THEORY

Question One

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

AI-driven code generation tools like GitHub Copilot accelerate development by predicting and suggesting code snippets in real-time as developers' type. This reduces the need to write repetitive code manually, speeds up syntax recall, and minimizes time spent searching documentation. These tools also enhance productivity by allowing developers to focus on logic and structure instead of boilerplate code.

Limitations of AI-driven code generation tools

AI-driven code generation tools sometimes generate incorrect or insecure code, especially when context is unclear. They also lack deep understanding of business logic or unique project needs. Over-reliance can reduce a developer's ability to think critically or understand underlying logic. Additionally, they can unintentionally produce biased or plagiarized code if not properly guided.

Question Two

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

In supervised learning, models are trained on labelled data where each input (code snippet or log) is tagged as "buggy" or "not buggy." This helps the AI learn patterns to predict future bugs in similar contexts. It is effective but requires a large, labelled dataset, which is often time-consuming to create.

In unsupervised learning, the model explores unlabelled data to find anomalies or unusual patterns that might indicate bugs. This approach is useful in unknown or evolving systems where labelled data is unavailable. However, it might raise false alarms since not all anomalies are bugs.

In summary, supervised learning offers accuracy when labels are available, while unsupervised learning offers flexibility in new or unlabelled environments.

Question Three

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

Bias mitigation is crucial because AI models learn from historical user data, which can contain hidden societal, gender, racial, or cultural biases. If unaddressed, the AI may reinforce these patterns, leading to discriminatory or exclusionary user experiences.

For example, a biased AI might personalize a tech product mainly for male users if it was trained on male-dominated usage data, ignoring female or minority user needs. Bias mitigation ensures that AI-driven personalization is inclusive, fair, and respects diversity, leading to a better and more ethical user experience for all users.

Case Study Analysis on AI in DevOps

AIOps (Artificial Intelligence for IT Operations) enhances software deployment efficiency by integrating machine learning and big data to automate and optimize operational processes in the software lifecycle. In DevOps environments, where speed, reliability, and accuracy are crucial, AIOps significantly reduces human errors, detects anomalies early, and supports faster releases through intelligent automation.

One major way AIOps improves deployment is through intelligent anomaly detection**.** Traditional monitoring tools require manual setting of thresholds and often generate noise with too many alerts. AIOps solutions, however, analyse patterns in real-time and learn the normal behaviour of systems. For instance, if a sudden CPU spike occurs during deployment, AIOps can instantly recognize it as an anomaly and either auto-resolve it or notify the appropriate team with relevant context. This greatly reduces downtime and ensures deployment stability.

Secondly, AIOps boosts efficiency through predictive analytics and automation of routine deployment tasks. It can identify deployment trends and anticipate potential failures before they occur. For example, if a particular microservice

consistently fails during updates, AIOps can flag it based on historical deployment data and recommend a rollback or an alternative strategy. Tools like Moog soft and Splunk AIOps use such capabilities to reduce Mean Time to Resolution (MTTR) by up to 40–60%.

In conclusion, AIOps transforms software deployment from reactive to proactive. By combining automation, intelligent analysis, and self-healing capabilities, it ensures smoother, faster, and more reliable software deliveries—key ingredients for modern DevOps success.