Q1: How do AI-driven code generation tools reduce development time? What are their limitations?

AI-driven code generation tools like GitHub Copilot reduce development time by:

- **Auto-completing code snippets** based on context, reducing boilerplate coding.
- **Suggesting functions or logic** based on natural language comments or partial code.
- **Accelerating prototyping** by generating scaffolding for new features or modules.
- **Reducing context switching**, allowing developers to stay focused within the IDE.

Limitations include:

- **Lack of deep understanding** of business logic or project-specific constraints.
- **Potential for insecure or inefficient code**, especially if suggestions are blindly accepted may reproduce insecure or copyrighted code.
- **Over-reliance** may hinder learning for junior developers.
- **Difficulty with complex logic**, where human reasoning and domain knowledge are essential.

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

Aspect	Supervised Learning	Unsupervised Learning
Definition	Learns from labeled data (e.g., "bug" vs "no bug")	Learns patterns from unlabeled data
Use in Bug Detection	Classifies code as buggy or clean based on historical labeled examples	Detects anomalies or unusual patterns in code that may indicate bugs
Strengths	High accuracy if labeled data is available; good for known bug types	Useful for discovering unknown or rare bugs; no need for labeled data
Limitations	Requires large, high-quality labeled datasets	May produce false positives or hard-to-interpret results

Q3: Why is bias mitigation critical when using AI for user experience personalization?

Bias mitigation is essential because:

- **Personalization algorithms trained on biased data** can reinforce stereotypes or exclude certain user groups.
- **Unfair treatment** (e.g., showing fewer opportunities or features to certain demographics) can erode trust and violate ethical or legal standards.
- **Inclusive personalization** ensures that all users receive equitable, relevant, and respectful experiences.

- **Transparency and fairness** in AI-driven UX fosters user satisfaction and long-term engagement.

2. Case Study Analysis

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

How AIOps Enhances Deployment Efficiency

1. **Automated Anomaly Detection**
    - AIOps tools continuously monitor logs, metrics, and events across the deployment pipeline.
    - They use machine learning to detect anomalies—such as unexpected spikes in resource usage or failed builds—**before they escalate into outages**, enabling faster resolution and reduced downtime.
2. **Predictive Failure Analysis**
    - By analyzing historical deployment data, AIOps can **predict potential points of failure** in the pipeline.
    - For example, if a specific microservice often fails under certain conditions, AIOps can flag it in advance, allowing teams to proactively fix issues and avoid rollback scenarios.