Theme: Building Intelligent Software Solutions

Part 1: Theoretical Analysis (30%)

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

Answer: GitHub Copilot reduces development time by suggesting context-aware code snippets, automating repetitive tasks, and assisting with boilerplate code. This accelerates coding, debugging, and prototyping. However, limitations include: over-reliance on AI-generated code, potential security flaws, lack of deep understanding of project context, and possible intellectual property issues as it may generate code similar to public repositories.

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

Answer: Supervised learning detects bugs using labeled datasets—code tagged as buggy or clean—and trains models to classify new code. It's accurate but requires annotated data. Unsupervised learning identifies anomalies or patterns in unlabeled code data. It's useful for uncovering unknown bug patterns but can be less precise and generate more false positives.

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

Answer: Bias mitigation ensures personalized experiences don't unfairly favor or exclude users based on sensitive attributes like race or gender. Personalization algorithms trained on biased data can perpetuate stereotypes or marginalize users. Mitigating bias promotes fairness, inclusivity, and trust, while reducing legal and reputational risks.

Case Study: AI in DevOps: Automating Deployment Pipelines

Answer: AIOps optimizes software deployment by using AI to predict, detect, and respond to system anomalies. It automates routine tasks, shortens incident response times, and reduces human intervention.

Examples:

1. Predictive Scaling: AI anticipates traffic surges and scales services proactively.

2. Failure Prediction: AI detects code changes likely to cause failures and halts deployment.