ID	User Story	MVP / Reach	Function	Task	Owner
1.1	As a researcher, I want to use and improve Google's neural-network quantum decoder solution, so that I can reduce errors in a quantum computer better than other state-of-the-art decoder solutions.	MVP	Set Up	Set up github repository under which out decoder solution lives	Christopher
1.2		MVP		Set up virtual environment for decoder solution	Christopher
1.3		MVP		Create Docker image and container for decoder	Christopher
1.4		MVP		Run test Docker containing virtualizations	Christopher
1.5		MVP	Run on Simulated Dataset	Create surface code code in Stim to simulate surface code error dataset	Christopher
1.6		MVP		Generate / simulate Stim datasets	Mara
1.7		MVP		Implement analog readout wrapper	Mara
1.8		MVP		Connect to the ARC	Arjun
1.9		MVP		Construct embedding scheme (with vectorizing training data and tokenization)	Arjun
1.10		MVP		Implement proposed transformer error correction architecture with pipeline	Christopher
1.11		MVP		Train on Stim data on ARC	Arjun
1.12		MVP		Test and evaluate performance on Stim data	Mara
1.13		MVP		Iteratively adjust parameters and model architecture to improve performance on Stim data	Tzu Chen
1.14		MVP	Run on Sycamore Dataset from Paper	Adjust embedding scheme / pipeline for sycamore dataset	Christopher
1.15		MVP		Train on Sycamore data on ARC	Arjun
1.16		MVP		Test and evaluate performance on Sycamore data	Tzu Chen
1.17		MVP		Iteratively adjust parameters and model architecture to improve performance on Sycamore data	Tzu Chen
2.1	As a researcher, I want to apply Google's neural-network quantum decoder to other quantum codes, so that I have a generalized decoder applicable to any type of quantum code / system.	Reach	Run on Simulated Dataset	Create code other quantum codes using Stim	Mara
2.2		Reach		Generate dataset for other quantum codes	Christopher
2.3		Reach		Adapt readout wrapper to accept other datasets	Christopher
2.4		Reach		Adapt embedding scheme to accept different quantum code data (with unique vectorization schemes)	Arjun
2.5		Reach		Train on Stim data for other quantum codes	Christopher
2.6		Reach		Test and evaluate performance on Stim data for other quantum codes	Christopher
3.1	As a graduate student, I would like to compare my decoder with a pre-built transformer-based decoder and baseline models	MVP	Implement baselines	Implement MWPM basline error corrction on simulated Stim data	Arjun
3.2		MVP		Evaluate MWPM basline performance on Stim data	Tzu Chen
3.3		MVP		Implement MWPM basline error corrction on Sycamore data	Tzu Chen
3.4		MVP		Evaluate MWPM basline performance on Sycamore data	Mara
3.5		MVP	Implement Gogle's transformer model	Build branch from neural net pipeline to send embedded data to Google's transformer model	Tzu Chen
3.6		MVP		Train Google's transformer model on Stim Data	Christopher
3.7		MVP		Evaluate Google's transformer performance on Stim data	Tzu Chen
3.8		MVP		Train Google's transformer model on Sycamore Data	Arjun
3.9		MVP		Evaluate Google's transformer performance on Sycamore data	Christopher
3.10		MVP	Evaluation Module	Compare performance with proposed solution using LER (p) evaluation metric	Arjun
3.11		MVP		Generate visualizations and report	Mara
4.1	As a graduate student, I want to integrate a neural-network quantum decoder solution into a simplified quantum model.	Reach	Real-time adoption	Adapt readout module to listen and accept realtime error data	Arjun
4.2		Reach		Feed into pipeline of transformer model that evaluates error correction output	Tzu Chen
4.3		Reach		Evaluate and verify real-time performance and throughput	Tzu Chen
4.4		MVP	Interface and packaging	Construct interface to provide option to modify parametrs of	Mara

ID	Priority	Est.	Acceptance Criteria
1.1	High	0.75	Accessible github repository in which everyone can run and edit simple Readme file
1.2	High	1.5	Only when environment is active does file inside run
1.3	High	1	Verify image and container exists using docker ps or docker container ls
1.4	Low	0.75	Check if helloworld python file runs only after running container
1.5	Medium	4.5	Stim circuits created for d ∈ {3,5} and specified noise model(s).
1.6	Medium	3.75	Script generates train/val/test splits with configurable shots and p values.
1.7	Low	3	Wrapper accepts soft information (e.g., logits/LLRs/probabilities) and returns calibrated analog values.
1.8	High	3.75	SSH key + MFA working; sinfo & sbatch accessible. SLURM template committed (e.g., slurm/train_surface.sh)
1.9	Medium	7	Deterministic mapping from syndrome/analog readout to tokens/embeddings.
1.10	Medium	5.25	Model modules implemented with clear interfaces and docstrings. End-to-end train.py and eval.py accept con
1.11	Medium	4.75	SLURM job(s) complete without error; logs and checkpoints saved.
1.12	Medium	2.5	eval.py runs across held-out splits and p-grid; outputs LER(p) table.
1.13	Medium	4	At least N (e.g., 5) controlled experiments executed (documented changes only). Ablation table added to repo
1.14	High	3.25	Loader for Sycamore data implemented; normalization/calibration documented.
1.15	Medium	5.25	SLURM job completes. Checkpoints/logs archived separately from Stim runs.
1.16	Medium	2.75	LER(p) (or per-round logical failure) computed on held-out Sycamore set.
1.17	Medium	3.75	At least N (e.g., 5) controlled experiments executed for Sycamore daataset specifc challenges (documented c
2.1	Low	7.25	At least two additional codes (e.g., color code, XZZX surface) implemented.
2.2	Low	4.5	Datasets built with the same schema; metadata includes code_family.
2.3	Low	4.25	Wrapper adapts to code family differences via config (no code changes).
2.4	Low	6.25	Single embedding API supports all code families (feature flags). Validation asserts consistent tensor shapes
2.5	Low	5.25	Training runs complete for each additional code (min one config each).
2.6	Low	5.5	LER(p) curves produced and saved for each code.
3.1	Medium	3.75	MWPM decoder (e.g., PyMatching) integrated with same data interface.
3.2	Medium	2.5	LER(p) for MWPM on surface code computed and saved.
3.3	Medium	4	Sycamore loader into MWPM path runs end-to-end.
3.4	High	3.5	LER results computed; plot and results exported.
3.5	Low	4.25	Separate training path --model google_transformer implemented.
3.6	Medium	5	Training completes with logged metrics and checkpoints.
3.7	Low	2.75	LER(p) computed and saved. Plots exported.
3.8	Medium	5	SLURM job completes. Artifacts saved.
3.9	High	3.25	Evaluation artifacts produced. Export plots and results.
3.10	High	3.5	- Unified comparison plot with all decoders (ours, Google's, MWPM) for Stim & Sycamore. - Table reports LER at each p, relative improvement vs MWPM, and crossover points. - AUC (LER vs p) computed; bootstrap CIs provided. - Narrative summary added to report.
3.11	Medium	3.75	Figures: LER(p), training curves, ablations, throughput/latency, calibration.
4.1	Low	2.75	Streaming input (e.g., ZeroMQ/Kafka/pipe) interface implemented.
4.2	Low	3.5	End-to-end streaming decode runs continuously ≥30 minutes without crash.
4.3	Low	4	Latency (p50/p95) and throughput measured at batch=1 and batch=256.
4.4	Medium	4.25	CLI + minimal UI (e.g., Streamlit or REST) exposes key params (depth, heads, lr, batch, calibration). Validation