Q1: Explain how AI-driven code generation tools (e.g., GitHub Copilot) reduce development time. What are their limitations?
AI tools like **GitHub Copilot** and **Tabnine** help developers by autocompleting code, generating boilerplate structures, and suggesting function logic. This significantly reduces development time, keeps programmers in flow, and accelerates prototyping. They also improve productivity by surfacing commonly used code patterns.

Limitations include:

- **Context Awareness**: Limited understanding of complex, cross-file logic.
- **Accuracy**: Potential to introduce incorrect or insecure code.
- **Over-reliance**: May reduce deep learning or expertise in junior developers.

Q2: Compare supervised and unsupervised learning in the context of automated bug detection.

- **Supervised learning** uses labeled data—code samples tagged as “buggy” or “clean”—to train models that detect similar patterns. It delivers high accuracy when trained on extensive, quality datasets but requires costly labeled data.
- **Unsupervised learning** identifies anomalies or outliers in code without labels, useful for discovering unknown bug types. It excels in early-stage bug exploration but may produce false positives and requires substantial fine-tuning for relevance.

Q3: Why is bias mitigation critical when using AI for user experience personalization?
Bias mitigation is crucial because AI models can unintentionally reinforce existing stereotypes or discriminate against underrepresented users. In personalization systems, this could lead to unfair content recommendations, limited access to opportunities, or exclusion. By integrating fairness frameworks, developers can ensure inclusive experiences, uphold ethical standards, and build trust in their AI systems.

---

2. Case Study Analysis

📈 *AI in DevOps: Automating Deployment Pipelines*

How does AIOps improve software deployment efficiency? Provide two examples.

AIOps (Artificial Intelligence for IT Operations) enhances deployment pipelines by automating decision-making, improving system reliability, and reducing human intervention in the DevOps lifecycle. Below are two specific examples from the Azati article:

1. Intelligent CI/CD Automation
   Azati's AI-integrated CI/CD pipelines use historical deployment data to make smart decisions, like identifying high-risk releases and triggering canary deployments. This reduces manual oversight and speeds up safe, consistent software delivery.
2. Self-Healing Infrastructure
   AIOps tools embedded in the deployment pipeline monitor live system metrics such as CPU usage, memory consumption, and error logs. If an anomaly—like a sudden spike in memory—is detected, the system can automatically take corrective actions, such as:
   - Restarting a service
   - Scaling up containers or virtual machines
   - Redirecting traffic

These self-healing capabilities reduce downtime, prevent cascading failures, and ensure high availability without requiring immediate human intervention.