🧠 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:
AI-driven code generation tools like **GitHub Copilot** analyze existing repositories and natural language prompts to suggest relevant code snippets in real-time.
They **reduce development time** by:

- Auto-completing repetitive code patterns (loops, conditionals, functions).
- Speeding up boilerplate generation (e.g., CRUD operations).
- Assisting in unfamiliar APIs or frameworks with context-aware suggestions.
- Minimizing syntax errors through auto-correction.

Limitations include:

- **Lack of context awareness:** Copilot may not fully understand project logic or architecture.
- **Security risks:** It can suggest vulnerable or outdated code.
- **Over-reliance:** Developers may accept AI suggestions without understanding them.
- **Intellectual property issues:** Generated code can unintentionally reproduce licensed snippets.

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

Aspect	Supervised Learning	Unsupervised Learning
Definition	Learns from labeled data (bug/non-bug).	Learns from patterns in unlabeled data.
Application	Classifies code segments as buggy or clean using historical labeled bug reports.	Detects anomalies or unusual patterns that may indicate bugs.
Example	Random Forest model trained on bug labels in GitHub Issues.	Clustering algorithms (e.g., K-means) that detect unusual code metrics.
Strength	High accuracy when quality labeled data exists.	Useful when labeled data is scarce.
Limitation	Requires extensive labeled datasets.	May produce false positives due to ambiguous clusters.

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

Bias mitigation ensures that personalization algorithms treat users **fairly**, avoiding stereotypes or exclusion.
Without mitigation:

- **Certain groups** may receive less relevant recommendations.
- **Discriminatory outputs** could harm user trust.
- **Feedback loops** reinforce bias (e.g., favoring one demographic).

Mitigation techniques:

- Diverse and representative training datasets.
- Using fairness libraries (e.g., IBM AI Fairness 360).
- Regular auditing for biased behavior.

Personalization should balance accuracy with **ethical fairness** to enhance inclusivity

Case Study: AI in DevOps — Automating Deployment Pipelines

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

Answer:
AIOps integrates **machine learning and analytics** into DevOps to automate decision-making in CI/CD pipelines.
It improves efficiency by:

1. **Automated anomaly detection:** ML models monitor deployment logs and alert teams before failures occur.
2. **Intelligent root cause analysis:** AI clusters log data to identify failure origins, reducing mean time to repair (MTTR).

Example 1: Predicting deployment rollbacks based on historical performance metrics.
Example 2: Auto-scaling infrastructure during high-load deployment windows.