Semester Long Internship 2026

Duration: 1 February 2026 – 30 June 2026

Project Report

On

AI-Assisted Python Tutoring:

Persona-Driven Hint Generation and LLM-Based Error Classification Research

Submitted by:

Rudrapratap Shinde

Vidyalankar Institute of Technology, Mumbai, India

Under the guidance of:

Dr. Kushal Shah

Prof. Prabhu Ramachandran

July 15, 2026

Acknowledgements

I would like to express my sincere gratitude to the FOSSEE team at IIT Bombay for the opportunity to work on this internship. I am especially thankful to my supervisors, Dr. Kushal Shah and Prof. Prabhu Ramachandran, for their sustained guidance, timely feedback, and encouragement throughout the five-month internship period.

I would also like to thank the wider FOSSEE team and my fellow interns with whom I collaborated closely on the large-scale error-labeling effort described in this report. Working alongside them made a genuinely large dataset (11,622 programs) tractable and taught me a great deal about coordinating a shared annotation workflow.

I would like to acknowledge **MANINI JAISWAL**, my fellow FOSSEE intern, with whom the entire project was completed collaboratively. The work described in this report reflects our joint efforts throughout the internship, and I sincerely appreciate their dedication, technical contributions, and collaborative spirit.

Finally, I thank my family and friends for their support during the course of this internship.

Abstract

This report presents the work completed during a five-month internship at FOSSEE, IIT Bombay, focused on two connected strands of AI-assisted programming education: persona-driven hint generation for beginner Python students, and large-scale LLM-based classification of student coding errors.

The first strand covers the design and evaluation of a tutoring prompt built around the teaching persona of Prof. H.C. Verma, IIT Kanpur physicist and author of Concepts of Physics. The persona was selected after a three-round shortlisting process that considered fictional archetypes, Indian public figures, and globally recognised personalities before a supervisor-suggested academic shortlist narrowed the choice to Richard Feynman, Socrates, and H.C. Verma. The final prompt preserves the original FOSSEE tutoring rules — no direct answers, four-to-five-line responses, an incomplete-program rule with no hints — while adding analogy-first, brick-by-brick explanations tailored to Indian STEM beginners. The persona prompt was applied across 40 evaluation entries spanning execution and syntax errors, and benchmarked qualitatively against the original neutral-voice prompt on tone, specificity of encouragement, and cultural relatability.

The second strand involved contributing to a research pipeline for classifying student errors from the Yaksh buggy-program dataset (2.81M submissions, 398 unique questions). This included manually labeling a 100-program reference set against an evolving error taxonomy, participating in a four-way team split that ran an LLM labeling pass across all 11,622 unique buggy programs in the PayPal dataset, curating a balanced 189-program subset for deeper manual verification, and reviewing and fixing the evaluation codebase. A scoring audit identified and corrected a lowercase-normalisation gap affecting 34 ground-truth rows, raising corrected Jaccard scores for all six benchmarked models by 0.015–0.090 points. A temperature variation study on GPT-OSS-120B further revealed that the default temperature setting used in the original paper was substantially better calibrated for classification alignment than either temperature 0.5 or 1.0.

These two strands address complementary parts of the same problem: how to explain a student's mistake in a way that builds genuine understanding, and how to classify that mistake reliably enough for the explanation to be pedagogically appropriate.

Table of Contents

List of Tables

1 Introduction

1.1 Motivation

Beginner Python students on platforms such as Yaksh typically receive one of two things when they submit incorrect code: a raw interpreter traceback, or a generic tutoring message that is technically correct but reads as flat, system-generated text. Neither helps a first-time programmer understand what went wrong or feel supported while debugging.

This problem is amplified at scale. The Yaksh platform at IIT Bombay hosts 2.81 million student submissions across 398 unique programming questions from introductory courses making human, one-on-one feedback practically impossible at the rate students need it. The distribution of error types across this corpus is shown below.

This internship project was conducted jointly with MANINI JAISWAL, and the outcomes documented in this report reflect our collaborative efforts across all stages of the project.

Table 1. Distribution of student error categories across the Yaksh submission corpus (2.81M submissions).

Error Category	Count	Percentage (%)
WrongAnswer	401,285	35.0%
SyntaxError	372,906	32.6%
NameError	142,148	12.4%
TypeError	92,294	8.1%
AttributeError	34,456	3.0%
ValueError	30,033	2.6%
AssertionError	18,893	1.6%
TimeLimitExceeded	25,001	2.2%
Other Runtime Errors	28,348	2.5%

This internship addressed two pieces of that gap: making automatically generated hints feel like guidance from a trusted teacher rather than a validator output and making the underlying error classification that hints depend on more accurate and auditable.

1.2 Objectives

- Design and evaluate a teaching-persona tutoring prompt for the FOSSEE Python platform, built on the existing hint-based pedagogical rules.
- Curate and manually label a reference set of buggy Python programs against an error taxonomy and refine that taxonomy based on labeling disagreements.
- Contribute to a team-run LLM labeling pass across the full 11,622-program PayPal dataset and manage an assigned chunk of 2,905 programs.
- Review, debug, and correct the shared evaluation codebase used to score model-predicted labels against ground truth.
- Audit the Jaccard scoring pipeline for correctness and reconcile discrepancies between internally computed scores and previously reported paper scores.
- Investigate the effect of temperature variation on LLM classification performance and label-set stability.

1.3 Scope and Contributions

This work spans two interconnected tasks that together support the FOSSEE error-tutoring research effort: communicating an error to a beginner in a way that builds understanding and classifying that error correctly and reproducibly in the first place.

Table 2. Summary of internship tasks and key deliverables.

Task	Description	Key Deliverables
Task 1	H.C. Verma persona design and hint-prompt evaluation	Final prompt, 40 evaluated entries (20 execution + 20 syntax), persona comparison report
Task 2	Dataset curation, taxonomy refinement, and codebase fixes for LLM error classification	Labeled datasets, taxonomy revisions, corrected scoring pipeline, temperature study, Jaccard audit

2 AI Tutoring Prompt: The H.C. Verma Persona

The first task built on FOSSEE's existing hint-based tutoring prompt for the Python platform. That prompt already enforced a sound pedagogical structure — never reveal the solution, limit responses to four or five lines, align feedback with the error type, and withhold hints entirely when a submission is incomplete. What it lacked was voice: the outputs read as correct but impersonal, and encouragement phrases such as "Nice start" repeated so often across entries that students learned to skim past them.

2.1 Persona Selection Process

Rather than hand-tune individual responses, the decision was made to wrap the existing rules in a consistent teaching persona. Three rounds of candidates were considered before the final character was confirmed.

Round 1 — Fictional and Archetypal Characters

Table 3. Round 1 candidate personas — fictional and archetypal characters.

Character	Voice Style	Reason for Rejection
Merlin the Wizard	Mystical, metaphor-heavy	Too abstract for technical feedback
ARIA the Robot	Logical, step-by-step	Reproduced the same mechanical flatness as the original prompt
Captain Syntax	Adventure, pirate energy	Inconsistent tone for academic platform
Dr. Py (Scientist)	Experimental, analytical	Too domain-specific, lacked warmth
Pixel (Gamer)	Game-based analogies	Limited reach beyond gaming audience
Rookie (Peer Tutor)	Casual, peer-level	Lacked authority and depth for debugging guidance

Round 2 — Indian Public Figures

Table 4. Round 2 candidate personas — Indian public figures.

Character	Voice Style	Reason for Rejection
Amitabh Bachchan	Dramatic, KBC quiz style	Too theatrical for technical guidance
APJ Abdul Kalam	Inspirational, dream-focused	More motivational than instructional
Kapil Sharma	Comedy, Punjabi humor	Inconsistent with academic seriousness
Sundar Pichai	Professional, Silicon Valley	Too polished, not warm enough
Virat Kohli	Sports coach, discipline	Coaching style doesn't map to debugging
Ranveer Singh	Hyper-energetic	Tone too extreme for calm guidance

Round 3 — Supervisor's Academic Shortlist (Final Three)

After a ten-candidate shortlist of globally recognized figures was submitted for supervisor review, the supervising professor suggested three alternative candidates rooted in academic and intellectual tradition:

Table 5. Round 3 candidate personas — supervisor's academic shortlist.

Character	Teaching Voice	Key Strength	Key Risk	Outcome
Richard Feynman	Playful, curious, thinks aloud	Exceptional clarity through curiosity	Risks drifting into storytelling within tight line budget	Runner-up
Socrates	Calm, probing — teaches only by questions	Perfect hint-style match	Dialogue-based teacher; awkward in one-way AI response	Runner-up
H.C. Verma	Warm, patient, structured — trusted elder	Best fit for Indian STEM beginners; universally trusted	Physics association needs bridging to coding context	SELECTED

WHY	*H.C. Verma's Concepts of Physics is the most beloved and trusted physics textbook among Indian students preparing for competitive examinations. His voice is instantly trusted by Indian STEM students, and his approach — calm, structured, and deeply patient maps directly to the hint-based teaching philosophy at the core of this project.*

2.2 Persona Design and Traits

The confirmed persona was built around six traits, each mapped onto a concrete prompting technique rather than left as a vague tone instruction.

Table 6. Persona traits and their implementation in the tutoring prompt.

Trait	How It Was Implemented in the Prompt
Brick-by-brick building	Every response opens by naming what the student did correctly before addressing the error.
Genuine, specific encouragement	Generic praise ('good job') was removed; praise names the exact correct step the student took.
Thinking alongside, not above	First-person collaborative phrasing 'let us look at this together', 'ask yourself'.
Analogy-first explanations	Every abstract concept (type mismatch, index error, scope) is introduced through an everyday, India-specific analogy before any programming term is used.
Precision without intimidation	Phrases like 'not almost right exactly right' carry Verma's physics-teaching exactness into syntax feedback without shaming.
Patient, varied endings	Responses close with a directive, a question, or a principle depending on what the specific error needs never one repeated sign-off.

Every rule from the original FOSSEE prompt was preserved unchanged:

- No corrected code, no final answer, no step-by-step solutions
- Responses limited to 4–5 short lines only
- Exactly two hints for logic errors; zero hints for incomplete programs
- Feedback aligned to error type: syntax, logic, name, type, runtime, index, indentation

2.3 Evaluation Across 40 Entries

The final prompt was applied to all 20 curated Execution Error cases and all 20 curated Syntax Error cases from the Yaksh buggy-program set, spanning logic, type, name, index, and indentation errors. Representative examples from each category are shown below.

Table 7. Representative evaluation cases across error types and analogies used.

#	Error Type	Core Issue	Analogy Used
1	Logic	Function returns input instead of computed result	Shopkeeper handing back the measuring tape instead of the computed area
6	Logic	Total overwritten instead of accumulated	Emptying a coin box every time a new coin arrives — not collecting, replacing
7	Logic	Comparing before count is complete	Declaring a cricket match winner after just three overs
9	Type	Arithmetic attempted on unconverted string input	The word 'three' written on paper versus the actual number 3
12	Type/Logic	Incorrect dictionary key type in lookup	Looking up a dictionary entry with a different spelling than it was stored under
13	Name	Variable name mismatch (fibb vs fib)	Two slightly different spellings — every letter must match, exactly
16	Name	Variable renamed between assignment and use	Labeling a lab bottle one name, then asking for a different name later
17	Index	Assigning into list position that doesn't exist yet	Trying to seat someone in a fifth chair when only three chairs exist
19	Index	Loop goes one step past last valid position	Reading page 6 of a 5-page book — that page simply is not there
4 (Syntax)	Indentation	Block indentation inconsistent with surrounding lines	Soldiers in a line — one out of formation breaks the entire pattern
12 (Syntax)	Indentation	if/else block left with no indented body	A promise: every if/else must be followed by at least one indented instruction
15 (Syntax)	Syntax	Smart quotes pasted from word processor	Curly quotes from a word processor vs. straight keyboard quotes — Python only accepts straight ones

2.4 Sample Outputs: Execution Errors (H.C. Verma Persona)

EX	*Error: Logic Error — Total overwritten instead of accumulated Buggy Code: total = int(num) Persona Output: You are reading the numbers and processing them — that is genuine progress. Now look at one thing together. Each time a number arrives, you are replacing the total with it rather than adding it to what you already have. Think of collecting money in a box: every time a coin comes in, you empty the box and put only the new coin inside. That is not collecting that is replacing. What should happen to the box each time a new coin arrives? Analysis: The coin box analogy maps directly onto the accumulation pattern a beginner needs to understand. Ending with a question is deliberate — the logic exists but is wrong, so the student must reason through the fix.*

EX	*Error: Index Error — Assigning into list index that has not been created Buggy Code: result [0] = 0; result [1] = 1 Persona Output: You understand the Fibonacci sequence — that is clear. But Python is giving an index error, which means you are trying to place a value into a position that does not yet exist. Think of a row of chairs: if only three are arranged, you cannot seat someone in the fifth — that chair is simply not there. Before assigning to a position, ask yourself: have I created enough space in the list for that position to exist? Analysis: The chairs analogy makes the idea of pre-allocated space immediate and physical. The question redirects the student to the initialization stage — the actual root of the problem.*

2.5 Sample Outputs: Syntax Errors (H.C. Verma Persona)

SY	*Error: IndentationError — unindent does not match outer indentation level Persona Output: The Fibonacci setup and variable initialization look reasonable — a good foundation. But Python found a problem with spacing. In Python, spacing is not just style — it is grammar. Every line in the same block must begin with the same amount of space, like soldiers in a line: if one steps out of formation, the pattern breaks. Look at the loop body and compare its spacing with the lines just above and below. Is everything aligned? Analysis: 'Spacing is not just style — it is grammar' reframes indentation as a structural rule, not a preference. The soldier's analogy makes alignment a physical, visual concept.*

SY	*Error: Syntax Error - missing closing parenthesis Persona Output: Reading a file and writing selected content into another — you have understood the task well. But Python found a syntax problem in one of your function calls. Parentheses must always come in pairs, like shoes — you cannot have one without the other. Look at the line where you write to the output file. Count the opening parentheses. Does each one have a matching closing parenthesis? Analysis: The shoes analogy is concrete and universal — impossible to misunderstand. The instruction to count parentheses gives an executable action, not a vague directive.*

3 Research on LLM-Based Error Classification

The second task supported an ongoing FOSSEE research effort to classify the kind of mistake present in a buggy student submission using LLMs, so that downstream tutoring prompts can be conditioned on a reliable error label rather than the raw traceback alone. This section covers dataset curation and manual labeling, the taxonomy's evolution, a large-scale team labeling pass, codebase and scoring corrections, a temperature variation study, and a label-distribution analysis.

3.1 Reference Dataset and Manual Labeling

A reference set of 100 buggy Python programs was manually labeled against an initial six-category error taxonomy: Algorithmic Strategy Failure, Interface Contract Failure, State Representation Failure, Control Flow Failure, Computational Failure, and Domain Coverage Failure, plus a No Error Detected category.

This category structure was subsequently revised into an eight-category taxonomy after reviewing where labelers disagreed most. Control Flow was split so that initialization- and scope-related bugs became their own category, and a new Mutability and Reference Traps category was added to capture aliasing and shallow-copy bugs.

Table 8. Manual label distribution across the 100-program reference set.

Label	Category	Final Count	% of 100
B	Interface & Formatting Errors	26	26%
NONE	No Error Detected	25	25%
A	Algorithmic Strategy Failure	22	22%
F	Iteration Errors	13	13%
G	Mutability & Reference Traps	8	8%
D	Data Type & Casting Errors	3	3%
H	Math & Transformation Errors	2	2%
E	Conditional Logic Errors	1	1%

3.2 Large-Scale Labeling: The PayPal 11k Dataset

Following the manual reference pass, the task was to run a PG (Pedagogical-Granular) taxonomy prompt across the full pool of buggy Python programs. The pool contained 11,628 unique buggy programs; six near-empty submissions were removed, leaving 11,622 programs for labeling. The PG taxonomy uses ten labels (A through J) plus a NONE category.

The dataset was split into four roughly equal chunks of about 2,905 programs and divided among the team. This author was assigned the third chunk (programs 5,811–8,715).

Table 9. PayPal dataset chunk assignments across the four-person labeling team.

Chunk	Assigned To	Program Range	Approx. Count
1	Abhishek	1 – 2,905	2,905
2	Tagore	2,906 – 5,810	2,905
3	Rudrapratap	5,811 – 8,715	2,905
4	Manini	8,716 – 11,622	2,907

As a consistency check, Chunk 4 (programs 8,716–11,622) is summarized below. The high 'NONE' rate (~47%) signaled that the taxonomy's NONE category needed tighter criteria going into the next revision.

Table 10. Label distribution for Chunk 4 (programs 8,716–11,622).

Label	Category	Count	% of Chunk
B	Interface Contract Failure	554	19.1%
C	State Representation Failure	476	16.4%
E	Computational Failure	211	7.3%
A	Algorithmic Strategy Failure	171	5.9%
D	Control Flow Failure	111	3.8%
F	Domain Coverage Failure	21	0.7%
NONE	No Error / Undetermined	1,363	46.9%

3.3 Balanced Subset for Deeper Analysis

From the labeled pool, a balanced subset of 189 programs was curated for closer manual verification: an original batch of 162 programs plus 27 additional programs added to fill gaps in under-represented labels (additional NameError and TypeError cases arising from mismatched function signatures). Each selected program was paired with its buggy code, execution feedback, and a multi-label ground-truth annotation. This 289-row combined ground-truth set became the benchmark used to score LLM-predicted labels.

3.4 Evaluation Codebase Fixes

Reviewing the shared scoring codebase (main_fixed.py) surfaced three related correctness bugs that were silently inflating or corrupting reported results, all fixed during this internship.

Table 11. Correctness fixes applied to the shared evaluation codebase.

Fix	Problem Found	Correction Applied
Fix 1A — Per-taxonomy VALID_OUTPUTS	A single shared VALID_OUTPUTS set (A–J, NONE) was applied to every taxonomy, so the narrower CG and SL taxonomies silently accepted out-of-vocabulary labels (~16 corrupted CG and ~8 corrupted SL assignments).	VALID_OUTPUTS split per taxonomy: CG restricted to {A–F, NONE}, SL to {A–G, NONE}. Illegal labels are now rejected at point of generation.
Fix 1B — Taxonomy-aware sanitize_output()	sanitize_output() built its label-matching regex from the full A–J vocabulary regardless of which taxonomy was active.	Function now takes a taxonomy argument and builds its regex from that taxonomy's legal label set only.
Fix 1C — Taxonomy-aware is_row_complete()	is_row_complete() checked labels against the shared vocabulary, so a row with illegal label (e.g. 'G' under CG) was treated as complete and never re-queried.	Completeness check now validates against the taxonomy-specific legal set, forcing re-query on any row holding a corrupted label.
Fix 2 — Jaccard denominator bug	average_jaccard() divided summed Jaccard scores by 100 (the full program pool) and treated both-	New jaccard() function returns (score, is_labeled) tuple. average_jaccard() divides only by

Fix	Problem Found	Correction Applied
	empty prediction/ground-truth pairs as a perfect score of 1.0 — 19 unlabeled programs padded every model's reported average by ~0.07–0.10.	the count of rows actually labeled (81), excluding both-empty rows entirely.

3.5 Scoring Audit: Why Our Scores Differ from the Paper

After applying the denominator fix, the recomputed Jaccard scores remained noticeably higher than the scores previously reported for the same models in the research paper. Tracing the discrepancy back to the ground-truth file (ground.csv) found that 34 rows carried labels in lowercase (e.g. 'j,i,b,d' instead of 'J,I,B,D'), while every model's predictions were always uppercase.

Where the scoring script did not normalise case before comparing, those 34 rows produced an empty ground-truth set, which forced Jaccard = 0 for all 34 rows regardless of whether the prediction was correct. Examples of the per-row impact:

Table 12. Per-row scoring impact of the lowercase-normalization gap.

Sr. No.	Ground Truth (raw)	Predicted	Paper Score (no norm)	Corrected Score
52	a,b,i	ABI	0.000	1.000
56	j,i,b,d	CDEFGHIJ	0.000	0.364
84	d,e,g,h,j	DEGHJ	0.000	1.000
90	b,i,d	BID	0.000	1.000
95	d,e,g,j	DEGJ	0.000	1.000

Adding uppercase normalization before scoring corrected all 34 affected rows and raised the measured score for every benchmarked model:

Table 13. Corrected Jaccard scores by model after the normalization fix.

Model	Paper Score (Multi, Iter 1)	Corrected Score	Difference
DeepSeek v3.2	0.2384	0.2802	+0.0418
GPT-OSS 120B	0.2810	0.3283	+0.0473
Qwen3-Coder	0.2633	0.2787	+0.0154
Claude Sonnet 4.5	0.2757	0.3433	+0.0676
Gemini 2.5 Flash	0.2608	0.2576	−0.0032
GPT-5.2	0.3278	0.4180	+0.0902

The corrected pipeline was then run across both available iterations and both single-label and multi-label prompting modes for all six models:

Table 14. Multi- and single-label Jaccard scores across two iterations.

Model	Iter 1 Multi	Iter 1 Single	Iter 2 Multi	Iter 2 Single
DeepSeek v3.2	0.2802	0.2440	0.2724	0.1982

Model	Iter 1 Multi	Iter 1 Single	Iter 2 Multi	Iter 2 Single
GPT-OSS 120B	0.3283	0.2098	0.3462	0.1898
Qwen3-Coder	0.2787	0.2023	0.2687	0.1923
Claude Sonnet 4.5	0.3433	0.2082	0.3436	0.2090
Gemini 2.5 Flash	0.2576	0.2157	0.2744	0.1915
GPT-5.2	0.4180	0.2248	0.4241	0.2115

KEY FINDING	*Multi-label mode outperforms single-label mode for every model in both iterations. GPT-5.2 is the strongest model overall (0.4241 multi-label, Iter 2). GPT-OSS-120B is the strongest freely available model. Sonnet's scores are the most stable across iterations. DeepSeek's scores drop from Iter 1 to Iter 2, suggesting lower run-to-run consistency.*

3.6 Temperature Variation Study

A dedicated temperature variation experiment was conducted to understand how sampling temperature affects GPT-OSS-120B's classification performance and label consistency. GPT-OSS 120B was the only free model from the paper that remained consistently accessible and stable throughout the experiment; all findings in this section therefore pertain exclusively to that model.

Three conditions were compared: the original run (default temperature, as used in the paper), temperature 0.5, and temperature 1.0. Three iterations were run at each temperature setting using the PG taxonomy in multi-label mode.

Jaccard Score Comparison

Table 15. Jaccard score comparison across temperature settings (GPT-OSS-120B).

Run	Temperature	Jaccard Score	vs. Original
Original (Iter 1)	Default	0.3283	—
Temperature 0.5	0.5	0.1720	−0.1563
Temperature 1.0	1.0	0.1851	−0.1432

KEY FINDING	*Both temperature 0.5 and temperature 1.0 produce substantially lower Jaccard scores than the original run (~0.15 points lower). The default temperature used in the original paper is significantly better calibrated for classification alignment than either explicitly tested temperature.*

Inter-Run Consistency

Inter-run consistency measures how similar label predictions are between two runs, independent of ground truth. A higher Jaccard score between runs indicates more stable outputs.

Table 16. Inter-run consistency across temperature settings.

Comparison	Avg Jaccard	Interpretation
Original vs Temp 0.5	0.1611	Low agreement — very different label sets
Original vs Temp 1.0	0.1544	Low agreement — very different label sets

Comparison	Avg Jaccard	Interpretation
Temp 0.5 vs Temp 1.0	0.4260	Moderate agreement — the two temperatures are more similar to each other than to the original

Label Frequency Shift Across Temperatures

Table 17. Label frequency shift across temperature settings.

Label	Description	Original	Temp 0.5	Temp 1.0	Δ (0.5)	Δ (1.0)
A	Loop Condition Misunderstanding	24	9	5	−15	−19
B	Input/Output Handling Error	38	23	22	−15	−16
C	Conceptual Gap in Algorithm	24	21	19	−3	−5
D	Variable Misuse or Confusion	39	31	24	−8	−15
E	Boundary/Edge Case Ignorance	28	5	6	−23	−22
F	Incorrect Operator or Expression	14	24	25	+10	+11
G	Control Flow Misunderstanding	16	10	9	−6	−7
H	Data Structure Misuse	9	1	2	−8	−7
I	Specification Misread	44	27	26	−17	−18
J	Accumulation/Aggregation Error	44	9	8	−35	−36

Labels I (Specification Misread) and J (Accumulation/Aggregation Error) show the sharpest drops at both temperature settings. Label F (Incorrect Operator or Expression) shows a notable increase at both temperatures, suggesting temperature-induced shift toward operator-level error detection. Labels B, C, and D remain comparatively stable — robust classifications less sensitive to temperature changes. The abstention rate (N/null predictions, ~26/100) is temperature-invariant, indicating model abstention is driven by program difficulty rather than sampling randomness.

3.7 Label Distribution and the Case Against Label F (Branching)

With 81 of the balanced subsets fully labeled, the PG-taxonomy label distribution was reviewed against a target of roughly 20 examples per label for a well-powered evaluation.

Table 18. PG-taxonomy label distribution against a 20-per-label target.

Label	Name	Count	Share %	Gap to 20	Status
A	Input	18	22.2%	+2	OK
B	Output	44	54.3%	−24	DOMINANT
C	Variable	3	3.7%	+17	RARE
D	Computation	37	45.7%	−17	OK
E	Condition	11	13.6%	+9	LOW
F	Branching	0	0.0%	+20	ABSENT

Label	Name	Count	Share %	Gap to 20	Status
G	Loop	7	8.6%	+13	LOW
H	Array/String	9	11.1%	+11	LOW
I	Function	43	53.1%	−23	DOMINANT
J	Conceptual	48	59.3%	−28	DOMINANT

Total labeled: 81 | Target per label: 20 | Total needed: 200 | Shortfall: ~119 more programs

The complete absence of Branching (Label F) in the ground truth motivated a closer look at how models handled that label:

Table 19. Label F (Branching) assignment across human annotators and models.

Model	Label F Assigned	Expected (Human GT)	Assessment
Human annotators	0	0	Correct — F absent in this dataset
DeepSeek v3.2	0	0	Correct
Claude Sonnet 4.5	7	0	7 hallucinated F assignments
GPT-5.2 (ChatGPT)	7	0	7 hallucinated F assignments
GPT-OSS-120B	7	0	7 hallucinated F assignments
Qwen3-Coder	Unknown	0	Needs API run to confirm
Gemini 2.5 Flash	Unknown	0	Needs API run to confirm

The root cause: models were consistently applying Branching (F) to any conditional-related error, including ones that were purely expression-level Condition (E) errors. A related pattern surfaced in Sonnet's zero-match cases: 12 of 23 collapsed to a generic Conceptual (J) label even when a more specific label clearly applied, indicating the model was using the broadest label as a fallback.

Table 20. Trade-off comparison for handling Label F (Branching).

Option	Action	Trade-off	Recommendation
A — Remove F (for paper)	Delete F from VALID_OUTPUTS["PG"]; re-run evaluation	Clean metric; DeepSeek/Qwen unchanged; Sonnet/GPT-OSS improve	Recommended for final paper results
B — Keep F with prompt note	Add NOTE in PG prompt: F = block structure wrong, NOT expression wrong	May still hallucinate without real F examples in dataset	Future dataset expansion — collect 10+ genuine F programs

4 Connection to the Research Paper

This internship ran in parallel with the preparation of a formal research paper by the FOSSEE team: "LLM-Based Error Explanation and Logical Error Classification in Python Programming Education". The work in this internship directly fed into the paper's dataset, evaluation methodology, and scoring pipeline.

Table 21. Mapping of internship contributions to research paper components.

Paper Component	Internship Contribution
100-program benchmark corpus	Manually labeled as part of the initial reference-set curation; taxonomy revised from six to eight categories based on labeling disagreements surfaced during this internship.
Syntax error multi-stage pipeline evaluation (69% preference rate, 98% non-degradation)	The same 100 programs used in Section III of the paper were also the evaluation corpus for the logical error taxonomy evaluation. This author's manual labels on those programs contributed to the ground truth used to assess model outputs.
Logical error taxonomy evaluation across CG, SL, PG schemes	Large-scale PayPal labeling pass (2,905 programs by this author) provided the empirical basis for label distribution estimates and calibration decisions, including the Branching-label removal recommendation.
Jaccard scores for all six models across PG taxonomy	The scoring audit performed in this internship identified and corrected the lowercase normalization gap in ground.csv, raising corrected scores for five of six models. The corrected scores are 0.015–0.090 higher than the paper's originally reported figures for those models.
H.C. Verma persona tutoring prompt	Directly adopted by the FOSSEE platform as the deployed tutoring voice, with the full 40-entry evaluation documented in the companion FOSSEE persona report.

The paper's central finding — that taxonomy structure governs model performance more than model choice — is supported by the label-distribution analysis and Branching-label investigation documented in this report. The paper's preference for coarser taxonomies (CG, average Jaccard ≈0.69) over fine-grained structural taxonomies (SL, average Jaccard ≈0.49) aligns with the finding here that even the PG taxonomy's Branching label — which is a relatively fine structural distinction — was systematically hallucinated by closed-source models in the absence of genuine positive examples in the training data.

5 Conclusion

5.1 Summary of Contributions

Table 22. Summary of contributions across the internship.

Area	Contribution	Impact
Persona Design	Selected and designed the H.C. Verma teaching persona after a three-round candidate review; layered it onto FOSSEE's existing rule-based tutoring prompt without weakening any pedagogical guardrails; evaluated the resulting prompt across 40 execution and syntax error cases.	Deployed on FOSSEE Python tutoring platform. Enables culturally resonant, analogy-first, student-centered feedback for Indian STEM beginners.
Dataset Curation	Manually labeled a 100-program reference set; contributed to revising the error taxonomy from six to eight categories based on labeling disagreements; curated a balanced 189-program subset with multi-label ground truth.	Provides the benchmark evaluation corpus used in the research paper and subsequent scoring runs.
Large-Scale Labeling	Ran the PG-taxonomy LLM labeling prompt across an assigned chunk of 2,905 programs (Sr. No. 5,811–8,715) as part of a four-person effort covering all 11,622 unique programs in the PayPal dataset.	Enables label-distribution estimates across the full PayPal corpus; identifies category imbalances informing taxonomy revisions.
Codebase Fixes	Corrected a shared-vocabulary bug (per-taxonomy VALID_OUTPUTS), made sanitization and completeness checks taxonomy-aware, and fixed a Jaccard-averaging denominator bug that let unlabeled rows inflate every model's reported score.	Eliminates silent corruption of classification results; ensures evaluation metrics are computed over the correctly labeled subset only.
Scoring Audit	Identified and corrected a lowercase-normalization gap affecting 34 ground-truth rows, raising corrected Jaccard scores for all six benchmarked models by 0.015–0.090 points; diagnosed a Branching-label hallucination pattern leading to a taxonomy-simplification recommendation.	Establishes the correct baseline scores for the research paper; removes confounded metric inflation from the evaluation.
Temperature Study	Conducted a temperature variation experiment across conditions 0.0, 0.5, and 1.0 for GPT-OSS-120B, quantifying the effect on Jaccard alignment and inter-run consistency.	Confirms that the default temperature used in the original paper is well-calibrated for classification; provides empirical basis for temperature recommendations in future experiments.

5.2 Limitations and Future Work

The persona evaluation is qualitative: the 40 entries were assessed for tone, analogy fit, and rule adherence, but no controlled study measured whether the H.C. Verma persona improves student learning outcomes or engagement compared to the neutral baseline. A follow-on A/B evaluation with real students on the Yaksh platform would be the natural next step.

On the classification side, the balanced ground-truth set stood at 81 fully labeled programs against a target of 200 at the time of this report, so several rare labels remain under-powered. The team-wide labeling pass used a single model per chunk without cross-validation between team members, so inter-annotator agreement across the four assigned ranges has not yet been measured.

- Extend the balanced subset toward the 200-program target, with additional programs concentrated on under-represented labels (Variable, Loop, Array/String).
- Cross-check a sample of each team member's chunk against another labeler or model to estimate inter-chunk consistency.
- Re-run the full benchmark once the Branching-label taxonomy change is finalized and collect 10+ genuine Branching examples for a future dataset expansion.
- Conduct an A/B study with real students on the Yaksh platform comparing the H.C. Verma persona prompt against the original neutral-voice baseline on measurable outcomes (time-to-fix, re-submission rate, student satisfaction).
- Run the full temperature experiment across all five intended temperature settings (0.0, 0.2, 0.5, 0.8, 1.0) and three iterations per setting for all three free models (DeepSeek, Qwen, GPT-OSS) when API access is restored.

6 References

[1] Yaksh, "Live Python Coding Environment." https://yaksh.fossee.in/

[2] H.C. Verma, Concepts of Physics, Bharati Bhawan, Patna.

[3] DeepSeek-AI, "DeepSeek-V3.2 Technical Overview," model documentation, 2025.

[4] H. Mishra, Y. Pandey, H. Priyanka, T. Rai, P. Ramachandran, and K. Shah, "LLM-Based Error Explanation and Logical Error Classification in Python Programming Education," FOSSEE, IIT Bombay, 2026. [Research paper — under submission]

[5] H. Priyanka, "Multi-and-single-llm," https://github.com/hpriyankaa/Multi-and-Single-LLM, 2026.

[6] Harshit, T. Rai, and Yuvraj, "Comprehensive human–model evaluation across three logical error taxonomies," https://github.com/atmabodha/fossee-iitb, 2026.

[7] FOSSEE Python Teaching Project, "H.C. Verma Persona Report — AI Tutoring Prompt Evaluation," Internal report, FOSSEE IIT Bombay, 2026.

[8] FOSSEE Python Teaching Project, "Project Summary: Prompt Evolution, Persona Design, Beginner-Friendly Strategy," Internal report, FOSSEE IIT Bombay, 2026.