**Part 1: Theory**

**Q1: How do 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 accelerate development by offering real-time code suggestions, autocompleting functions, and generating boilerplate code. This reduces manual effort, speeds up prototyping, and minimizes context-switching. They help developers write cleaner, faster code especially for repetitive or standardized tasks.

However, their limitations include potential inaccuracies, lack of context awareness for complex or project-specific logic, and the risk of generating insecure or non-optimized code. Over-reliance can reduce critical thinking and understanding of the code. Furthermore, they may unintentionally reproduce biased or copyrighted code patterns from their training data.

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

**Answer:**  
In automated bug detection, **supervised learning** relies on labelled data—code samples marked as buggy or clean. This allows models to learn known patterns of bugs and make accurate predictions, but it requires large, high-quality labelled datasets.

**Unsupervised learning**, on the other hand, identifies anomalies without labelled data. It’s useful for detecting unknown or emerging bugs by clustering similar code and flagging outliers. While it can reveal hidden issues, it’s prone to false positives and lacks precision without ground truth labels.

Combining both approaches can improve robustness: supervised for known bugs, unsupervised for novel issues.

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

**Answer:**  
Bias mitigation is essential in AI-driven personalization to ensure fairness, inclusivity, and ethical decision-making. If an AI model is trained on biased data (e.g., favouring one demographic or region), it can reinforce stereotypes, exclude certain users, or deliver unequal experiences—especially in recommendation systems or adaptive interfaces.

This not only harms user trust and satisfaction but can also lead to legal and reputational consequences. Mitigation techniques like data balancing, fairness-aware algorithms, and continuous auditing help ensure equitable outcomes and personalized experiences that serve all users fairly.

**Q4: How does AIOps improve software deployment efficiency? Provide two examples.**

**Answer:**  
AIOps (Artificial Intelligence for IT Operations) enhances software deployment by using machine learning to automate, monitor, and optimize DevOps workflows. It reduces downtime, identifies issues early, and streamlines CI/CD pipelines.

**Example 1:** *Anomaly detection* – AIOps tools can detect irregularities in deployment logs or system behaviour in real-time, preventing failures before they affect users.

**Example 2:** *Automated root cause analysis* – AIOps can correlate events across systems to quickly pinpoint and resolve deployment errors, reducing time-to-resolution and minimizing human intervention.

Overall, AIOps leads to faster, more reliable, and scalable software releases.

**Part 2: Testing**

### ****150-Word Summary: AI-Enhanced Login Testing****

Using Testim.io, I automated test cases for a sample login page involving both valid and invalid credentials. The tool’s AI-assisted recorder accurately detected input fields, buttons, and validation messages without requiring manual selectors. This drastically reduced setup time and minimized the chance of errors during element selection. AI also helped auto-suggest the most reliable locators, ensuring the tests remained stable even if minor UI changes occurred.

Unlike manual testing—which is repetitive and error-prone—AI-powered testing provided instant feedback, visual validation, and reusable test steps. I could quickly duplicate and edit tests to check various credential combinations, increasing test coverage with minimal effort. Testim also provided structured reports with pass/fail outcomes and screenshots, simplifying result interpretation.

Overall, AI enhanced both speed and accuracy while enabling scalable test design. It allowed me to simulate real user behaviour and reliably test different login scenarios that would be tedious and time-consuming to do manually.

**Part 3: Accuracy**

Accuracy: 0.9941520467836257

F1 Score: 0.9941694320031468

Classification Report:

precision recall f1-score support

High 0.97 1.00 0.99 34

Low 1.00 1.00 1.00 78

Medium 1.00 0.98 0.99 59

accuracy 0.99 171

macro avg 0.99 0.99 0.99 171

weighted avg 0.99 0.99 0.99 171

**Part 4: Ethical considerations**

In **Task 3: Predictive Analytics for Resource Allocation,** we used the **Kaggle Breast Cancer Dataset** to simulate and train a model that predicts **issue priority (high/medium/low)**. After preprocessing the data (cleaning, labeling, and splitting), we trained a **Random Forest classifier** and evaluated its performance using **accuracy and F1-score**. While technically effective, deploying such a predictive model in a real company setting introduces ethical challenges—especially around bias.

The dataset, although medical, was repurposed to model organizational priorities. If applied in actual resource allocation—such as determining which employee requests or departmental issues deserve urgent attention—biases may arise from the dataset itself. For instance, if certain teams (e.g., urban-based, senior-level, or male-dominated) are overrepresented in the training data, the model could unfairly assign higher priority to their issues, systematically deprioritizing underrepresented groups.

To address such risks, fairness tools like **IBM AI Fairness 360** can be integrated. These tools detect biases in the data and apply corrective techniques (e.g., reweighting, adversarial debiasing) to improve equity in model outcomes. By using these tools, we ensure AI systems promote fair and transparent decision-making, fostering organizational trust and ethical responsibility in predictive analytics.