Project Requirements Document

Born Rule Breakers:
Steve Cao, Haadi Khan, Rhik Mazumder, Alice Wang

January 2026

1 Technical Roles

Our technical roles were assigned as follows:

- Rhik: Project Lead,
- Haadi: GPU Acceleration PIC,
- Steve: Quality Assurance PIC,
- Alice: Technical Marketing PIC.

2 Technical Approach

2.1 Verification Strategy

The LABS problem, which in the quantum context corresponds to minimizing an Ising Hamiltonian, is equivalent classically to maximizing the merit factor (MF):

$$MF = \frac{N^2}{2\sum_{k=1}^{N-1} C_k^2} = \frac{N^2}{2E}.$$

A list of known solutions to LABS up to $N \leq 82$, alongside their corresponding Ising Hamiltonian values and merit factors, can be found in [1, 3]. We can simply check our results against these sequences in unit tests to perform verification.

2.2 Hybrid Algorithm Design

We adopt the **Pauli Correlation Encoding (PCE)** as our quantum primitive to replace the counterdiabatic approach. This choice is motivated by a combination of empirical performance metrics and practical scalability considerations [5].

First, PCE demonstrates **significantly better runtime scaling** compared to QAOA and classical heuristics for combinatorial optimization. For example, for problems with even N:

Method	Runtime Scaling (Even N)	Runtime Scaling (Odd N)
QAOA [2]	$O(1.46^N)$	$O(1.46^N)$
PCE [5]	$O(1.330^N)$	$O(1.325^N)$
Memetic Tabu Search [7]	$O(1.358^N)$	$O(1.409^N)$

This is a roughly 9–10% improvement in the exponential base and compounds for large N, translating to an approximately 4× speedup at $N = 100$. PCE also shows a clear advantage for near-optimal or approximate solutions, which are often sufficient in real-world settings [7].

Furthermore, PCE compresses problem instances into a **polynomial number of qubits** (i.e. $N = \text{poly}(n)$, where n is the number of qubits). For example, it has been shown that $N = 45$ requires only 4 qubits, $N = 66$ requires 8, $N = 120$ requires 10, and so on with just 20 qubits required for $N > 200$. This represents an 11–30× reduction in qubit requirements relative to standard encodings, enabling tractable experiments on current quantum hardware [5].

Finally, **PCE circuits are shallow**, with depth scaling as $O(\sqrt{N})$, significantly lower than $O(N)$ in standard approaches. The shallowness of these circuits, together with the reduced qubit count, help **suppress barren plateaus** in training by keeping gradients appreciable. For example, an instance with $N = 45$ requires only 30 two-qubit gates, 15 layers, and ~ 150 parameters, allowing effective optimization of the circuit on near-term devices [5].

The combination of extreme qubit efficiency and shallow circuits makes PCE immediately deployable on NISQ devices. For instance, on IonQ Forte, instances with $N = 120$ were solved using only 10 qubits, ~ 3000 shots, and a bitstring error rate of 1.6% [5]. This resource-efficient structure ensures both scalability to larger problem sizes and compatibility with current quantum hardware, enabling exploration of combinatorial optimization problems beyond the reach of standard variational approaches.

2.3 GPU Acceleration

Stage 1: GPU-Accelerated PCE using CUDA-Q

We employ `CUDA-Q`'s `nvidia` or `nvidia-mgpu` backend to perform state vector simulation entirely on the GPU. The PCE quantum circuit—a hardware-efficient brickwork ansatz with 10 layers of single-qubit rotations (R_x, R_y, R_z) and CNOT gates—executes natively on GPU, avoiding costly CPU-GPU memory transfers.

For gradient estimation, we parallelize the parameter-shift rule: each of the ~ 150 circuit parameters requires two forward passes (parameter $\pm\pi/2$ shift), and `CUDA-Q` can batch these evaluations across GPU cores or distribute them across multiple GPUs when using the multi-GPU target. Expectation values for all N Pauli operators in our $\Pi^{(NC)}$ set are computed simultaneously using `CUDA-Q`'s batched observable evaluation, which internally parallelizes across the Pauli set.

We run multiple independent PCE optimizations (10–20 runs with different random initializations) in parallel across available GPU resources to explore the loss landscape, selecting the best solution as our warm start. This GPU-native approach accelerates PCE training by approximately 50–100× compared to CPU-based simulation for problems in the $N = 20$–50 range.

Stage 2: GPU-Accelerated Memetic Tabu Search using CuPy

Once we obtain the PCE seed, we transition to classical refinement using `CuPy` (NVIDIA's GPU-accelerated `NumPy`) for all population-based operations. The population of 50 binary sequences is stored in GPU memory as a $(50, N)$ `CuPy` array throughout the entire MTS execution, eliminating CPU-GPU transfer overhead.

The critical computational bottleneck—energy evaluation—is fully vectorized: we compute LABS autocorrelation energies for all population members simultaneously using `CuPy`'s parallel reduction operations. For each individual's tabu search step, we generate all N single-bit-flip neighbors on GPU (creating an (N, N) array via `cp.tile` and diagonal masking), then evaluate all N energies in a single batched kernel call. This leverages GPU parallelism to explore the entire 1-flip neighborhood simultaneously rather than sequentially.

Genetic operations (crossover, mutation) are implemented as vectorized `CuPy` operations on GPU arrays. For maximum performance on critical loops, we optionally deploy custom CUDA kernels written with `CuPy`'s `RawKernel` interface for autocorrelation calculation, which can provide an additional 2–5× speedup over pure `CuPy` by optimizing memory access patterns and using shared memory. The entire MTS population never leaves GPU memory until final convergence, enabling our implementation to process 1000+ MTS generations in minutes rather than hours.

3 Execution Tactics

3.1 Agentic Workflow

We used numerous AI agents all via Claude Opus 4.5. Our first agent (Alan) was tasked with creating custom a Model Context Protocol (MCP) server. First, Alan web crawled the `CUDA-Q` documentation. Then he semantically chunked and embedded the text using OpenAI's `embedding-3-small-model`, putting these into our ChromaDB vector database. We then built MCP tools to query the database effectively and accurately.

Following that, our agentic workflow consisted of a lead agent (Kole) orchestrating a team of subagents. Kole was given access to any files involved in our research plan, including the provided Jupyter Notebooks as well as any papers we wanted to reference (both for verification and for developing a quantum algorithm).

Our first subagent (Isabella) was tasked with creating the initial implementation based on [6] as context. Our second subagent (Tanish) worked in tandem,

receiving that paper in addition to our verification sources [1, 3] in order to serve the purpose of **LLM as a judge**. Our third subagent (Ethan) was tasked with performing hardware optimization, including the GPU acceleration tasks discussed in subsection 2.3.

3.2 Success Metrics

To assess the success of our original hybrid algorithm, we plan to measure and plot the maximum merit factor of our algorithm after exactly 200000 loss function calls for select values of N. We believe this is a good quantification of our program's solution quality relative to its time efficiency. Furthermore, this metric also avoids the risk of having a single program execute for many hours in our data-taking process, as in measuring time-to-solution (TTS) through number of loss function calls directly. This was a special source for concern because PCE-based variational algorithms have no performance guarantees [4].

3.3 Resource Management and GPU Selection

We adopt a tiered GPU strategy to optimize our $20 Brev credit allocation:

- **Development and testing:** We will use L4 instances ($1.00/hour) for small PCE instances ($N \leq 30$) to validate correctness, tune hyperparameters, and perform initial benchmarks. These local benchmarks provide a preliminary estimate of the crossover point between the quantum PCE algorithm and the classical MTS algorithm.
- **Production runs:** We will switch to A100 instances ($2.00/hour) for target instances ($N = 35$–45) where the 40GB memory and higher compute capability justify the cost. Thus PCE experiments are concentrated around the predicted crossover point to maximize efficiency and minimize wasted credits.
- **Session management:** We will implement aggressive orchestration with automated 30-minute idle checks and immediate instance termination after runs complete. This timer-based monitoring prevents "zombie instances," and a shared team log tracks all active GPU sessions, updated every 30 minutes.

Budget allocation reserves $5 for L4 development (5 hours), $12 for A100 production benchmarking (6 hours), leaving a $3 buffer for debugging. By testing code locally and concentrating full-scale benchmarking toward the end, we avoid credit depletion, reduce idle time, and ensure that machines are used efficiently. To prevent credit depletion, we enforce the following protocols:

- **Pre-GPU testing**: We will validate all code locally on CPU simulations ($N \leq 10$) before any GPU instance is launched.

- **Zombie prevention**: The GPU Acceleration PIC maintains a shared team log tracking all active instances. We will set 30-minute phone alarms and confirm instance status in the log. Instances are terminated immediately after job completion using automated shutdown scripts.
- **Session coordination**: Only one team member will launch GPU instances at a time.
- **Iterative benchmarking**: We will test hyperparameters on L4 instances before committing to full-scale A100 runs, concentrating expensive experiments near the end of the hackathon when our strategy is validated.

References

[1] J. Bernasconi et al. Low-autocorrelation binary sequences: exact results and bounds, 2015.

[2] M. Harrigan et al. Evidence for scaling advantage of the quantum approximate optimization algorithm, 2023.

[3] A. N. Leukhin and E. N. Potekhin. A bernasconi model for constructing ground-state spin systems and optimal binary sequences. *Journal of Physics: Conference Series*, 613(1):012006, 2015.

[4] Camilo G. Maciel T. O. Canabarro A. Borges L. Aolita L. Sciorilli, M. A competitive nisq and qubit-efficient solver for the labs problem, 2026.

[5] M. Sciorilli, G. Camilo, T. O. Maciel, A. Canabarro, L. Borges, and L. Aolita. Quantum correlation encoding for combinatorial optimization, 2026.

[6] M. Sciorilli et al. Perturbative correlation encoding for optimization problems, 2024. Version 2.

[7] J. Smith et al. Scaling advantage with quantum-enhanced memetic tabu search for low autocorrelation binary sequences, 2025. NVIDIA.