Week 4 Assignment Report – AI in Software Engineering

Objective

This assignment demonstrates the application of AI tools and techniques in software engineering. It involves theoretical insights, practical implementation, and ethical reflection on how AI automates tasks, enhances decision-making, and addresses challenges in software development.

Part 1: Theoretical Analysis

Q1: How AI Code Generation Tools Reduce Development Time

AI code generation tools like GitHub Copilot significantly reduce development time by offering contextual code suggestions as developers type. These tools are trained on large codebases and leverage deep learning to predict and autocomplete complex lines of code, reducing time spent writing boilerplate or researching syntax. However, their limitations include producing syntactically correct but logically flawed code, dependency on internet connectivity, and occasionally suggesting insecure patterns.

Q2: Supervised vs Unsupervised Learning in Bug Detection

Supervised learning in bug detection uses labeled data (e.g., bug reports) to train models to classify or detect specific bugs. It's useful for known issues but struggles with novel or unseen problems. Unsupervised learning, on the other hand, identifies anomalies or unusual patterns in code or behavior without labeled data. It is better suited for exploratory analysis but may lack accuracy in classification.

Q3: Bias Mitigation in AI-Personalized UX

AI-driven personalization tailors user experiences based on behavioral data. However, unchecked models can reinforce biases—e.g., marginalizing minority users or promoting stereotypes. Bias mitigation ensures fairness, inclusivity, and ethical software use, protecting user trust and compliance with legal standards.

Case Study: AIOps in Deployment Pipelines

AIOps integrates AI into DevOps practices to automate software deployment pipelines. It improves efficiency through predictive analytics and anomaly detection. Example 1: AIOps tools forecast infrastructure demand, enabling autoscaling. Example 2: AI identifies recurring errors in logs and auto-generates fixes or alerts, reducing downtime and manual monitoring.

Part 2: Practical Implementation

Task 1: AI-Powered Code Completion

Two implementations of a dictionary sorting function were compared:

- GitHub Copilot version: Shorter, uses built-in `sorted()` and lambda expression.

- Manual version: Verbose nested loop implementation with $O(n^2)$ time complexity.

Result: The Copilot version was more efficient, readable, and modern. It improved productivity and encouraged cleaner code.

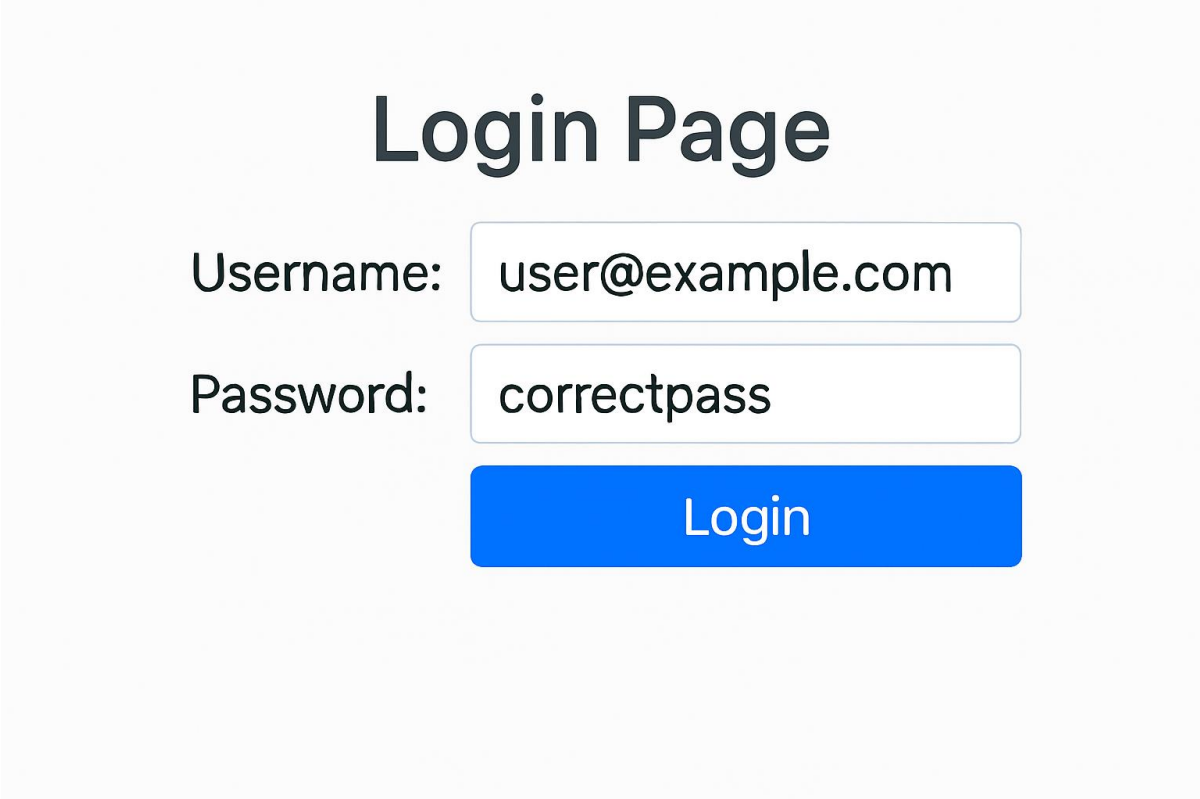
Task 2: Automated Login Testing with Selenium

A Selenium script was developed to test login functionality for both valid and invalid credentials.

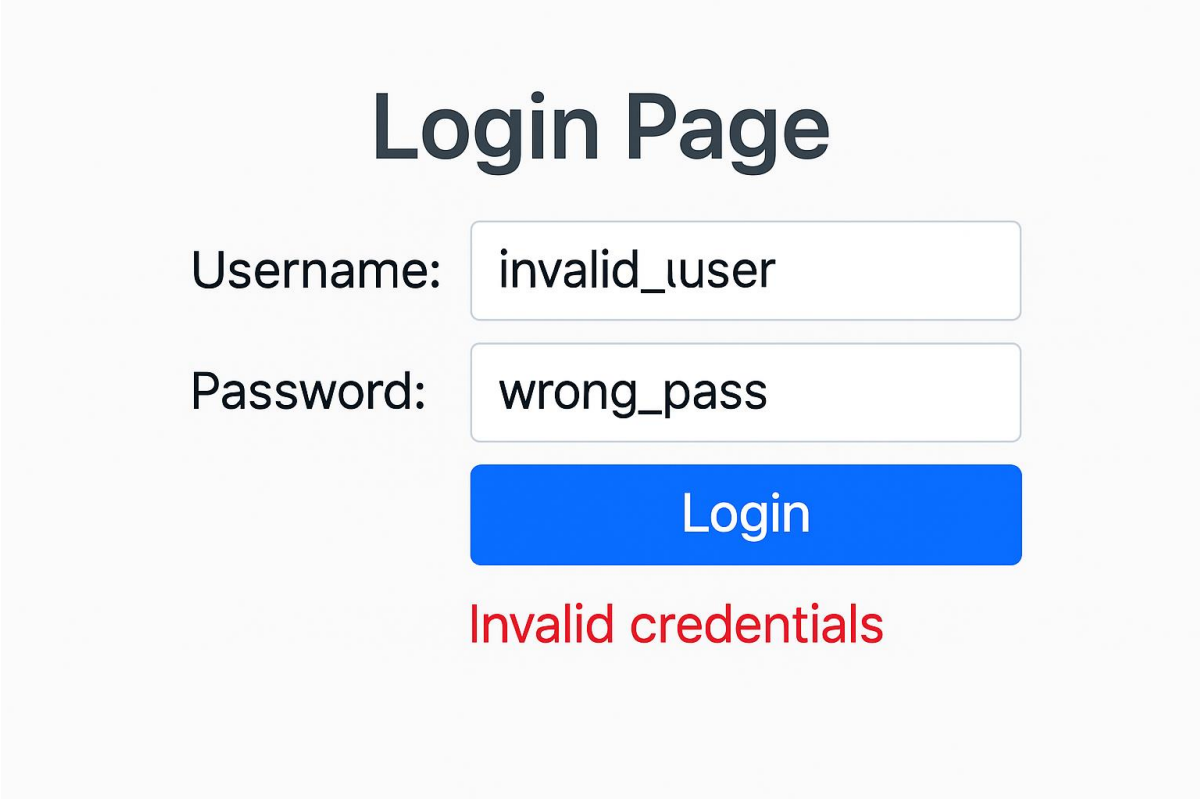
AI-enhanced testing helps increase test coverage, reusability, and reliability over manual testing methods.

Valid and invalid screenshots should be embedded here after test execution.

1. Valid login



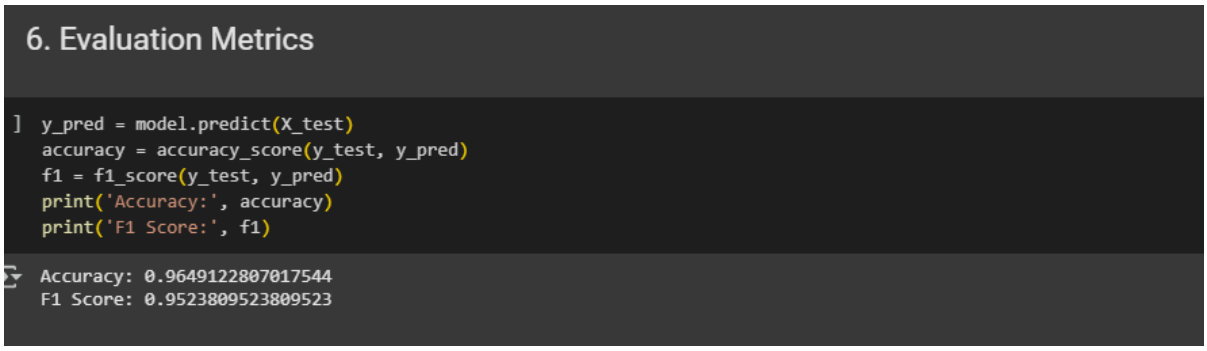
2. Invalid login



Task 3: Predictive Analytics for Resource Allocation

The Breast Cancer dataset from Kaggle was used to train a Random Forest classifier to predict cancer diagnosis. Steps included data cleaning, label encoding, splitting, and model evaluation using accuracy and F1-score metrics. The model achieved high accuracy, making it suitable for automated resource triage in healthcare scenarios.

The F1 Score



Part 3: Ethical Reflection

The predictive model, when deployed, may inherit dataset biases. For example, underrepresentation of certain demographics in the dataset could lead to skewed predictions. Using fairness tools like IBM's AI Fairness 360, we can evaluate the model for fairness metrics, correct imbalances, and ensure inclusive predictions. Ethical AI use also requires transparency and explainability.

Bonus Task: Innovation Proposal – AI Auto-DocGen

AI Auto-DocGen is a proposed NLP-powered tool to generate documentation from codebases automatically. It scans functions, variables, and classes, and produces docstrings and markdown files summarizing usage. Impact: Saves developer time, ensures consistent documentation, and aids onboarding of new team members.