Part 1: Theoretical Analysis (30%)

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

AI-driven code generation tools such as GitHub Copilot accelerate software development by suggesting code snippets, functions, and even complete algorithms based on context. These tools are trained on massive datasets of public code repositories, allowing them to predict the developer's intent and generate syntactically correct and relevant code. They reduce development time by automating repetitive coding tasks, providing intelligent suggestions that allow developers to focus more on logic and architecture, and enhancing learning for beginners through real-time examples. However, they have limitations: they can generate inaccurate or insecure code because they do not fully understand project context; there are ethical and copyright concerns when generated code resembles proprietary data; and they can lead to overreliance, where developers accept suggestions without deep understanding.

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

In supervised learning, models are trained on labeled data—datasets where inputs and their corresponding outputs (e.g., "bug" or "no bug") are known. In automated bug detection, supervised models can learn from historical bug reports and source code to predict whether new code changes are likely to introduce defects. In unsupervised learning, models work with unlabeled data, identifying hidden patterns or clusters without explicit outcomes. In bug detection, unsupervised learning can identify anomalies in software metrics, log files, or runtime behavior that deviate from normal patterns, indicating potential bugs. In summary, supervised learning offers higher accuracy but requires labeled data, while unsupervised learning is valuable for early anomaly detection when labels are unavailable. Together, they enhance automated bug detection systems.

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

Bias mitigation ensures that AI-driven personalization systems treat all users fairly and inclusively. Personalization relies on user data, which may reflect societal or historical biases. If left unchecked, these biases can lead to discriminatory or unequal experiences across demographic groups. For example, an AI system might favor recommendations for one user group while neglecting others. Such bias can harm reputation and violate ethical or legal standards. Bias mitigation promotes fairness, builds user trust, and ensures compliance with frameworks like GDPR. Methods such as dataset balancing, bias detection metrics, and tools like IBM AI Fairness 360 help maintain equity and transparency in AI-driven personalization systems.

Case Study: AI in DevOps – Automating Deployment Pipelines

AIOps (Artificial Intelligence for IT Operations) enhances deployment efficiency by applying machine learning and analytics to automate and optimize DevOps workflows. It minimizes human intervention, detects issues early, and accelerates release cycles. Two examples include: 1. Intelligent Log Analysis & Anomaly Detection: AIOps tools continuously monitor logs and performance data, automatically identifying anomalies or failures during deployment. They can trigger rollbacks or alerts, minimizing downtime. 2. Automated Root Cause Analysis: Platforms like Dynatrace and Splunk AI use pattern recognition to pinpoint the source of deployment issues. Instead of manually reviewing logs, engineers receive instant insights, enabling faster recovery and smoother operations. In essence, AIOps transforms deployment pipelines into self-learning,

self-healing systems that enhance reliability and speed.