**Part 1 :**

**Theoretical Analysis**

**Short Answer questions**

Q1: ***How AI-driven code generation tools reduce development time and their limitations***

**Benefits**

\* Faster Code Writing: Tools like GitHub Copilot autocomplete boilerplate code, suggest common patterns, and reduce the need for repetitive manual coding.

\*Improved Productivity: Developers can focus on higher-level logic while the AI assists with syntax and implementation details.

\*Fewer Context Switches: Integrated directly into IDEs, these tools allow developers to stay in flow without frequently switching to documentation or Stack Overflow.

\*Learning Aid: Junior developers can learn by observing suggestions and understanding best practices embedded in the model’s training data.

**Limitations**

\*Context Awareness: AI tools lack deep understanding of project-specific requirements and may suggest code that doesn’t align with architecture or intent.

\*Security & Bugs: Generated code can contain vulnerabilities or logical flaws, especially when suggestions are accepted without scrutiny.

\*Dependence Risk: Over-reliance may stifle skill development or critical thinking in developers.

\*Licensing & IP Concerns: There are ongoing legal questions about whether generated code violates software licenses from the training data.

***Q2: Supervised vs. Unsupervised Learning in Automated Bug Detection***

**Feature Supervised Learning Unsupervised Learning**

Definition Trained on labeled data (buggy vs. clean) Finds patterns or anomalies without labels

Usage in Bug Detection Classifies code as buggy or not based Detects anomalies that

on past labeled examples may indicate bugs

Accuracy Higher when sufficient labeled data is available Can work when labeled data is scarce

Examples Training a model on bug reports and code fixes Using clustering to flag outlier code behavior

Limitation Requires large, high-quality labeled datasets May generate false positives (not all anomalies are bugs)

**Conclusion:** Supervised learning excels with sufficient labeled data, while unsupervised is useful in exploratory or novel bug discovery without relying on past labels.

***Q3: Why Bias Mitigation Is Critical in AI for User Experience Personalization***

Reasons bias mitigation is essential:

\*Fairness: Without addressing bias, AI may offer unequal experiences based on gender, race, or location, disadvantaging certain user groups.

\*User Trust: Biased personalization erodes trust. For instance, offering different content quality or pricing to users based on assumptions can seem discriminatory.

\*Regulatory Compliance: Laws like the EU’s AI Act or GDPR require transparency and fairness in automated decision-making.

\*Business Impact: Biased systems may alienate users and damage brand reputation, especially when biased outputs go viral or face scrutiny.

\*Model Generalization: Bias reduces the system's ability to generalize across diverse users, harming overall experience quality.

Examples:

-A job portal that recommends fewer leadership roles to women due to historical data patterns.

-A news app prioritizing echo chambers for certain users, reinforcing stereotypes.

-Mitigating bias ensures inclusivity, improves model robustness, and promotes ethical AI usage.