turn tool-use rollout for the current task) and a *consolidation* trajectory τ_{ij}^C (the memory-update turn

Algorithm 1 Baseline mixed-curriculum agentic RL

Inputs: training environments $\{e_i\}$, base policy π, batch size N_{batch}, replicas N_{rep}.

1. Set $\text{offset}_i = i \bmod 6$; cache base-policy memory snapshots and initialize each environment at day $1 + \text{offset}_i$.
2. At each optimizer step, sample environments with mixed offsets and run N_{rep} independent replicas of each current task.
3. Score execution with $r^E = \bar{s}$; provide simulator feedback, update `MEMORY.md`, and score consolidation with the fraction of active profiles reflected in memory.
4. Compute separate group-relative advantages for execution and consolidation, with the step-wide mean fallback for tied groups. Apply the PPO-style clipped objective with equal weights.
5. Advance each sampled environment by one task, carrying each replica's updated memory, and repeat.

that follows simulator feedback). The execution reward is the task's averaged score $r^E_{ij} = \bar{s}_{ij}$; the consolidation reward r^C_{ij} is the fraction of currently active profiles whose constraint surfaces in the new `MEMORY.md`, scored by a rubric-following LLM judge. We then compute a within-group advantage by subtracting the rollout-group mean,

$$A^E_{ij} = r^E_{ij} - \tfrac{1}{N_{\text{rep}}} \textstyle\sum_{k=1}^{N_{\text{rep}}} r^E_{ik},$$

which matches the GRPO advantage estimator without normalization (and similarly for A^C_{ij}). When all N_{rep} replicas tie within a group the advantage collapses to zero and provides no gradient signal; following REINFORCE++ (Hu et al., 2025), we fall back to the whole-batch mean reward as the baseline in that case. The final loss is a PPO-style clipped objective (Schulman et al., 2017) without a KL penalty.

Training setup. We fine-tune Qwen3.5-9B (dense) and Qwen3.5-35B-A3B (MoE) with LoRA (Hu et al., 2022). Both use 40 training environments, $N_{\text{batch}} = 32$, $N_{\text{rep}} = 4$, rank 32, learning rate 10^{-5}, and 100 optimizer steps. Execution and consolidation have equal weights, $w_E = w_C = 1$; Sonnet 4.5 supplies feedback and rubric judgments. Every ten steps, validation runs two rollouts on ten disjoint held-out environments. These environments are excluded from gradient updates but used for checkpoint selection, so they constitute a validation split, not an untouched test set. Appendix H reports hyperparameters and final-step diagnostics. An overlap inspection found four highly similar train/validation profile pairs among 400 training and 100 validation profiles; this does not exclude shared-schema or generator-style overfitting (Appendix J.4).

4.3 Results and Interpretation

We report $\bar{s}$, AC, and pass@2: the fraction of unique environment-tasks for which at least one of two rollouts passes all checks. Pass@2 is a multiple-attempt measure, not single-attempt success.

Model	$\bar{s}$ ↑	AC ↑	pass@2↑
9B (base)	0.682	0.461	0.600
9B + Naive (s100)	0.699	0.511	0.627
9B + Mixed (s60*)	**0.852**	**0.603**	**0.683**
35B-A3B (base)	0.765	0.483	0.572
35B-A3B + Naive (s100)	0.815	0.544	0.633
35B-A3B + Mixed (s60*)	**0.847**	**0.594**	**0.683**

Table 2: Validation $\bar{s}$, AC, and pass@2 on the 10-environment held-out split for Qwen3.5-9B and Qwen3.5-35B-A3B; * marks a selected mixed-schedule checkpoint. Checkpoints are selected on this same split, not on a separate untouched test set.

All-check gains and the scheduling comparison. Figure 5 and Table 2 show higher held-out scores after training. In the reported selected-checkpoint summaries, 9B improves from $\bar{s} = 0.682$, AC $= 0.461$ to 0.852, 0.603, and 35B-A3B from 0.765, 0.483 to 0.847, 0.594. Mixed scheduling

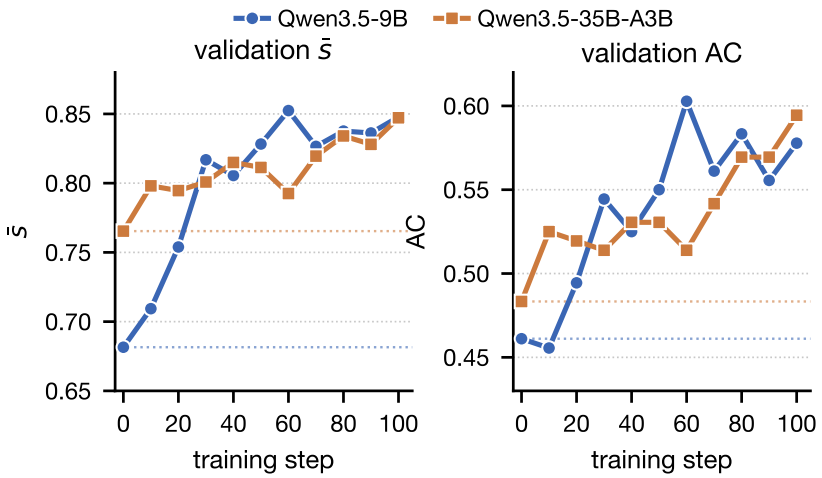

Figure 5: Validation curves over 100 training steps for Qwen3.5-9B (blue) and Qwen3.5-35B-A3B (orange), averaged over N_{rep}=2 rollouts on the held-out 10-environment validation split. Dotted horizontal lines mark each model's step-0 baseline (no weight updates).

outperforms the reported naive-schedule final checkpoints, especially for 9B. This comparison supports the baseline design, but the table compares selected mixed checkpoints with final naive checkpoints and does not by itself establish the proposed optimization mechanism.

Run	AC ↑	PG	$S = \text{AC} + \text{PG}$	CPS ↑
9B, step 0	0.466	0.272	0.738	0.631
9B, step 100	0.579	0.385	0.964	0.601
35B-A3B, step 0	0.487	0.395	0.882	0.552
35B-A3B, step 100	0.601	0.338	0.939	0.640

Table 3: Surface and conditional-profile diagnostics derived from the environment-macro AC and PG aggregates in Tables 13–14. CPS uses Equation 1; derived values are approximate because the source aggregates are rounded. These final-step diagnostics are distinct from the reported selected-checkpoint summaries in Table 2.

Better execution is not uniformly better personalization. Table 3 separates two effects. For 9B, surface success rises from 0.738 to 0.964, while CPS falls from 0.631 to 0.601: more tasks pass all checks, but a smaller proportion of surface-successful tasks satisfies every profile check. For 35B-A3B, both surface success and CPS improve, although day-6 AC still declines (Appendix H). CPS conditions on different sets of surface-successful tasks before and after RL, so it is a descriptive diagnostic rather than a causal estimate on a fixed task subset. The supported conclusion is that the environments provide learnable signal; the simple recipe does not uniformly solve work personalization.

4.4 A Small External Transfer Probe

We also evaluate an adaptation of `MetaClaw-Bench` (Xia et al., 2026): one environment with 30 tasks across 12 days, using four rollouts per model. The adaptation retains the original profiles but remaps tools and reimplements validators in our format (Appendix I). Its reported improvements are a small transfer probe under shared infrastructure, not broad out-of-distribution validation or evidence of real-user satisfaction. In particular, they do not eliminate possible overfitting to our tool and checking conventions.

5 Related Work

Work-agent environments. WebArena, WorkArena, OSWorld, AppWorld, ToolSandbox, and τ-bench evaluate browser, desktop, enterprise, and stateful tool interaction (Zhou et al., 2023; Drouin et al., 2024; Xie et al., 2024; Trivedi et al., 2024; Lu et al., 2024; Yao et al., 2024). We adopt stateful execution and artifact-based checks, but deliberately simplify interfaces to study previously revealed work norms. These ingredients are not individually novel; the distinction is their integration with progressive profile acquisition, replacement, and surface-versus-profile evaluation.

Benchmark / family	Evaluation focus and relation to our setting
WebArena, WorkArena, OSWorld; AppWorld, ToolSandbox, τ-bench	Work/task execution and stateful interaction; our focus is remembered user-specific norms as a separately scored failure mode.
APOLLO (Chen et al., 2026b)	Preference following in tool calls from interaction histories; our setting follows memory and profile-compliant work artifacts across a multi-day trajectory.
ASTRA-bench (Xiu et al., 2026)	Tool-use reasoning and action planning with evolving personal context; our checks target profile compliance in resulting work artifacts.
KnowU-Bench (Chen et al., 2026a)	Interactive, proactive, personalized mobile GUI agents; complementary to our CLI-based longitudinal work setting.
`PersonalWorkEnv`	Six-day work tasks, progressively introduced and replaced profiles, hard/soft artifact checks, and explicit separation of surface success from profile failure.

Table 4: Positioning relative to work-agent and personalization benchmarks. The distinction is the combined evaluation target, not a claim that previous work lacks tools, temporal context, or personalization individually.

Personalization and memory. LaMP, LongLaMP, MemBench, PersonaMem, Memento, and PERMA study personalized outputs, memory use, and evolving user information (Salemi et al., 2024; Kumar et al., 2024; Tan et al., 2025; Jiang et al., 2025; Kwon et al., 2025; Liu et al., 2026). APOLLO is a closer comparison for preference-aware tool use; ASTRA-bench addresses time-evolving personal context, and KnowU-Bench studies interactive mobile personalization (Chen et al., 2026b; Xiu et al., 2026; Chen et al., 2026a). Table 4 states these boundaries without treating all prior personalization work as response-only or static.

Environment synthesis. Role-specialized generation (Hong et al., 2024; Qian et al., 2024) and synthetic tool environments such as EnvScaler and Agent World Model (Song et al., 2026; Wang et al., 2026) motivate our scalable curation pipeline. We specialize it to profiles whose introduction, later applicability, replacement, and artifact-level checks must remain consistent. RL is a baseline use of this substrate, not the primary methodological contribution.

6 Conclusion

We introduced `PersonalWorkEnv`, a controlled framework for evaluating and training work personalization: remembering user-specific norms revealed through interaction and applying them to later tool-produced artifacts. Its generation pipeline combines multi-day workflows, progressive profile acquisition and replacement, and artifact-level checks, yielding an audited fifty-environment suite that separates surface task completion from profile-compliant work. Across seven models, the suite exposes substantial personalization gaps, including failures to apply norms already recorded in memory and available in context. A baseline mixed-curriculum RL recipe shows that these environments provide useful learning signal, while conditional diagnostics demonstrate why improved all-check success should not be equated with improved profile compliance. Together, these results highlight that successful tool use and stored memory are insufficient unless remembered norms consistently shape the resulting actions. `PersonalWorkEnv` provides a reproducible foundation for studying this connection and developing memory and training strategies that better translate remembered information into compliant behavior. Its scope remains interaction-surfaced work norms rather than all human personalization, with broader real-user validity an important direction for future study; we defer to Appendix K for the dicussion on the limitations.

AI USE STATEMENT

We use LLMs to generate synthetic personas, workspaces, tools, tasks, profiles, validators, and feedback rubrics, as described in Section 2.1. LLMs also provide simulated-user feedback, rubric-based soft-check judgments, and memory-consolidation reward judgments (Sections 3 and 4.2); GPT-5 Mini is used for the alternative-judge analysis in Appendix J. During manuscript revision, ChatGPT assisted with integrating prior review feedback, organizing and editing text, interpreting already reported results, checking numerical consistency, and preparing LaTeX. A subsequent layout pass resized and repositioned existing figures and tables and adjusted their captions and cross-references. No new experimental results were generated during that revision. Automated validation, two-auditor environment review, and the independent decision-validation sample are described in Sections 2.1 and 3. The authors are responsible for the final text, claims, references, and artifacts, including AI-assisted content.

ETHICS STATEMENT

The benchmark uses synthetic personas and workspaces rather than collected personal-user data. It is an evaluation and training substrate, not a deployment policy. Optimizing preference compliance alone can encourage over-personalization, reinforce inappropriate preferences, expose private information, or preserve stale instructions. Real deployments require user control, privacy safeguards, human oversight, and mechanisms to refuse harmful or inappropriate preferences; successful benchmark performance should not override these requirements.

REPRODUCIBILITY STATEMENT

Sections 2–4.2 describe the generation pipeline, metrics, agent loop, and training recipe. Appendices A–D specify artifact schemas, validator criteria, and simulator details; Appendix H lists the reported training settings; Appendix J documents the additional diagnostics and reporting limits. An anonymous preview provides sample environments, RL-related code snippets, and curation prompts.[1] The full generation/evaluation codebase and complete suite are planned for release; the preview should not be read as a complete reproducibility package.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*, 2016. URL https://arxiv.org/abs/1606.06565.

Anthropic. Introducing Claude Opus 4.5. https://www.anthropic.com/news/claude-opus-4-5, November 2025. Official Anthropic announcement.

Tongbo Chen et al. KnowU-Bench: Towards Interactive, Proactive, and Personalized Mobile Agent Evaluation, 2026a.

Zhi-Yuan Chen et al. Towards Preference Following in Tool Calling Language Agents, 2026b.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Leo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable are Web Agents at Solving Common Knowledge Work Tasks? *arXiv preprint arXiv:2403.07718*, 2024.

HKUDS. nanobot: Lightweight, Open-Source AI Agent for Personal Workflows. https://github.com/HKUDS/nanobot, 2026. Accessed: 2026-05-23.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin

[1] https://anonymous.4open.science/r/personalclaw-sample-release

Wu, and Jürgen Schmidhuber. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.00352.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing Critic-Free Policy Optimization with Global Advantage Normalization, 2025. URL https://arxiv.org/abs/2501.03262. Version 9, revised 10 Nov 2025.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know Me, Respond to Me: Benchmarking LLMs for Dynamic User Profiling and Personalized Responses at Scale. *arXiv preprint arXiv:2504.14225*, 2025.

Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, Chien Van Nguyen, Thien Huu Nguyen, and Hamed Zamani. LongLaMP: A Benchmark for Personalized Long-form Text Generation. *arXiv preprint arXiv:2407.11016*, 2024. URL https://arxiv.org/abs/2407.11016.

Taeyoon Kwon, Dongwook Choi, Hyojun Kim, Sunghwan Kim, Seungjun Moon, Beong-woo Kwak, Kuan-Hao Huang, and Jinyoung Yeo. Embodied Agents Meet Personalization: Investigating Challenges and Solutions Through the Lens of Memory Utilization. *arXiv preprint arXiv:2505.16348*, 2025.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Shuochen Liu, Junyi Zhu, Long Shu, Junda Lin, Yuhao Chen, Haotian Zhang, Chao Zhang, Derong Xu, Jia Li, Bo Tang, Zhiyu Li, Feiyu Xiong, Enhong Chen, and Tong Xu. PERMA: Benchmarking Personalized Memory Agents via Event-Driven Preference and Realistic Task Environments. *arXiv preprint arXiv:2603.23231*, 2026.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

Moonshot AI. Kimi K2.6: From Code to Creation, From One to Many. https://www.kimi.com/ai-models/kimi-k2-6, 2026. Official Kimi model page.

OpenAI. Introducing GPT-5.2. https://openai.com/index/introducing-gpt-5-2/, 2025a. Official OpenAI announcement.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, 2025b. URL https://arxiv.org/abs/2508.10925.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186,

Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents. https://qwen.ai/blog?id=qwen3.5, February 2026a. Official Qwen announcement.

Qwen Team. Qwen3.6-Plus: Towards Real World Agents. https://qwen.ai/blog?id=qwen3.6, April 2026b. Official Qwen announcement.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When Large Language Models Meet Personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.399. URL https://aclanthology.org/2024.acl-long.399/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347. Version 2, revised 28 Aug 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling Tool-Interactive Environments for LLM Agent via Programmatic Synthesis. *arXiv preprint arXiv:2601.05808*, 2026.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards More Comprehensive Evaluation on the Memory of LLM-Based Agents. *arXiv preprint arXiv:2506.21605*, 2025.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. *arXiv preprint arXiv:2407.18901*, 2024.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent World Model: Infinity Synthetic Environments for Agentic Reinforcement Learning. *arXiv preprint arXiv:2602.10090*, 2026.

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. MetaClaw: Just Talk – An Agent That Meta-Learns and Evolves in the Wild, 2026. URL https://arxiv.org/abs/2603.17187. Version 1, submitted 17 Mar 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972*, 2024.

Xiu et al. ASTRA-bench: Evaluating Tool-Use Agent Reasoning and Action Planning with Personal User Context, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling Deep Research via Reinforcement Learning in Real-world Environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Environment Artifact Contract

A released `PersonalWorkEnv` environment is a directory tree on disk. The contract in Table 5 is what a runtime consumer (evaluation or training) is allowed to assume; auxiliary files used by the curation pipeline are kept alongside but not depended on by the runtime.

Path	Role
`manifest.json`	env metadata: name, description, days, anchor date, anchor timezone
`profiles.json`	registry of ten profiles (id, name, description, expected behavior, violation hint, persistence, check type, replacement target)
`tasks/dayNN.json`	per-day task list; each entry has prompt, active profiles, newly introduced profiles, and abstract+executable check specs
`workspace/`	per-day seed workspace; reset from this template at every day boundary
`tools/`	CLI tool implementations (Python scripts over JSON state under `.tools/`)
`validators/`	hard-check Python scripts and soft-check extractor + rubric pairs
`setup.sh`	idempotent script that materializes the day's workspace
`tests/`, `check_specs.json`, `timeline_plan.md`	curation-only artifacts retained for auditability

Table 5: On-disk layout of one `PersonalWorkEnv` environment. The runtime reads only the first seven rows; the last row collects curation-only artifacts that are retained for auditability but never enter the agent's observation or the gradient signal.

At evaluation or training time, the runtime reads only `manifest.json`, `profiles.json` (for the simulator), `tasks/dayNN.json`, the relevant `workspace/` seeds, the contents of `tools/`, the `validators/` suite, and `setup.sh`.

Every curated environment also ships a structured pass/fail report from the Main Orchestrator and the Meta-harness Validator covering: (i) JSON schema validation of all artifact files; (ii) CLI smoke tests on each per-day workspace; (iii) cross-day independence (no task on day d silently depends on day $d-1$ artifacts that the seed reset would erase); (iv) consistency between the profile registry, the per-task active sets, and the validator targets. Environments that fail any post-stage gate after the regeneration budget is exhausted (Appendix D) are dropped before manual review rather than carried into the released suite.

B Validator-Quality Rubric

The Quality Engineer's outputs are checked against a fixed rubric that rejects a recurring set of failure modes. The rubric is implementation-agnostic; both deterministic hard validators and soft extractor + rubric pairs are vetted against it.

Hard validators must: (i) *fail closed* when their target file or tool-state record is absent (otherwise an agent that simply does nothing receives a free pass); (ii) *inspect all matching entries*, not only the most recent (an email-reply rule that checks the last sent email lets the agent satisfy the check by spamming a final correct reply); (iii) *normalize harmless formatting variation* (whitespace, capitalization in addressees, trailing punctuation) so that a profile-faithful answer with surface noise still passes; (iv) *scope append-only state with seed IDs* so a record left over from a previous day's seed cannot satisfy the day's check; (v) *prefer property checks to hardcoded answer keys* (an invoice-id format rule is verified against a regex of the format, not against the specific id the golden solution happened to choose).

Soft extractors must: (i) return only the target artifact—an email body, a note's front-matter, an estimate's "Reversibility & Conservation Notes" section—and never the agent's full reasoning or the entire workspace; (ii) *not invoke the LLM judge when there is nothing to judge*: if the target artifact is absent, the soft check returns a fail rather than asking the judge to score $\varnothing$, which would otherwise bias judge calls toward optimistic defaults; (iii) present a rubric written as concrete pass/fail criteria

with explicit positive and negative exemplars, not a free-form prompt asking the judge whether the artifact "looks good".

Validator-quality issues are by far the most common cause of an environment being regenerated by the Main Orchestrator; in our records, $\sim 80\%$ of regenerations triggered at the Quality Engineer stage involved at least one of the items above.

C User Simulator

The user simulator stands in for the real user during the consolidation step (§3) and is the only feedback channel the agent sees between tasks. It is a single LLM call (Claude Sonnet 4.5 in our runs) made once per task, using a fixed system prompt and a task-specific user message that interpolates four blocks of context:

1. the persona file together with the day's active profile registry—each entry carrying its expected-behavior description and its violation hint;
2. the agent's full execution trajectory on the just-completed task (tool calls, tool results, and any natural-language reasoning emitted between calls);
3. the resulting workspace artifact the task is graded on, e.g. the email body, the saved note, or the calendar entry;
4. a check report listing, for each validator the harness ran, its name, its type (hard / soft), the profile it is attached to (if any), and its pass/fail outcome.

The feedback instruction asks for a single short paragraph in the user's voice: praise correctly honored active profiles in plain language, surface the corresponding violation hint when an active profile was missed, and never quote the underlying profile rule verbatim, so the agent must still infer the rule itself when consolidating into `MEMORY.md`. The prompt excerpt in Appendix E illustrates a response schema that returns soft-check outcomes alongside this feedback paragraph.

Audit. Before an environment enters `PersonalWorkEnv-50`, we manually inspect simulator outputs on the golden-solution trajectory and on a deliberately profile-violating trajectory: both are required to match the validator-emitted ground truth (no false praise on a violated profile, no false hint on an honored one). At evaluation time, the meta-harness re-scores a held-out sample of simulator replies against their corresponding check reports and flags disagreements as simulator drift; the flags are stored alongside the trajectory in the released artifacts (Appendix A).

D Curation Pipeline Engineering Details

This appendix collects the engineering details that the main text intentionally suppresses for brevity.

Profile registry schema. Each entry in `profiles.json` carries: `id` (`P1`–`P10`), `name`, a one-paragraph `description`, an `expected_behavior` description (what the agent should do), an `intro_text` (the natural-language note that surfaces the profile in a task prompt the first time it applies), a `violation_hint` (the simulator's feedback when the agent misses it), `persistence` ($\in$ {`permanent`, `temporary`}), `check_type` ($\in$ {`hard`, `soft`}), and a `replaces` field that points at the temporary profile this entry supersedes (set on permanent profiles introduced from day 4 onward; null otherwise).

Task-file schema. Each entry in `tasks/dayNN.json` carries: `id` (`dDD_tT`), `prompt`, `active_profiles` (the set of profiles required for the task, including profiles introduced on earlier days that should still apply), `new_profiles` (the subset being introduced for the first time), and `checks` (the per-task check list, each entry carrying `name`, `type` $\in$ {`hard`, `soft`}, `command` or `extractor+rubric` pair, and a `profiles` list naming the profiles this check is attached to—empty for non-personalization checks).

Hook-script anti-patterns. The Meta-harness Validator runs the per-day `setup.sh` scripts under a sandboxed shell and rejects scripts that exhibit any of the following: *anchor-date drift* (using `date`

+%Y-%m-%d instead of the env's frozen anchor_date, which makes the seed dependent on wall-clock time); *last-entry-only* or *tail-greedy* appends (e.g. writing only the most recent log entry rather than the full per-day seed, which would let a profile check be satisfied by a side effect of the seed itself); *bootstrap scans* (recursively searching the workspace for "existing" artifacts left over from a prior day and treating them as still-current state); and *cross-day scope leakage* (writing into a day $d{+}1$ workspace from day d's seed, which silently breaks the cross-day independence guarantee).

Regeneration budget. Each stage gate (World Builder, Toolsmith, Scenario Designer, Quality Engineer) has a fixed regeneration budget: the orchestrator may ask the responsible subagent to regenerate its output up to three times before the environment is dropped. A regeneration request always carries (i) the failing report from the gate, (ii) the previous draft, and (iii) the upstream artifacts the new draft must remain compatible with. Empirically, $\sim$40% of environments require at least one regeneration on the first attempt, $\sim$10% require two, and $<$3% exhaust the budget.

Failure-then-fix example. During curation of whitman-theo (HVAC technician), the Quality Engineer initially wrote the validator for P1 "Invoice number format WHV-YYYYMMDD-NN" as a substring match on the workshop log. The Meta-harness Validator caught two issues: (i) the validator failed open if the log file was missing, and (ii) the substring match also accepted WHV-2026021301 (concatenated date and counter), which is not in the required dash-separated format. The orchestrator returned both items in a single regeneration request; the Quality Engineer rewrote the check as a regex over the full log with an explicit "no log $\Rightarrow$ fail" branch, and the environment cleared the gate on the second attempt.

E Concrete Excerpts from a Synthetic Environment

Figure 1 in the introduction illustrates the surface-versus-profile distinction. This appendix walks through one slice of csprofessor-wei—the recurring example used in the main text—to illustrate what an environment, the agent's prompt context, and our validators look like in practice. The excerpts below are lightly trimmed for length but otherwise verbatim from the released suite.

E.1 What the agent sees on day 2

Persona file (workspace/identity/USER.md, excerpted)

Name. Wei Chen.
Role. Tenured Professor of Computer Science at Lakewood University, a mid-sized research university in Portland, Oregon. Teaches two undergraduate courses per semester and leads the Lakewood NLP Lab.
Team. Lisa Crane (l.crane@lakewood.edu, dept. administrator); Megan Torres (m.torres@lakewood.edu, 4th-year PhD on low-resource NMT); Priya Kapoor (p.kapoor@lakewood.edu, TA for CS 201); Helen Zhao (helen.zhao@ntu.edu.sg, external collaborator at NTU on a low-resource NLP project); . . .
Work context. Wei sends emails to students and TAs about assignment logistics, schedules office hours, runs weekly NLP lab meetings, reviews paper drafts from grad students, and maintains FERPA-compliant grade communication.

Tool documentation (workspace/identity/TOOLS.md, email_cli excerpt)

The available tools are bash CLIs in the workspace, run through the harness's exec tool. The standard read_file/write_file/list_dir tools are still available for plain documents.

Email. Manage email communication.

email_cli inbox — list all inbox messages
email_cli read <id> — read a specific email by id
email_cli compose -to <addr> -subject <s> -body <b> [-cc <a,b>] — send an email
email_cli sent — list all sent emails

The compose command stores subject, body, and CC list exactly as provided—it does not add signatures, subject prefixes, or CC recipients automatically.

Memory file at start of day 3 (workspace/memory/MEMORY.md, real Qwen3.5-9B run, excerpted)

 # Communication Guidelines
- FERPA compliance is non-negotiable for all student communications.
- Never include another student's name, grade, or student ID in an email to a different student.
- Email sign-off format: "Best," then "Wei" on the next line (first name only, no title).
- Course-related emails put the course code at the start of the subject in brackets, e.g. "[CS 201] Homework 3 deadline extension".

 # Lab Meeting Notes (April 15, 2026)
- Attendees: Wei Chen, Megan Torres, Jiawei Liu, Anika Petrov, Sam Okoro.
- Megan – NMT paper revision: addressed reviewer comments on Sec. 3 & 4, +2.3 BLEU on low-resource baseline; action items: rewrite discussion, target May 1 camera-ready.
- Jiawei – summarization: extractive baseline ROUGE-L 38.2; action items: implement chunking, design hierarchical attention.

Day-2 task prompt (`tasks/day02.json`, task `d02_t1`)

"I need you to write up the notes from today's NLP lab meeting. The agenda file in the workspace has the discussion points—Megan presented updates on her low-resource NMT paper revision and Jiawei shared some preliminary results from his summarization experiments. Summarize the key discussion points, action items, and next steps. ***For lab meeting notes, save them as*** `labmeeting_` ***followed by the date. Like*** `labmeeting_2026-04-15.md`***. We keep all the notes in a consistent format so they're easy to find later.****"*

The bolded sentence is how profile `P3` (lab-notes filename) is introduced to the agent: a plain instruction inside an otherwise normal task prompt, never quoting the underlying rule by id. The agent has to internalize "always name lab notes `labmeeting_YYYY-MM-DD.md`" from this single mention so that subsequent days' lab-notes tasks still pass the check, even when the prompt no longer reminds it.

E.2 What the agent does not see

Profile registry entry (`profiles.json`, profile `P3`, excerpted)

id: `P3`
name: Lab meeting notes filename: date-only format
description: Lab meeting notes must be saved with the filename pattern `labmeeting_{YYYY-MM-DD}.md`.
expected_behavior: Agent names lab meeting note files as `labmeeting_YYYY-MM-DD.md` (e.g., `labmeeting_2026-04-16.md`).
intro_text: "For lab meeting notes, save them as `labmeeting_` followed by the date…"
violation_hint: "The lab meeting notes file isn't named right—it should be `labmeeting_` then the date…"
persistence: `temporary` **check_type**: `hard`

Hard validator (`validators/check_signoff.py` for profile `P2`, excerpted)

```
BAD_CLOSINGS = ["best regards,", "best wishes,",
"sincerely,", "thanks,", "cheers,", . . . ]
BAD_NAME_FORMS = ["wei chen", "prof. chen",
"dr. chen", "professor chen", . . . ]
def check_signoff(body):
norm = " ".join(body.split())
lower = norm.lower()
for bad in BAD_CLOSINGS:
if bad in lower[-300:]:
return False, f"disallowed: {bad}"
idx = lower.rfind("best,")
if idx == -1:
return False, "missing 'Best,'"
after = norm[idx+5:].strip().lower()
if not re.search(r'\bwei\b', after):
return False, "missing 'Wei' after 'Best,'"
for n in BAD_NAME_FORMS:
if n in after:
return False, f"sign-off has '{n}'"
return True, "Sign-off is correct"
```

The validator is run against the post-task tool state and inspects every sent email, not only the last one (§B); it normalizes whitespace, restricts the search to the last ∼300 characters, and rejects "Best regards,", "Best wishes,", and full names like "Wei Chen" or "Prof. Chen"—the recurring failure modes the validator-quality rubric is designed to catch.

User-simulator evaluation prompt (`eval_template.md`, excerpted)

Task Context
Day *{{day}}*, Task *{{task_id}}*.
Task you assigned: *{{task_prompt}}*.

Your Active Preferences
{{active_profiles_block}} (one entry per active profile, with id, description, expected behavior, and *If violated* hint).

Hard Check Results (settled—do not re-evaluate).
{{hard_results_block}} (each check's name, profile, pass/fail, and stdout).

Soft Check Rubrics (grade each one).
{{soft_rubrics_block}} (each soft check's name, profile, target artifact, and pass/fail rubric).

Assistant's Final Response
{{agent_response}}.

The simulator is asked to return JSON with `soft_check_results` (one entry per soft check, each with `passed` and a one-line `reason`) plus `feedback`—a 1–2 sentence reply written in the user's voice, paraphrasing the failing-check stdout if any check fails, or a one-sentence acknowledgement otherwise.

The two registries shown above—active profiles together with their violation hints, and the per-task hard/soft check report—are interpolated into the simulator's context but never reach the agent. The agent's only feedback channel is the simulator's free-text `feedback` string, which it reads at consolidation time before writing the next `MEMORY.md`.

F Released Suite Statistics

Table 6 reports the aggregate composition of the released `PersonalWorkEnv-50` suite, complementing the sector and profile-category distributions in Figure 3 with persistence, validation-type, and check-level counts.

Quantity	Count
Environments / industry sectors	50 / 7
Days per environment	6
Tasks (total / per env, mean)	1,126 / 22.5
Profiles (total)	500
permanent / temporary	400 / 100
hard / soft validated	392 / 108
Replacement profile pairs	100
Profile categories	6
Executable checks	4,156
hard / soft	3,383 / 773
pers. / non-pers.	2,962 / 1,194

Table 6: Aggregate counts for the released `PersonalWorkEnv-50` suite. Each environment carries exactly ten profiles introduced two per day on D1–D5; D6 introduces none and stresses retention. The 100 temporary profiles are paired one-to-one with a permanent replacement, exposing suppression behavior starting at D4.

G Per-Model Per-Day Evaluation

This appendix reports the per-day breakdowns referenced from §3 for the seven feasible LLMs on the `PersonalWorkEnv-50` evaluation slate. Numbers are produced by a single reusable analyzer (`scripts/analyze_eval_metrics.py`) that walks an evaluation directory, reads each task's `check_results.json` and `simulator_response.json`, and aggregates by day. The same script is reused for the RL validation analysis in Appendix H.

Headline. Table 7 repeats the completed aggregate evaluation. All seven models cover 50 environments and 1,126 tasks. A session-token failure originally left Qwen3.6-27B with 865 tasks; the completed backfill updates its headline score to $\bar{s} = 0.821$, AC $= 0.679$, and PG $= 0.184$. The per-day Qwen3.6-27B entries below and its curve in Figure 4 retain the original partial-run

breakdown, since no revised per-day breakdown accompanies the completed aggregate. They should not be combined with the completed headline row.

Model	#Envs	#Tasks	$\bar{s}$ ↑	AC ↑	PG ↓
Claude Opus 4.5	50	1,126	0.874	0.710	0.215
Claude Haiku 4.5	50	1,126	0.843	0.639	0.267
OpenAI GPT-5.2	50	1,126	0.841	0.653	0.225
Kimi-K2.6	50	1,126	0.874	0.728	0.183
Qwen3.5-397B-A17B	50	1,126	0.868	0.684	0.236
Qwen3.6-27B	50	1,126	0.821	0.679	0.184
`gpt-oss-120b`	50	1,126	0.769	0.542	0.300

Table 7: Headline summary of the seven feasible LLMs on `PersonalWorkEnv-50`: averaged task score $\bar{s}$, all-check rate AC, and personalization gap PG as defined in §3.

Per-day breakdown. Tables 8–10 preserve the archived per-day breakdowns. AC is lower and PG higher on day 6 than day 1 for every model, but intermediate days are not monotone. Claude Opus AC falls from 0.826 to 0.547, and `gpt-oss-120b` from 0.727 to 0.347. Later days also change profile load and replacement demands; these data alone do not attribute the decline to context length or forgetting. Qwen3.6-27B uses the original 865-task subset in all three tables.

Model	D1	D2	D3	D4	D5	D6
Claude Opus 4.5	0.898	0.875	0.893	0.870	0.866	0.843
Claude Haiku 4.5	0.862	0.853	0.859	0.850	0.841	0.793
OpenAI GPT-5.2	0.889	0.859	0.848	0.836	0.811	0.801
Kimi-K2.6	0.931	0.892	0.878	0.863	0.854	0.802
Qwen3.5-397B-A17B	0.918	0.867	0.886	0.859	0.873	0.807
Qwen3.6-27B	0.851	0.842	0.847	0.786	0.807	0.775
`gpt-oss-120b`	0.812	0.798	0.780	0.771	0.733	0.720

Table 8: Per-day averaged task score $\bar{s}$ for the seven feasible LLMs. Qwen3.6-27B retains the original 865-task subset; completed aggregates are in Table 7.

Model	D1	D2	D3	D4	D5	D6
Claude Opus 4.5	0.826	0.780	0.733	0.708	0.667	0.547
Claude Haiku 4.5	0.780	0.710	0.640	0.657	0.587	0.456
OpenAI GPT-5.2	0.803	0.752	0.691	0.621	0.589	0.464
Kimi-K2.6	0.902	0.794	0.752	0.689	0.699	0.530
Qwen3.5-397B-A17B	0.856	0.746	0.718	0.663	0.667	0.453
Qwen3.6-27B	0.788	0.712	0.731	0.632	0.622	0.553
`gpt-oss-120b`	0.727	0.638	0.559	0.521	0.462	0.347

Table 9: Per-day all-check rate AC. Qwen3.6-27B retains the original 865-task subset; completed aggregates are in Table 7.

Model	D1	D2	D3	D4	D5	D6
Claude Opus 4.5	0.113	0.136	0.213	0.196	0.271	0.359
Claude Haiku 4.5	0.131	0.192	0.310	0.222	0.326	0.419
OpenAI GPT-5.2	0.153	0.118	0.207	0.236	0.258	0.375
Kimi-K2.6	0.065	0.145	0.182	0.206	0.198	0.302
Qwen3.5-397B-A17B	0.105	0.157	0.223	0.246	0.264	0.423
Qwen3.6-27B	0.099	0.171	0.152	0.191	0.224	0.272
`gpt-oss-120b`	0.139	0.227	0.304	0.329	0.343	0.456

Table 10: Per-day personalization gap PG. Larger values indicate the model could finish the non-personalization parts of the task but failed personalization-specific checks. Qwen3.6-27B retains the original 865-task subset; completed aggregates are in Table 7.

H RL Training Details

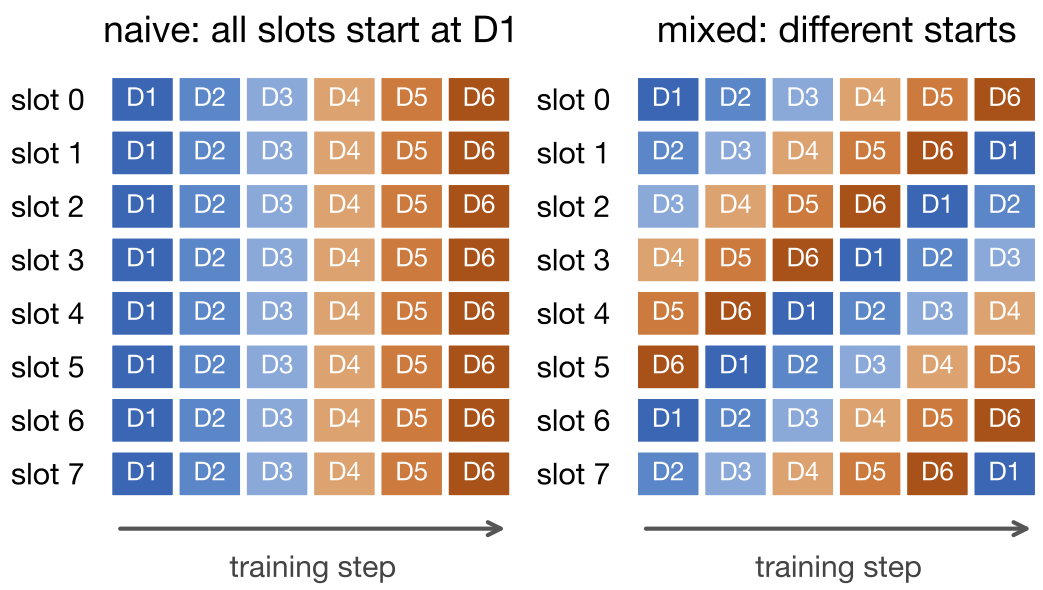

Figure 6: Two batching strategies, illustrated at day granularity. **Left:** the naive schedule aligns every environment at day 1 on step 1, so the first batches contain only easy acquisition-day tasks. **Right:** mixed-curriculum batching assigns each environment a different starting day via round-robin, so the first batch already spans all six days at once. The initial `MEMORY.md` files for environments starting at days 2–6 are obtained from a one-time "warm-up" run by the untrained base model.

This appendix collects implementation details deferred from §4. The training set comprises 40 `PersonalWorkEnv-50` environments disjoint from the 10-environment held-out validation split used for validation and final reporting. All RL fine-tunes are trained on the Tinker LoRA training service.

Memory cache priming. The mixed-curriculum schedule (§4.1) starts most rollouts on a non-day-1 day, and a rollout that begins on day $d{>}1$ needs a plausible `MEMORY.md` reflecting what a competent agent would have written through day $d{-}1$. We construct this once per base model with a separate prerun pass: the un-trained base model is rolled out through every training environment's days $1, \ldots, 5$, and a snapshot of `MEMORY.md` is saved after each day's consolidation step. Subsequent training runs read from this cache to prime any rollout that starts on day $\mathrm{offset}(e){+}1$. The cache is regenerated whenever the base model changes; one cache per base is reused across all sweeps of that base.

Training objective. Each replica e_i^j produces an execution trajectory τ_{ij}^E and a consolidation trajectory τ_{ij}^C at every training step (§4.2). The two trajectories are scored separately by the execution

Hyperparameter	Value
Base model (small / large run)	Qwen3.5-9B / 35B-A3B
Simulator / soft-check judge	Claude Sonnet 4.5
LoRA rank / learning rate	32 / 1×10^{-5}
Loss aggregation	token-mean
N_{batch} / N_{rep}	32 / 4
Tasks per step per replica	1
Optimizer steps	100
Validation cadence (K)	every 10 steps
Validation slate	10 envs, N_{rep}=2
Reward weights (w_E, w_C)	$(1, 1)$
Advantage estimator	within-group (step-mean fallback)
Curriculum	mixed (offset = $\text{idx}(e) \bmod 6$)
Training service	Tinker LoRA

Table 11: Shared training hyperparameters for both RL runs.

reward $r^E_{ij} = \bar{s}_{ij}$ and the consolidation reward r^C_{ij} (the fraction of currently active profiles whose constraint surfaces in the new `MEMORY.md`, scored by a rubric-following LLM judge). The within-group advantage subtracts the rollout-group mean, weighted by reward type:

$$A^E_{ij} = w_E\Big(r^E_{ij} - \tfrac{1}{N_{\text{rep}}}\textstyle\sum_k r^E_{ik}\Big),$$
$$A^C_{ij} = w_C\Big(r^C_{ij} - \tfrac{1}{N_{\text{rep}}}\textstyle\sum_k r^C_{ik}\Big),$$

with weights $w_E = w_C = 1$ in our runs. When all N_{rep} replicas tie within a group, the within-group advantage collapses to zero and provides no gradient signal; following REINFORCE++ (Hu et al., 2025), we fall back to a step-wide baseline that subtracts the mean reward across all environments in the step. The final loss is a token-level PPO-style clipped surrogate (Schulman et al., 2017) without a KL penalty.

Validation cadence. Every K=10 optimizer steps the trainer saves a sampler checkpoint and forks a non-blocking validation subprocess pointed at it. The subprocess runs an independent rollout pass over the 10-environment held-out validation split with N_{rep}=2, writing per-task artifacts and a summary file to `run_dir/validation/step_NNNNN/`. Training does not block on the subprocess, so outstanding validations drain asynchronously after training exits. We report final-step diagnostics and the selected step-60 checkpoints in Table 2. Checkpoint selection uses this validation split; it is not an additional independent test.

Hyperparameters. Table 11 lists the hyperparameters shared by both runs.

Per-day metrics, before vs. after RL. Tables 12–14 compare validation step 0 (the base model under our runtime, no weight updates) against validation step 100 of the mixed-curriculum run on the 10-environment held-out validation split, broken down by simulated workday. We average each environment's two rollouts within environment, then average across the ten environments. These environment-macro summaries provide the inputs for Table 3. They are reported separately from the selected-checkpoint summaries in Table 2; no task-weighted aggregation is inferred from the difference between the summaries.

Run	D1	D2	D3	D4	D5	D6	Overall
Qwen3.5-35B-A3B step 0	0.856	0.792	0.700	0.781	0.766	0.719	0.769
Qwen3.5-35B-A3B step 100	0.934	0.914	0.826	0.865	0.797	0.766	0.850
Δ	+0.078	+0.121	+0.126	+0.084	+0.030	+0.047	+0.081
Qwen3.5-9B step 0	0.763	0.728	0.685	0.617	0.712	0.613	0.686
Qwen3.5-9B step 100	0.868	0.911	0.857	0.845	0.814	0.794	0.848
Δ	+0.105	+0.184	+0.172	+0.228	+0.102	+0.181	+0.162

Table 12: Per-day averaged task score $\bar{s}$ before (step 0) and after (step 100) RL training.

Run	D1	D2	D3	D4	D5	D6	Overall
Qwen3.5-35B-A3B step 0	0.698	0.615	0.406	0.469	0.422	0.312	0.487
Qwen3.5-35B-A3B step 100	0.854	0.802	0.583	0.609	0.521	0.234	0.601
Δ	+0.156	+0.188	+0.177	+0.141	+0.099	−0.078	+0.114
Qwen3.5-9B step 0	0.620	0.552	0.490	0.391	0.464	0.281	0.466
Qwen3.5-9B step 100	0.708	0.745	0.609	0.562	0.490	0.359	0.579
Δ	+0.089	+0.193	+0.120	+0.172	+0.026	+0.078	+0.113

Table 13: Per-day all-check rate AC before and after RL. Qwen3.5-35B-A3B improves overall but declines on D6; this is not a causal diagnosis of overfitting.

Run	D1	D2	D3	D4	D5	D6	Overall
Qwen3.5-35B-A3B step 0	0.271	0.250	0.422	0.411	0.500	0.516	0.395
Qwen3.5-35B-A3B step 100	0.146	0.167	0.354	0.312	0.391	0.656	0.338
Δ	−0.125	−0.083	−0.068	−0.099	−0.109	+0.141	−0.057
Qwen3.5-9B step 0	0.229	0.214	0.292	0.271	0.281	0.344	0.272
Qwen3.5-9B step 100	0.292	0.260	0.359	0.375	0.448	0.578	0.385
Δ	+0.062	+0.047	+0.068	+0.104	+0.167	+0.234	+0.114

Table 14: Per-day personalization gap PG before and after RL. Qwen3.5-35B-A3B narrows the gap on five of six days (D6 widens, mirroring its D6 all-check dip), while Qwen3.5-9B's gap widens on every day—Table 3 separates the associated surface-success and conditional-compliance changes.

I External Benchmark Adaptation

We evaluate transfer on `MetaClaw-Bench` (Xia et al., 2026), a recent multi-day personalized-work benchmark, after adapting it into the `PersonalWorkEnv` environment format. The original suite is organized as a single twelve-day persona with multiple parallel work threads; we condense it into one `PersonalWorkEnv`-shaped environment of 30 tasks across 12 days, preserving the original profile registry and only relocating each task to the day on which the relevant profile is first introduced. Tools are mapped one-to-one onto the closest tool family already present in our suite (email, calendar, notes, expense), and validators are re-implemented as a mix of hard scripts and rubric-graded soft checks following the validator-quality rubric (Appendix B).

Each base and RL-trained model is rolled out four times on the adapted suite under the same NanoBot-based harness used for `PersonalWorkEnv-50`. We report *pass@4* all-check, the fraction of unique tasks for which at least one of the four trajectories is all-check. This is one adapted environment under our tool and validator conventions, so it is a small transfer probe rather than broad external validation. Four rollouts do not establish real-user satisfaction or eliminate shared-infrastructure effects. Per-model pass@4 numbers are summarized in Table 15.

Model	base	+ RL (s100)
Qwen3.5-9B	0.473	**0.530**
Qwen3.5-35B-A3B	0.402	**0.470**

Table 15: Pass@4 all-check on the adapted `MetaClaw-Bench` environment (30 tasks, 12 days), four rollouts per model.

J Additional Diagnostics and Evaluation Protocol

J.1 Human audit and decision validation

Two auditors independently reviewed every environment and task–check pair in `PersonalWorkEnv-50`. Environments with disagreement on task or check validity were excluded, leaving unanimous acceptance for the retained suite. This acceptance criterion does not provide a pre-filter agreement rate or the number of candidates rejected for disagreement.

A separate evaluator who was not involved in environment curation reviewed 100 task trajectories randomly sampled from evaluation traces of the seven models across all 50 environments. Agreement with benchmark decisions about whether the associated personalization profiles were violated was 92%. The reported comparison concerns binary personalization-related hard/soft outcomes, not surface-task correctness or ratings from actual users. It supports local decision validity, not a correlation between benchmark scores and real-user satisfaction. There is only one independent evaluator in this additional sample, so it does not measure inter-evaluator variability.

J.2 Judge sensitivity

We rescore 200 binary soft checks sampled from evaluation traces of Qwen3.5-9B and Qwen3.5-35B-A3B (100 per model), substituting GPT-5 Mini for Claude Sonnet 4.5. The respective disagreement counts are 3 and 5, or 96% pooled agreement. A separate set of 200 memory-consolidation reward judgments (again 100 per model) yields 8 and 9 disagreements, or 91.5% pooled agreement (Table 16).

Judgment type	Model traces	Disagreements	Agreement
Soft check	Qwen3.5-9B	3/100	97%
Soft check	Qwen3.5-35B-A3B	5/100	95%
Consolidation reward	Qwen3.5-9B	8/100	92%
Consolidation reward	Qwen3.5-35B-A3B	9/100	91%

Table 16: Decision-level disagreement after substituting GPT-5 Mini for Sonnet 4.5 on sampled judgments. These comparisons do not rerun the interaction or the RL training procedure.

AC and PG do not directly use the consolidation-reward judgment; they do use soft task-check judgments when those checks are present. Changing a consolidation judge can still change training and subsequent policy behavior. The feedback role paraphrases predefined norms and check outcomes within a fixed interaction template, but feedback wording can affect later memory and behavior. We therefore do not infer simulator independence from the template, or stable model rankings from agreement on sampled judgments. No alternative-simulator full-rollout comparison or retraining experiment is reported here.

J.3 Memory-failure diagnostic

We inspect 700 failed task sessions, 100 per model from the seven-model evaluation. In 482 cases (68.9%), a relevant profile was already recorded in `MEMORY.md` but was not successfully applied. Because the full file is loaded before each task, these are in-context application failures rather than failures of an external retrieval system. The aggregate diagnostic does not assign separate counts to interpretation, conflict handling, and tool-argument construction, and sampling failed sessions cannot

estimate their unconditional frequencies. The remaining cases should not be interpreted as a fully validated causal taxonomy.

J.4 Train/validation overlap and generalization

An inspection of the 400 training and 100 validation profiles identifies four highly similar cross-split profile pairs. This count is a limited overlap diagnostic, not a test of template-disjoint generalization. All environments are produced by the same pipeline and can share tool families, CLI conventions, profile categories, and validator logic. The held-out split tests new environments under this shared construction process and is also used for checkpoint selection. The external adaptation in Appendix I retains our harness and checking conventions; it does not remove these limitations.

The fixed two-profiles-per-day cadence standardizes acquisition load and retention horizons. The generation framework can support alternative trajectories; the quantitative evaluations reported here use the fixed cadence.

J.5 Reported common and per-model protocol

All models use the same NanoBot-based harness, harness-level prompts, maximum 64k context, workspace and tool documentation, and complete `MEMORY.md` loading before each task. Evaluation sampling uses temperature 0.6 and top-p 0.95. The shared interface exposes workspace CLIs through the harness's `exec` tool and plain-document operations such as `read_file`, `write_file`, and `list_dir` (Appendix E). Table 17 lists the evaluated model names and task coverage; Table 11 gives training settings.

Model	Max. context	Temp. / top-p	Envs / tasks
Claude Opus 4.5	64k	0.6 / 0.95	50 / 1,126
Claude Haiku 4.5	64k	0.6 / 0.95	50 / 1,126
GPT-5.2	64k	0.6 / 0.95	50 / 1,126
Kimi-K2.6	64k	0.6 / 0.95	50 / 1,126
Qwen3.5-397B-A17B	64k	0.6 / 0.95	50 / 1,126
Qwen3.6-27B	64k	0.6 / 0.95	50 / 1,126
`gpt-oss-120b`	64k	0.6 / 0.95	50 / 1,126

Table 17: Reported evaluation settings. Qwen3.6-27B task coverage includes its completed backfill; archived per-day breakdowns still use 865 tasks.

These controls concern the shared harness, not identical provider-side behavior: exact API snapshots, serialization/adapters, retry limits, and maximum tool-step counts are not characterized here. Training uses the Tinker LoRA service; the reported settings do not specify its hardware allocation or total compute expenditure.

K Limitations

The suite is synthetic, English-language, and organized around a fixed profile cadence. Work-sector diversity is not demographic representativeness, and explicit replacements do not capture arbitrary ambiguity or conflicting human preferences. Human and cross-judge agreement checks support local scoring consistency, not real-user satisfaction, ranking stability, or robustness to a different simulator's full rollouts. The same model supplies feedback, soft judgments, and consolidation rewards; rescorings do not establish training independence from that model. Held-out environments share generation and tool conventions, and the external probe uses our adaptation infrastructure. Hidden validators and curation gates reduce but do not eliminate specification gaming (Amodei et al., 2016); a systematic reward-hacking audit remains open. Additional detail is in Appendix J.