



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**Evaluating the Reliability of AI-Assisted Code Generation Using
Large Language Models for Small-Scale Software Engineering Tasks**

MD Azizul Khan
2514708

ACFA7004- Research Methods and Professional Development



Research Background



- LLMs such as GPT, LLaMA models, CodeT5, StarCoder, and tools like ChatGPT and GitHub Copilot are now widely used for code generation, debugging and documentation [1], [2], [3].
- Python is a popular target language due to its readability and strong use in data-driven applications [2].
- While LLM-generated code is often practical, it can include hidden logical mistakes or partially developed reasoning [1], [6].



Research Problem



- LLMs often produce confident sounding but unreliable Python code for small, everyday tasks.
- Existing studies focus mainly on large benchmarks, not practical real-world tasks.
- Limited research explores how developers adjust trust when LLMs make mistakes.
- The relationship between code reliability and trust calibration remains under-explored.



Literature Support for the Identified Research Problem

- Research shows that LLMs can generate code that appears correct at first glance but is actually inaccurate or inconsistent, even for relatively simple tasks [1], [2].
- Existing research frequently uses benchmark suites such as HumanEval-style tasks, which do not reflect practical, everyday developer workflows [1], [3].
- Trust-calibration work shows that users struggle to judge when to trust or question LLM outputs, especially when models present errors with high confidence [4], [5].
- Although reliability and trust have been examined separately, there is limited research that connects code correctness, model confidence, and developer trust behaviour within code-generation contexts [3], [6].



Research Aim & Objectives

Aim

- Investigate how different LLMs generate accurate python code for small-scale code writing tasks.

Objectives

- Design a set of small Python tasks for testing.
- Generate solutions from multiple LLMs.
- Use automated unit tests to measure correctness and identify error patterns.
- Compare models on accuracy, consistency, exception handling and code quality.
- Interpret findings through trust calibration concepts.



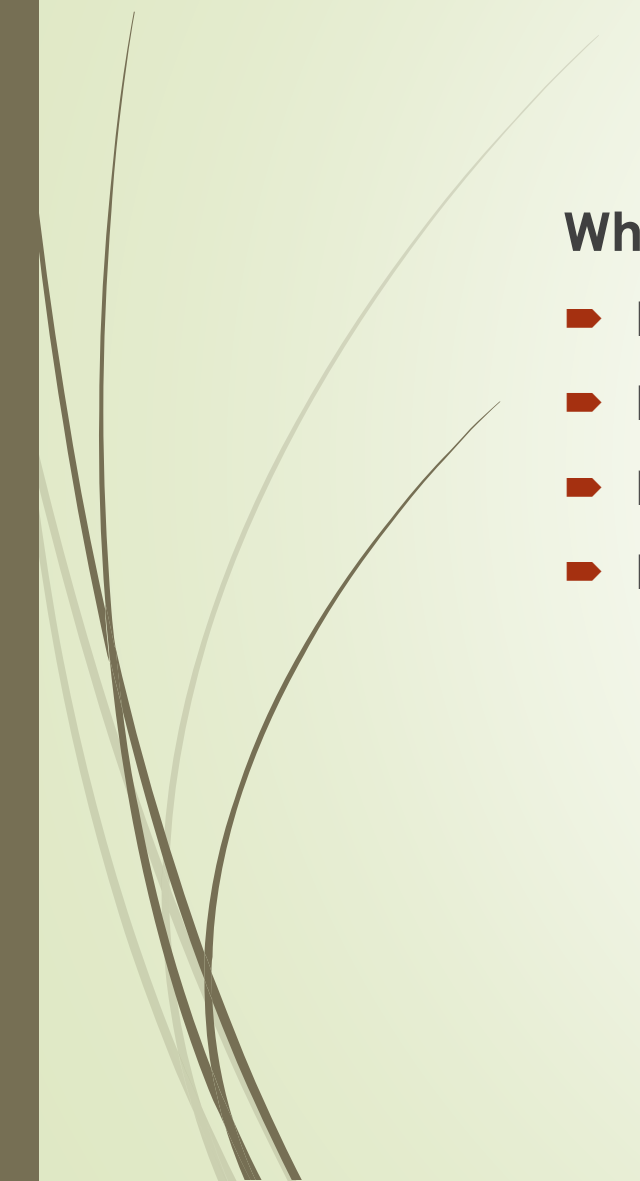
Literature Review Summary & Gap

What research shows

- Recent studies show that LLMs such as GPT, LLaMA-based models, CodeT5, StarCoder, ChatGPT and GitHub Copilot are widely used for code generation, debugging and documentation in software development [1]–[3]
- Systematic reviews and comparative evaluations highlight improvements in accuracy, productivity and automation, but also reveal persistent issues in reliability, consistency and error-prone outputs [1], [2].
- Research on trust, privacy and human–AI interaction shows that developers often struggle to calibrate trust appropriately, especially when LLMs produce confident but incorrect or insecure code [4], [5].
- Security-focused studies identify recurring vulnerabilities in LLM-generated code, including CWE-related weaknesses and unsafe patterns [6].

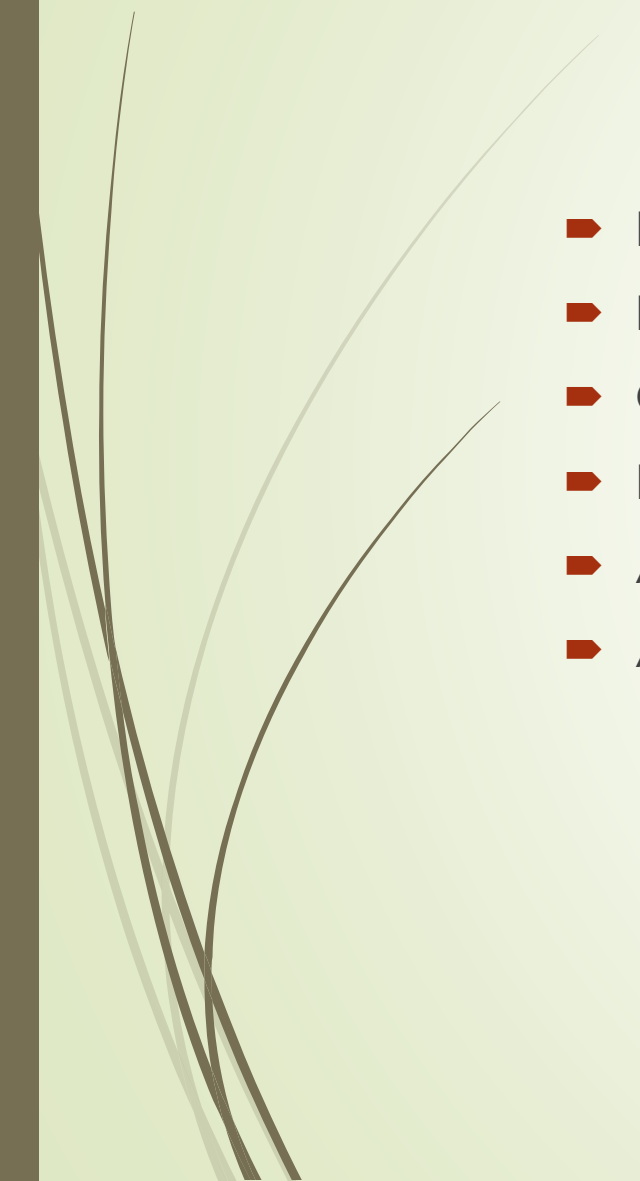


What is missing

- ▶ Limited evaluation of LLM reliability on small, everyday Python tasks.
 - ▶ Lack of structured classification of common error types.
 - ▶ No studies linking LLM code reliability with developer trust calibration.
 - ▶ Few automated, repeatable testing approaches in existing research.
- 



Methodology



- Experimental comparison of multiple LLMs.
- Develop small Python tasks representing common developer activities.
- Generate code from each model.
- Run automated unit tests to assess correctness.
- Apply static analysis to evaluate code quality and potential vulnerabilities.
- Analyse results using trust calibration theory.



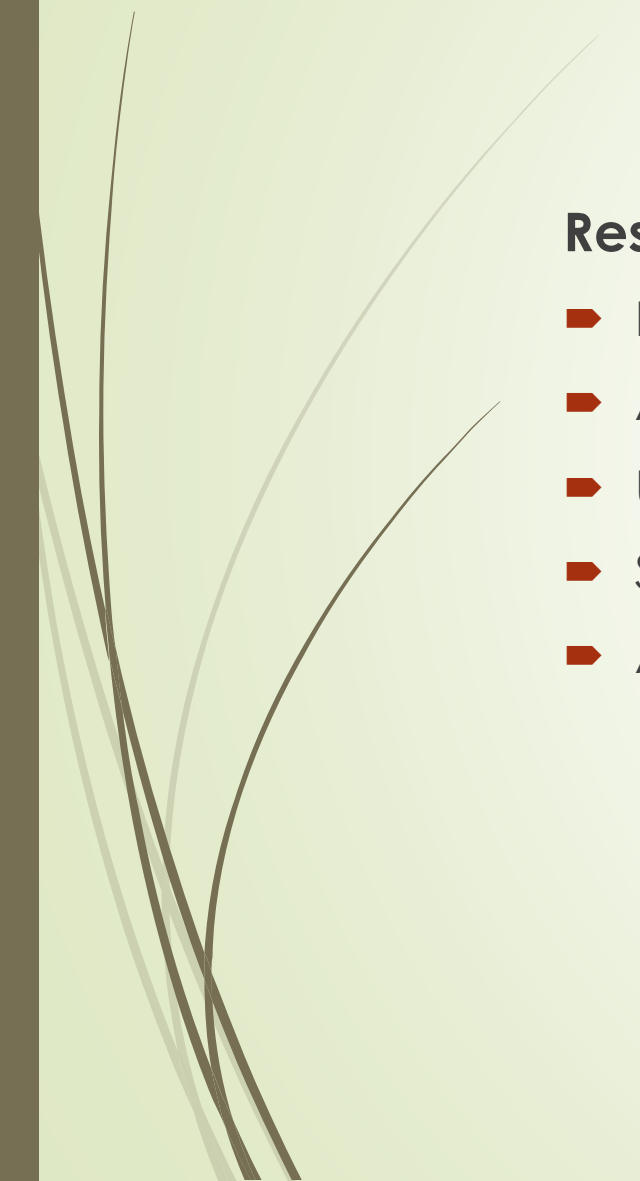
Project Planning & Resources

Timeline

- January- project planning.
- February- Literature review and scoping.
- March- Methodology finalisation and proposal submission.
- April- Experiment setup and run test.
- May- Full experimental evaluation.
- June- Analysis and interpretation.
- July- Writing and refinement.
- August- Finalisation and submission.



Resources

- Python environment.
 - Access to multiple LLMs.
 - Unit testing tools.
 - Static analysis tools.
 - Academic databases.
- 



Project Impact & Limitations

Impact

- Gives a clearer understanding of how reliable LLM-generated Python code really is.
 - Helps developers spot common mistakes and potential risks more easily.
 - Encourages a more realistic and balanced level of trust in AI coding tools.
 - Promotes safer and more responsible use of LLMs in software development.
- 



Limitations

- We were only able to evaluate a limited number of LLMs due to access restrictions.
 - The small set of tasks used in this study may not fully reflect the wide range of real-world programming scenarios.
 - Automated testing cannot identify every semantic flaw or architectural weakness in generated code.
 - Because LLMs are evolving rapidly, the findings may not remain fully applicable over time.
- 



References

- [1] A. Umama *et al.*, “LLM-Based Code Generation: A Systematic Literature Review With Technical and Demographic Insights,” *IEEE Access*, vol. 13, pp. 194915–194939, 2025, doi: 10.1109/ACCESS.2025.3631952.
- [2] D. Palla and A. Slaby, “Evaluation of Generative AI Models in Python Code Generation: A Comparative Study,” *IEEE Access*, vol. 13, pp. 65334–65347, 2025, doi: 10.1109/ACCESS.2025.3560244.
- [3] Z. Ságodi, I. Siket and R. Ferenc, “Methodology for Code Synthesis Evaluation of LLMs Presented by a Case Study of ChatGPT and Copilot,” *IEEE Access*, vol. 12, pp. 72303–72316, 2024, doi: 10.1109/ACCESS.2024.3403858.

- 
- 
- [4] K. Okamura and S. Yamada, “Empirical Evaluations of Framework for Adaptive Trust Calibration in Human-AI Cooperation,” *IEEE Access*, vol. 8, pp. 220335–220351, 2020, doi: 10.1109/ACCESS.2020.3042556.
 - [5] R. Meier, “Balancing Minds and Data: The Privacy Dilemma of LLMs and Anthropomorphism in LLMs,” *Journal of Social Computing*, vol. 6, no. 3, pp. 173–183, Sep. 2025, doi: 10.23919/JSC.2025.0014.
 - [6] J. Zhao *et al.*, “Towards Secure Code Generation With LLMs: A Study on Common Weakness Enumeration,” *IEEE Transactions on Software Engineering*, vol. 51, no. 12, pp. 3507–3523, Dec. 2025, doi: 10.1109/TSE.2025.3619281

Thank You

