PART 1

- Q1. Explain how AI-driven code generation tools (e.g., GitHub Copilot) reduce development time. What are their limitations?
- ANS

AI-driven code generation tools like GitHub Copilot reduce development time by automating repetitive tasks, suggesting code snippets, and completing lines of code in real time. They leverage large language models trained on vast code repositories to provide context-aware recommendations, minimizing manual typing and debugging. These tools accelerate prototyping, reduce syntax errors, and help developers discover best practices or unfamiliar APIs faster. However, limitations include potential code quality issues, over-reliance on generated code without understanding, and security vulnerabilities from unvetted suggestions. They may also struggle with complex logic, lack domain-specific knowledge, or produce outdated or non-optimized solutions. Additionally, legal and licensing concerns arise when generating code resembling proprietary sources. Proper review and testing remain essential.

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

Supervised learning for automated bug detection relies on labeled datasets (e.g., buggy/non-buggy code) to train models that classify or predict defects, offering high accuracy but requiring extensive annotated data. Unsupervised learning detects anomalies or unusual patterns in code without labeled examples, making it useful when labeled data is scarce but potentially yielding higher false positives. Supervised approaches excel in identifying known bug types, while unsupervised methods can uncover novel or rare defects. However, supervised models may struggle with unseen bug patterns, whereas unsupervised techniques require additional validation to distinguish true bugs from benign outliers. The choice depends on data availability and the need to detect known versus unknown defects.

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

ANS

Bias mitigation is critical in AI-driven user experience personalization because biased algorithms can reinforce stereotypes, exclude underrepresented groups, or deliver unfair recommendations. If training data reflects historical prejudices, the AI may favor certain demographics, leading to discriminatory outcomes (e.g., skewed job ads or loan offers).

Unchecked bias reduces inclusivity, alienates users, and harms brand trust. It can also create feedback loops, where biased recommendations further skew future data. Legal and ethical risks arise if personalized experiences violate fairness regulations (e.g., GDPR or anti-discrimination laws). Mitigation strategies—like diverse datasets, fairness-aware algorithms, and continuous bias audits—ensure equitable, relevant experiences for all users while maintaining compliance and fostering long-term engagement.

CASE STUDY ANALYSIS

AIOps (Artificial Intelligence for IT Operations) enhances software deployment efficiency by automating and optimizing DevOps processes using AI and machine learning. It reduces manual intervention, accelerates error detection, and improves decision-making through predictive analytics and real-time data processing.

Example 1: Automated Anomaly Detection
AIOps tools analyze logs, metrics, and traces to detect anomalies early—such as performance bottlenecks or deployment failures—before they escalate. For instance, an AI model can predict a server overload during deployment and automatically scale resources, preventing downtime.

Example 2: Intelligent Root Cause Analysis
Instead of manual troubleshooting, AIOps correlates incidents across systems to pinpoint root causes. If a deployment fails due to a dependency conflict, AIOps identifies the exact issue and suggests fixes, reducing mean time to resolution (MTTR).

By streamlining monitoring, incident response, and resource management, AIOps ensures faster, more reliable deployments while minimizing human error.