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 leverage large language models trained on vast code datasets to suggest functions, lines of code, and documentation in real-time. They significantly reduce development time by automating repetitive coding tasks, improving syntax accuracy, and minimizing the need for manual lookups. These tools accelerate prototyping and testing, allowing developers to focus on logic and design rather than boilerplate code.

However, their limitations include potential generation of insecure or incorrect code, lack of understanding of project-specific context, and possible license or copyright risks. They may also lead to overreliance on automation, reducing developers' problem-solving and debugging skills.

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

Supervised learning involves training a model on labeled datasets, such as past bug reports marked as 'bug' or 'no bug'. This enables the model to learn patterns associated with defective code segments. For example, a logistic regression model can classify new code snippets based on learned characteristics.

Unsupervised learning, in contrast, uses unlabeled data to detect hidden patterns or clusters. It is particularly effective for discovering new or unknown types of bugs by grouping unusual code behaviors or identifying anomalies in system performance.

Comparison:
• Supervised learning → Predicts known bug patterns.
• Unsupervised learning → Detects anomalies and new issues.
Both are complementary in improving automated bug detection systems.

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

Bias mitigation is essential in AI-driven user experience personalization because models learn from user interaction data, which may underrepresent certain demographics. Without correction, the system might deliver unequal or exclusionary experiences, favoring majority user groups. This could reduce inclusivity and user satisfaction.

By applying bias mitigation, developers ensure fairness, regulatory compliance, and trust. It allows systems to provide balanced recommendations that reflect diversity in user behavior and preferences.

Case Study: AIOps in DevOps - Automating Deployment Pipelines

AIOps enhances software deployment efficiency by applying artificial intelligence to automate monitoring, incident response, and performance optimization in DevOps pipelines.

Example 1: Automated anomaly detection – AI algorithms monitor logs and metrics to identify irregular patterns (e.g., sudden CPU spikes) before failures occur.
Example 2: Predictive scaling – AI predicts workload surges and automatically provisions additional resources to maintain performance stability.

Through automation and prediction, AIOps reduces manual interventions, minimizes downtime, and accelerates reliable software releases.