Problem Statement and Goals
Tandem

Team 11, Tandem
Justin Kwinecki
Zayed Umer
William Pol
Manav Patel
Musab Mondal

Table 1 records the changes to this document.

Table 1: Revision History

Date	Developer(s)	Change
Sept 22, 2026	Justin Kwinecki	Initial problem statement and goals
Sept 26, 2026	Justin Kwinecki	Finished Section 1
Sept 27, 2026	Zayed Umer	Revise problem statement and goals; see disclosure appendix
Sept 27, 2026	Zayed Umer	Cut repetition and clarify wording
Sept 27, 2026	Zayed Umer	Revise the problem section from review
Oct 11, 2026	Justin Kwinecki	Pivot idea update
Oct 11, 2026	Justin Kwinecki	Align scope and goals with semantic routing
Oct 14, 2026	Justin Kwinecki	Clarify serving-engine integration focus

1 Problem Statement

Tandem aims to reduce response latency when semantic routing switches between models by reducing repeated prompt prefill for shared conversation context.

1.1 Problem

Semantic routing selects a model for each request based on the request's meaning or needs. In a multi-turn interaction, a router may send successive requests with the same conversation history to different models. Before a model can generate a response, it processes the prompt in a phase called prefill. A newly selected model may need to prefill the conversation history again, adding delay before it can respond.

Repeated prefill increases time to first token and uses computing capacity that could serve other requests. Existing serving software can reuse a prefix after it has been processed for the same model [**?**]; after a model switch, the selected model may need to process that shared prefix again. This makes model-switch latency and repeated prompt processing the project's measurable serving concerns. The problem is to reduce that repeated work while preserving the selected model's response and the independence of each request.

1.2 Scope

Tandem targets text-generation requests in a multi-turn interaction where semantic routing may select different models for successive turns. The requests share conversation history, while the newest user message may differ. The project investigates serving-engine integration to reduce repeated prefill on a model switch. Its planned approach is to transfer a cross-model key-value (KV) cache into the serving engine's KV memory and use it when compatible with the request. When reuse is unavailable or unusable, the engine uses its native prefill path. The evaluation will use one server and a defined set of models and serving configurations. It will include same-model turns and turns routed to a different model, with and without reusable context. If feasibility work requires a different reuse approach, the same workload and native-prefill comparison will apply.

Evaluation will compare identical workloads, routing decisions, models, cache hits, and generation settings with and without Tandem's changes. The baseline will retain the serving engine's existing prefix reuse features. The project excludes training the base language models, development of a semantic router, agent orchestration, document retrieval, application interfaces, and operation across multiple servers. Reuse of similar but differently worded context is outside the initial scope. Reuse across different models will be limited to model combinations for which correctness can be established.

The project focuses on serving-engine integration and evaluation, rather than developing a new method to predict one model's cache from another. Evaluation will compare time to first token after a model switch using the chosen reuse approach against the engine's native full-prefill path, while checking output correctness and recording reuse failures and fallback behavior. The Development Plan describes the implementation and evaluation approach.

1.3 Inputs and Outputs

The inputs are:

- Requests, each with an identifier, conversation history, latest user message, and generation settings such as maximum response length and sampling controls.
- The set of trained models available to the router and the encoding rules that convert text into tokens, the units each model processes.
- A routing decision for each request, specifying the selected model.
- For the planned transfer approach, a candidate KV cache and its source model, target model, associated conversation context, and compatibility metadata.
- For evaluation, request arrival times, the serving configuration, and a baseline run using the same routing decisions.

The outputs are:

- Generated text for each accepted request, with its identifier, selected model, and completion status or failure indication.
- Evaluation results: time to first token, end-to-end response time, completed requests per second, prompt tokens processed on each model, and comparison with the baseline for the same selected model. The comparison reports generated-text agreement and output-score differences under controlled settings. Results include the workload and configuration needed to reproduce them.

1.4 Stakeholders

- **Application developers, including independent developers:** need faster responses without changing their agents' tasks or results.
- **Serving-engine maintainers:** need changes they can integrate and maintain, supported by correctness and compatibility tests.
- **Infrastructure operators and deploying organizations:** need to serve more requests within their hardware capacity and budget.
- **Application end users:** need timely responses that are unaffected by other agents' private instructions.
- **The capstone team, TA, and instructor:** need a feasible scope and reproducible evidence that the project meets its goals.

1.5 Environment

Developers and operators will run Tandem on a local or remote Linux server with a supported graphics processing unit (GPU) and enough memory for the models and requests. They supply the models, routing decisions, and compatible serving software. Applications submit requests through a programmatic interface; end users interact with those applications.

Initial evaluation uses one server and a fixed set of model configurations at a time. The Development Plan covers developer tools and laboratory hardware.

2 Goals

These goals will be refined into acceptance criteria in the Software Requirements Specification and test procedures in the Verification and Validation Plan. Performance comparisons use the baseline in Section 1.2, repeated runs, and reported variation. Improvements must exceed that variation.

G1. **Reduce repeated prefill after model switches.** Reduce the processing of shared conversation history when a request is routed to a different model. Compare the chosen reuse approach with the serving engine's native full-prefill path, and report time to first token, repeated prompt tokens, and reuse or fallback outcomes.

G2. **Increase serving capacity without unacceptable delay.** Seek higher completed requests per second on the same hardware and matched workloads, while reporting median and 95th-percentile time to first token alongside throughput.

G3. **Preserve model behaviour and request independence.** Compare generated text and output scores with the selected model's native serving baseline under controlled settings, including multi-token generation. Assess whether one request's private instructions affect another request's output scores.

G4. **Maintain ordinary serving.** Preserve the serving engine's ability to handle same-model requests and requests without shared context, and measure any effect of the integration on their delay and throughput.

Evaluation will report improvements, regressions, and workloads with no measurable change.

3 Stretch Goals

After meeting the core goals, the team may pursue:

- **Support deeper branching.** Extend correctness and performance evaluation to conversations with several shared branching points, including branches introduced during an ongoing interaction.
- **Support more configurations.** Repeat the evaluation on additional models or GPUs to test whether the benefits persist.
- **Reduce memory demand.** Use less peak GPU memory than the matched baseline for shared-context workloads and measure whether this permits more concurrent requests within the same memory limit.

4 Custom Documents (CDs)

The Development Plan draft proposes these custom documents, subject to instructor approval:

- **Fall: Machine Learning Model Report.** Justify the selected model, evaluation prompts, and correctness checks. Discuss model and data limitations, and explain why training procedures do not apply.
- **Winter: Performance Report.** Analyze repeated processing, response delay, throughput, and resource use. Record configurations, workloads, repeated trials, variation, and failure cases. The Verification and Validation Report will reference this analysis when assessing whether requirements were met.

Appendix — Generative AI Use Disclosure

OpenAI Codex revised all sections using Justin Kwinecki's initial draft and proposed additions, the Development Plan draft, repository checklists, and course outline. It checked the cited prefix-caching documentation, edited for clarity and concision, and built and inspected the PDF. No performance experiments were run or novelty established.

Zayed Umer supplied reflection notes about collaborative brainstorming, the difficulty of finding a novel idea, and the motivation to improve memory use even modestly. Codex edited those notes and connected them to the proposed scope. The team remains responsible for reviewing the wording, claims, and scope before submission.

Justin Kwinecki used ChatGPT (free web version) to research which groups of professionals would benefit from this project and their specific use cases, for Section 1.4, Stakeholders. The team reviewed the resulting list before including it.

Appendix — Reflection

1. **What went well while writing this deliverable?**

   We agreed early that we wanted to work on inference. That shared interest made brainstorming collaborative: we could build on each other's ideas and focus on which problem to tackle.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**

   Finding a novel idea was harder than choosing the topic. With so much inference research underway, we struggled to tell whether an idea addressed an open problem. That uncertainty remains. We plan to review existing work and measure the selected serving engine while testing the feasibility of cross-model cache reuse. The evaluation can adjust the reuse approach if feasibility work shows it is needed.

3. **How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project?**

   We wanted to improve inference resource use, especially memory, even if the gain was modest. The proposed scope turns that motivation into one workload we can measure: successive requests with shared conversation history that semantic routing sends to different models. Using existing models lets us focus on serving-system integration, correctness, and performance. Memory savings and broader model support remain stretch goals. This gives us a bounded implementation to evaluate within the course, with success judged by reproducible results.