Part 1: Theoretical Analysis (30%)

1. Short Answer Questions

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

How they reduce development time (Efficiency):

- Real-time Suggestions
- Boilerplate & Repetition
- Focus Shift

Limitations:

- Code Quality and Bugs
- Context Depth
- Security and Licensing

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

Feature	**Supervised Learning**	**Unsupervised Learning**
Training Data	**Labeled Data:** Code segments are explicitly tagged as either **"buggy" (defective)** or **"non-buggy" (clean)**.	**Unlabeled Data:** The model is given code without any prior bug/no-bug labels.
Goal for Bug Detection	**Classification/Prediction:** To learn the mapping from code features to the known labels, allowing it to **predict** if new code will be buggy based on past examples.	**Anomaly/Outlier Detection:** To discover the structure of "normal" (non-buggy) code and **flag anything that deviates significantly** from this established norm.
Strengths	**High Accuracy for Known Bugs:** Very effective at detecting **recurrent, well-defined bug patterns** present in the training data (e.g., null pointer exceptions).	**Detects Novel Bugs:** Excellent for flagging **new, unknown, or zero-day bug types** because it focuses on deviations, not just learned labels.

Weaknesses	**Expensive to Train:** Requires significant **manual effort and expertise** to create large, accurately labeled datasets. **Poor at detecting novel bug types** not represented in the training set.	**Higher False Positive Rate:** Anything statistically rare, even unique but correct code, may be flagged as an anomaly, requiring more **human triage**.

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

- Because unchecked AI bias in user experience (UX) personalization can lead to unfair, discriminatory, and non-inclusive outcomes.