Information and Consent to Participate in a Research Study

A. PURPOSE
The purpose of this research is to explore software developers' perceptions, concerns, and experiences when using Large Language Models (LLMs) like ChatGPT for software bug fixing. Insights from this study will support the design of responsible AI usage policies and better tools that enhance transparency, trust, and accountability in AI-assisted programming.

B. PROCEDURES
If you choose to participate, you will complete an anonymous online survey that includes multiple-choice and matrix-style questions about your programming experience, use of LLMs, and views on AI-related risks and responsibilities. Participation will take approximately 25-30 minutes and is entirely voluntary. No identifying information will be collected. The survey is hosted on SurveyMonkey (privacy policy: https://www.surveymonkey.com/mp/legal/privacy).

C. RISKS
There are no known risks to participation.

D. BENEFITS
You may not personally benefit from participation, but your input will help shape ethical guidelines and frameworks for AI-driven development practices.

E. CONFIDENTIALITY
Your responses will be anonymous, and no personal information (e.g., name or email) will be collected. Any quoted feedback will be anonymized. All data will be securely stored in OneDrive for five years post-publication, after which it will be destroyed.

F. CONDITIONS OF PARTICIPATION
Participation is voluntary, and you may withdraw at any time by closing your browser. Once submitted, responses cannot be withdrawn due to anonymity. No compensation is provided.

* 1. By choosing "I give my consent" below, you are indicating that you

- Have read and understand the Consent Form provided to you.
- Consent to participate in the research project.
- Understand that a copy of this consent form is available to you for your records.

Do you give your consent?

- ◯ I give my consent
- ◯ I do not give my consent

Demographic Information

In this section, we invite you to share a few details about your background as a developer.
Your responses will help us better understand the context behind your experience and perspectives on AI usage.

* 2. Please select your age group.

- ◯ Under 18
- ◯ 18 - 24
- ◯ 25 - 34
- ◯ 35 - 44
- ◯ 45 - 54
- ◯ 55 or above

* 3. To which gender identity do you most identify?

- ◯ Woman
- ◯ Man
- ◯ Transgender Woman
- ◯ Transgender Man
- ◯ Gender Variant/Non-Conforming
- ◯ Prefer Not to Answer
- ◯ Prefer to self-describe (Please specify)

* 4. **Including education**, how many years of **programming experience** do you have?

- ◯ Less than 1 year
- ◯ 1 - 3 years
- ◯ 4 - 6 years
- ◯ 7 - 10 years
- ◯ More than 10 years

* 5. **NOT including education**, how many years of **professional programming experience** (for example, industry experience, commercial freelancing, or participation in recognized open-source projects) do you have in software development?

- ◯ Less than 1 year
- ◯ 1 - 3 years
- ◯ 4 - 6 years
- ◯ 7 - 10 years
- ◯ More than 10 years

* 6. What is your current professional affiliation?

- ◯ Academia
- ◯ Industry – Software Development
- ◯ Industry – QA
- ◯ Student
- ◯ Freelance
- ◯ Unemployed
- ◯ Other (please specify)

Experience with Large Language Models (LLMs)

In this section, we invite you to share your personal experiences using LLMs (e.g., ChatGPT, GitHub Copilot) in your software development workflow.

* 7. Please indicate your level of experience in using LLMs (such as ChatGPT, Claude, Gemini, etc.) for bug fixing.

- ◯ Never used LLMs before
- ◯ Less than 1 year
- ◯ 1 – 2 years
- ◯ More than 2 years

* 8. How frequently do you use LLMs for bug fixing?

- ◯ Always (almost daily)
- ◯ Frequently (a few times per week)
- ◯ Occasionally (a few times per month)
- ◯ Rarely (a few time per year)
- ◯ Never

* 9. Please select your level of support for using LLMs in the bug-fixing process.

- ◯ Strongly support
- ◯ Somewhat support
- ◯ Neutral
- ◯ Somewhat against
- ◯ Strongly against

* 10. Please identify your concerns about using LLMs in bug fixing. (Select all that apply)

- [ ] Inaccurate, hallucinated or misleading suggestions leading to new bugs
- [ ] Unclear boundaries of LLM application in bug fixing (e.g., critical vs. non-critical code, production vs. experimental).
- [ ] Security and privacy risks
- [ ] Risk of skill degradation due to excessive LLM dependence
- [ ] Lack of proper training on how to use LLMs effectively and safely
- [ ] Risk of unfair, biased, or non-inclusive code suggestions
- [ ] Limited real-world impact of LLM-generated fixes
- [ ] Lack of accountability in LLM-generated fixes
- [ ] Transparency issues in LLM-generated bug fixes (e.g., declaration, documentation etc.)
- [ ] I don't have any concern
- [ ] Not Listed (Write multiple by going to the next line)

Introduction to LLM for fixing software bug

* 11. Since you have NEVER USED LLMs for any software development task, please click on the video (approximately 3 minutes) to watch a demonstration of how LLMs can assist developers in fixing software bugs.
Please select one of the following problem sets (Java or Python) and use an AI tool such as ChatGPT or Microsoft Copilot to seek assistance. After verifying, please copy and paste the response code obtained from it.

Developer Perceptions of LLM Use, Verification, Liability, Credit, and Reoccurrence Risk

In this section, we invite you to share your personal views on using LLMs like ChatGPT for bug fixing, especially around trust, verification, and accountability. Your responses will help us understand how developers perceive liability, assign credit, and evaluate risk when working with AI-generated fixes.

The following definitions are provided for your understanding to answer questions in this section.

- **Low-Risk Software Applications:**
  Applications where bugs do not directly impact safety, finances, or sensitive data.
  Example: A content management system for public blogs.
- **High-Risk Software Applications:**
  Applications where bugs can lead to significant harm, including financial loss, health risks, or data breaches.
  Example: A medical diagnosis platform or online banking system.
- **Liability:**
  The degree of responsibility attributed to a person or AI system when an AI-assisted bug fix results in failure or causes new issues.
- **Credit:**
  The share of recognition or success attributed to either the developer or the AI tool when a bug is fixed effectively and the system performs as intended.
- **Bug Reoccurrence:**
  The event where a software bug, previously believed to be resolved, appears again in the same or similar form, often indicating an incomplete or flawed fix.
- **Code Syntactic Assistance:**
  LLMs help by fixing syntax errors or formatting issues.
  Example: Auto-correcting a missing semicolon or indentation.
- **Code Semantic Assistance:**
  LLMs help generate or modify logic based on code understanding.
  Example: Suggesting a corrected algorithm for calculating tax.
- **Contextual Suggestions:**
  LLMs provide code suggestions based on your overall project or previously written functions.
  Example: Completing a helper function based on naming and surrounding code.
- **Documentation Lookup:**
  LLMs retrieve or generate explanations or usage examples for APIs or libraries.
  Example: Providing usage syntax for a Python requests library function.
- **Code Debugging Assistance:**
  LLMs identify and help fix logical or runtime errors.
  Example: Explaining why a function returns None and suggesting a correction.

* 12. In a **LOW-RISK** software application, how likely are you to use LLMs for the following types of support during bug fixing?

	Very Likely	Likely	Moderate	Unlikely	Rarely
Code Syntactic Assistance	○	○	○	○	○
Code Semantic Assistance	○	○	○	○	○
Contextual Suggestions	○	○	○	○	○
Documentation Lookup	○	○	○	○	○
Code Debug Assistance	○	○	○	○	○

* 13. In a **LOW-RISK** software application, how much verification effort (on a scale from 1 to 10) would you assign when receiving the following types of support from an LLM during bug fixing?

(1 = No verification required, 10 = Full manual verification required)

	1	2	3	4	5	6	7	8	9	10
Code Syntactic Assistance	○	○	○	○	○	○	○	○	○	○
Code Semantic Assistance	○	○	○	○	○	○	○	○	○	○
Contextual Suggestions	○	○	○	○	○	○	○	○	○	○
Documentation Lookup	○	○	○	○	○	○	○	○	○	○
Code Debug Assistance	○	○	○	○	○	○	○	○	○	○

* 14. In a **LOW-RISK** software application, imagine a scenario where a bug is fixed using LLM. For each type of support provided by the LLM, please indicate: (1) the level of credit you would assign to the LLM for the bug fix and (2) the level of liability you would assign to the LLM if the bug were to reoccur.
(For each row, select one option under each column.)

	Credit	Liability
Code Syntactic Assistance		
Code Semantic Assistance		
Contextual Suggestions		
Documentation Lookup		
Code Debug Assistance		

* 15. In a **HIGH-RISK** software application, how likely are you to use LLMs for the following types of support during bug fixing?

	Very Likely	Likely	Moderate	Unlikely	Rarely
Code Syntactic Assistance	○	○	○	○	○
Code Semantic Assistance	○	○	○	○	○
Contextual Suggestions	○	○	○	○	○
Documentation Lookup	○	○	○	○	○
Code Debug Assistance	○	○	○	○	○

* 16. In a **HIGH-RISK** software application, how much verification effort (on a scale from 1 to 10) would you assign when receiving the following types of support from an LLM during bug fixing?

(1 = No verification required, 10 = Full manual verification required)

	1	2	3	4	5	6	7	8	9	10
Code Syntactic Assistance	○	○	○	○	○	○	○	○	○	○
Code Semantic Assistance	○	○	○	○	○	○	○	○	○	○
Contextual Suggestions	○	○	○	○	○	○	○	○	○	○
Documentation Lookup	○	○	○	○	○	○	○	○	○	○
Code Debug Assistance	○	○	○	○	○	○	○	○	○	○

* 17. In a **HIGH-RISK** software application, imagine a scenario where a bug is fixed using LLM. For each type of support provided by the LLM, please indicate: (1) the level of credit you would assign to the LLM for the bug fix and (2) the level of liability you would assign to the LLM if the bug were to reoccur.
(For each row, select one option under each column.)

	Credit	Liability
Code Syntactic Assistance		
Code Semantic Assistance		
Contextual Suggestions		
Documentation Lookup		
Code Debug Assistance		

LLM Usage Policy

In this section, we invite you to share your thoughts on what an ideal LLM usage policy should include for software development, especially in bug fixing.

* 18. Please rank all the options when you would recommend using an LLM to help fix a software bug.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] For general code/program suggestions
- [ ] For software documentation
- [ ] For test case generation
- [ ] For software testing
- [ ] For software reviewing
- [ ] When organizational policies and risk guidelines are followed

* 19. Please rank all the options when you would restrict the use of an LLM to help fix a software bug.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] Security-critical modules (e.g., authentication, encryption)
- [ ] Proprietary or patented method (e.g., internal fraud detection algorithm, recommendation engine)
- [ ] Compliance-sensitive areas (e.g., legal, finance)
- [ ] Ethical decision-making systems (e.g., AI-driven hiring tools)
- [ ] User privacy handling modules (e.g., consent, data collection)
- [ ] High-availability or safety-critical systems (e.g., medical, aviation)

* 20. Please rank all the security policies that an organization should enforce when using LLM for bug fixing.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] Comply with data protection regulations (e.g., GDPR, HIPAA, ISO/IEC 27001)
- [ ] Restrict LLM access to confidential or proprietary business data
- [ ] Encrypt or anonymize data before sending to LLM
- [ ] Use on-premises LLMs for sensitive projects instead of cloud-based services.
- [ ] Organizational approval before using third-party AI tools
- [ ] Limit copy-paste from AI to avoid licensing/legal issues

* 21. Please rank all the validation processes LLM-generated code should go through.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] Manual review by developers
- [ ] Automated security scans and vulnerability testing
- [ ] Extended testing for high-risk applications
- [ ] Tag LLM-generated code for reviews
- [ ] Require dual validation: one human reviewer + one automated test pass
- [ ] Route AI-generated fixes to a separate branch for independent review and testing before merge

* 22. Please rank all the measures LLM-generated bug fixes should undergo to ensure accountability.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] Assign credit to the LLM for code contributions
- [ ] Assign liability if LLM fixes cause damage or failure
- [ ] Assign responsibility to a human reviewer for reviewing LLM fixes
- [ ] Document who approved or deployed the LLM-generated fix
- [ ] Log AI tool usage metadata (prompt, model, timestamp) in version control
- [ ] Require sign-off from responsible person before merging LLM-generated fixes

* 23. Please rank all the measures LLM-generated bug fixes should undergo to ensure transparency.
Here, 1 = most effective and 6 = least effective. You can also drag each option to adjust its ranking.

- [ ] Maintain logs of LLM-generated code
- [ ] Define the types of help taken from LLM
- [ ] Require LLM to provide reasoning behind their suggested fixes.
- [ ] Tag reviewer approval
- [ ] In-code labeling to flag AI-generated logic/code
- [ ] Mandatory LLM disclosure note in commit messages

LLM Usage Disclosure Agreement

In this section, we invite you to share your thoughts on how AI/LLM usage should be disclosed within your team or organization.

* 24. Please rank all the following topics that are essential to disclose regarding LLM system usage.
Here, 1 = most effective and 8 = least effective. You can also drag each option to adjust its ranking.

- [ ] **Purpose & Scope** – What the LLM is used for and its limitations
- [ ] **Code Contribution** – What parts were AI-generated, including liability and credit
- [ ] **Data Handling** – Specify how data is collected, stored, and processed
- [ ] **Model explanation** – Description of how the AI model produced the fix (e.g., logic, reasoning, or influencing factors)
- [ ] **Ethical Standards** – Steps taken to ensure fairness and avoid discrimination
- [ ] **Governance and Accountability** – Define oversight roles and responsibilities for LLM management
- [ ] **Justification or intention** – Why the AI tool was used
- [ ] **Validation level** – How thoroughly the AI-generated fix was reviewed or tested

* 25. Please rank all the channels through which 'Disclosures of LLM Systems' should be communicated.
Here, 1 = most effective and 8 = least effective. You can also drag each option to adjust its ranking.

- [ ] **Official Reports** – Internal reporting documents
- [ ] **Version Control** – Commit messages, diffs, or pull request summaries
- [ ] **User Documentation** – Manuals, FAQs, or product help sections
- [ ] **Developer Dashboards** – Logs or summaries tracking AI-generated code by developer each session
- [ ] **Stakeholder Meetings** – Presentations to partners, users, or clients
- [ ] **Regulatory Filings** – Reports submitted for compliance/legal needs
- [ ] **Method/ Class Level Code Documentation** – Before each method/class, document types of help taken from LLM, liability%, credit%, and risk level
- [ ] **In-line Code Comments** – Mark specific lines where LLM contributed (e.g., // generated by Copilot)

* 26. Please select the access level to the LLM usage disclosure documents.

	Full access	Limited access	No access
Peer Developers	○	○	○
Team Leads	○	○	○
Project Supervisor	○	○	○
QA Engineers	○	○	○
Legal/Compliance	○	○	○
Stakeholder	○	○	○

* 27. Please select all the potential benefits of disclosing information about LLM systems.

- ☐ **Enhanced Trust** – Increases user confidence and transparency
- ☐ **Meets Regulations** – Ensures legal and policy compliance
- ☐ **Improved Accountability** – Makes roles and decisions transparent
- ☐ **Stakeholder Engagement** – Encourages user and expert input
- ☐ **Enables Traceability** – Makes AI-generated code easier to track and review
- ☐ **Supports Fair Use** – Reduces risks of misuse, bias, or hidden automation
- ☐ None
- ☐ Not Listed (Write multiple by going to the next line)

* 28. Please select all the potential drawbacks of disclosing information about LLM systems.

- [ ] **Intellectual Property Risks** – May reveal proprietary or competitive information
- [ ] **Security Concerns** – Disclosures might reveal system vulnerabilities
- [ ] **Resource Intensive** – High effort needed to maintain documentation
- [ ] **Misinterpretation Risks** – Non-experts may misinterpret technical details
- [ ] **Misuse of Info** – Disclosed details could be exploited
- [ ] **Reputational Impact** – May raise doubts about code quality or team competence
- [ ] None
- [ ] Not Listed (Write multiple by going to the next line)

* 29. What additional rules, safeguards, or practices do you recommend to be included in an organizational policy to ensure responsible and trustworthy use of LLMs in software bug fixing?