A20551352- Waseem pasha Mohammad

Analysis report

Experiment Results for Chicago Wide Cast Smart-Home Services

Overview

This report presents the analysis of the first five experiments conducted on the Chicago WideCast Smart-Home Services requirements and task estimation project. Each experiment utilized a different combination of generative AI model and framework to generate tagged requirements and corresponding role-based tasks and time estimates. The models assessed were from Ollama, Replicate, and OpenAI platforms, and the frameworks used were LangChain, LangGraph, and LlamaIndex.

Experiment Summary

Experiment 1

- **Platform:** Ollama
- **Model:** LLaMA3.2:3b
- **Framework:** LangChain/LangGraph

Generated Requirements:

- Mixed coverage of functional and non-functional requirements.
- Captured system validations and customer flows.
- Several redundant entries noted.

Task Assignment:

- Inconsistent formatting and some duplication.
- Documentation Engineer seemed overloaded, suggesting task leakage.
- Useful quantitative estimates, albeit with some exaggerated totals (e.g., 18,750 hours for login module).

Issues:

- Task overlap and repetition.

- Requirements list contained both clean and redundant entries.

Experiment 2

- **Platform:** Replicate
- **Model:** Meta LLaMA-3 70b-Instruct
- **Framework:** LangChain/LangGraph

Generated Requirements:

- Clean, well-structured functional requirements based on services.
- Mapped to WideCast's service offerings.
- Clear delineation by role and system functionality.

Task Assignment:

- High accuracy in matching tasks to roles.
- Explicit and consistent productivity-based estimates.
- Tasks spanned from planning to rework phases.

Strengths:

- Best clarity and coherence so far.
- Realistic and actionable estimates for each role.

Experiment 3

- **Platform:** OpenAI
- **Model:** GPT-4o-mini
- **Framework:** LangChain/LangGraph

Generated Requirements:

- Extensive requirement set, including business logic and constraints.
- Incorporated promotional and conditional flows.

Task Assignment:

- All roles appropriately covered.

- Task estimates were calculated accurately.
- Responsibilities were logically grouped and clearly formatted.

Highlights:

- Strong coherence across all outputs.
- High alignment between requirements and tasking.

Experiment 4

- **Platform:** Ollama
- **Model:** LLaMA3.2:3b
- **Framework:** LlamaIndex

Generated Requirements:

- Functional but limited in number.
- Focused on subscription and service restrictions.

Task Assignment:

- Massive task duplication across all roles.
- Misalignment between roles and responsibilities.
- Estimates seemed copy-pasted across agents.

Issues:

- No clear differentiation between what each role contributes.
- Inflated and repeated time estimates.

Experiment 5

Platform: Replicate**Model:** Meta LLaMA-3 70b-Instruct**Framework:** LlamaIndex

Generated Requirements:

- Concise and relevant service-based requirements.

- Covered dependency constraints for service access.
- Minimal overlap or noise.

Task Assignment:

- Clean division of labor by role.
- Reasonable, productivity-based time estimates.
- Roles matched accurately to domain areas.

Strengths:

- Most refined outcome among LlamaIndex experiments.
- Logical and well-balanced workload distribution.

Experiment 6

Platform: OpenAI**Model:** GPT-4o-mini**Framework:**LlamaIndex"

Generated Requirements:

- Clearly articulated preconditions for service access.
- Introduced contract duration selection logic.
- Reinforced rule-based dependencies among services.

Task Assignment:

- Full task lifecycle covered: writing, review, rework.
- Tasks spread logically across roles.
- Estimates were productivity-driven and coherent.

Strengths:

- Comprehensive planning and estimation.
- Strong consistency in role alignment and effort breakdown.

Experiment 7

Platform: Replicate**Model:** deepseek-ai/deepseek-r1**Framework:** LangChain/LangGraph

Generated Requirements:

- 31 well-tagged functional and constraint-based requirements.
- Derived from user roles, business rules, and contract logic.
- Each requirement paired with a specific use-case.

Task Assignment:

- Role-based breakdown reflecting real-world responsibilities.
- Effort estimates derived using structured productivity assumptions.
- Task names and outputs tailored to role semantics.

Highlights:

- Strongest alignment between business logic and system behavior.
- Excellent clarity, coverage, and use-case integration.

Experiment 8

Platform: Replicate **Model:** deep seek-ai/deepseek-r1**Framework:**LlamaIndex

Generated Requirements:

- Reiteration of core functional dependencies.
- Requirements echoed those in earlier experiments but fewer in number.
- Focused on service eligibility and contract durations.

Task Assignment:

- Redundant task allocation across roles (e.g., multiple roles writing same documents).
- Minimal task differentiation by role.
- Time estimates seemed auto applied without refinement.

Issues:

- Poor separation of concerns across roles.

Unnecessary overlap in task ownership and duplication in efforts.

Which model and platform produced the best results considering
correctness, completeness, and coherence? Explain your answer.

Let's compare the **top three**:

Experiment 3: OpenAI + GPT-4o-mini + LangChain/Langraph

- **Correctness**: Excellent — requirements were complete, accurate, and included constraints, dependencies, and promotions.
- **Completeness**: Included all business logic, roles, and user flows. No major gaps.
- **Coherence**: Roles were properly mapped, task effort was realistic, and the document was clear and structured.
- **Edge**: Best balance of accuracy and coherence. Strong agent behavior, low duplication, and high-quality output.
- **Downside**: None major.

Experiment 7: Replicate + DeepSeek R1 + LangChain/Langraph

- **Correctness**: High — requirements well-structured and matched domain semantics.
- **Completeness**: Covered user roles, business rules, and contract logic.
- **Coherence**: Very clear role mapping and task names. Slightly more verbose than E3.
- **Downside**: Slightly less fluent output formatting compared to GPT-4o.

Experiment 2: Replicate + LLaMA-3 70b + LangChain/Langraph

- **Correctness**: High, but less nuanced than GPT-4o.
- **Completeness**: Good, but didn't capture complex business logic like contract rules.
- **Coherence**: Excellent task-role matching, but a little flatter in detail.

Lower Performers

- **Experiments 4 & 8 (LlamaIndex)**: Weak role separation, repeated tasks across roles, copy-paste estimates.
- **Experiment 1**: Exaggerated estimates and inconsistent format.
- **Experiment 5**: Good attempt but less complete than 2, 3, and 7.

- **Experiment 6**: Very close to 3, but limited creativity in task mapping.

Final Decision

Best Overall Performer: Experiment 3
Model: GPT-4o-mini
Platform: OpenAI
Framework: LangChain/Langraph

Why?

- **Correctness**: Clean, logical, and aligned with business context.
- **Completeness**: Captured services, constraints, role-specific actions, and dependencies.
- **Coherence**: Roles and tasks were well-aligned, estimates were realistic, and the output was easy to follow.

Experiment	Platform	Model	Framework	Correctness	Completeness	Coherence
1	Ollama	LLaMA3.2:3b	LangChain	Medium	Medium	Low
2	Replicate	LLaMA-3 70b-Instruct	LangChain	High	High	High
3	OpenAI	GPT-4o-mini	LangChain	High	High	High
4	Ollama	LLaMA3.2:3b	LlamaIndex	Medium	Low	Low
5	Replicate	LLaMA-3 70b-Instruct	LlamaIndex	Medium	Medium	Medium
6	OpenAI	GPT-4o-mini	LlamaIndex	High	High	High
7	Replicate	deepseek-ai/deepseek-r1	LangChain	High	High	High
8	Replicate	deepseek-ai/deepseek-r1	LlamaIndex	Medium	Low	Low