📘 AI in Software Engineering – Week 4 Assignment

Student: Longoli Joseph Eyanae
Part 1: Theoretical Analysis

1. Short Answer Questions

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

AI-driven code generation tools like **GitHub Copilot** and **Tabnine** are transforming the way developers write code. These tools leverage **large language models (LLMs)** trained on massive codebases to understand context and offer **real-time code suggestions**, completions, or even full-function implementations based on natural language prompts.

🟢 **Benefits:**

- **Faster development**: Reduces time spent on boilerplate and repetitive coding tasks.
- **Boosts productivity**: Developers focus on solving logic problems rather than syntax.
- **Context-aware suggestions**: Improves efficiency during code reviews and pair programming.

🔴 **Limitations:**

- **Inaccuracy**: Sometimes the code generated is logically incorrect or insecure.
- **Lack of understanding**: Copilot doesn't understand project-specific constraints.
- **Bias and copyright issues**: It may generate code snippets from licensed or biased sources.
- **Over-reliance**: Developers may stop understanding the code deeply if overly dependent.

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

Supervised Learning and **Unsupervised Learning** are both valuable in automating software bug detection, but they approach the problem differently.

🔵 **Supervised Learning:**

- Uses **labeled datasets**, where each code snippet or issue is tagged as "bug" or "no bug".

- Algorithms like **Logistic Regression** or **Random Forest** can be trained to classify code or logs.
- **Example**: A model trained on bug-labeled GitHub issues can predict if a new issue is a bug or enhancement.

🔴 **Unsupervised Learning:**

- Works with **unlabeled data** to detect patterns or anomalies.
- Useful when there's no historical labeling available.
- Techniques like **clustering** or **anomaly detection** can identify outliers in system logs or error rates.
- **Example**: Detecting a spike in log frequency that doesn't match normal usage — indicating a hidden bug.

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

Bias mitigation is essential in AI-powered user experience (UX) personalization to ensure **fairness, inclusivity, and ethical integrity**. AI models often learn from user data—which can reflect societal biases related to **gender, race, language, or region**. If unchecked, these models may deliver skewed recommendations or **exclude certain user groups**.

2. Case Study Analysis

- **Read the article**: *AI in DevOps: Automating Deployment Pipelines.*
- **Answer: How does AIOps improve software deployment efficiency? Provide two examples.**

AIOps (Artificial Intelligence for IT Operations**)** enhances deployment pipelines by **automating monitoring, predicting failures**, and **self-healing systems**. It combines big data, machine learning, and automation to improve software delivery speed and reliability.

🔹 **How AIOps Improves Efficiency:**

1. **Automated Root Cause Analysis**:
   - AI scans logs, metrics, and traces to identify where deployment failed.
   - This saves time vs. manually sifting through logs.
2. **Predictive Scaling**:
   - Monitors user traffic trends and auto-scales infrastructure before bottlenecks occur.

    - Reduces downtime and enhances user experience.

- **Real-World Examples:**

1. **Netflix** uses AIOps for proactive monitoring — automatically rerouting traffic when deployment issues arise.
2. **Spotify** applies AIOps for continuous integration — flagging risky releases and pausing deployments based on anomaly predictions.