AI for Software Engineering Report

Part 1: Theoretical Analysis

1. Short Answer Questions

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

- ✓ AI tools like GitHub Copilot accelerate development by generating boilerplate code, suggesting context-aware completions, and reducing repetitive tasks.It helps streamline workflows, especially in test writing, API integration, and syntax-heavy tasks.
- ✓ Limitations include:
  - (i) Code quality concerns**:** AI may suggest insecure or inefficient code.
  - (ii) Context blindness: Copilot lacks full project awareness, leading to mismatched suggestions.
  - (iii)Over-reliance risk: Developers may accept flawed code without review, reducing critical thinking.

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

- ✓ *Supervised learning* uses labeled data (e.g., known bugs and fixes) to train models that classify or predict bug types. It's effective for bug triage, severity prediction, and fix assignment, especially when historical data is available.
- ✓ *Unsupervised learning* identifies patterns in unlabeled data, useful for anomaly detection, clustering similar issues, or discovering unknown bug types. It's ideal when labeled datasets are scarce.

Comparison:

Feature	**Supervised Learning**	**Unsupervised Learning**
Data Requirement	Labeled	Unlabeled
Use Case	Classification, prediction	Clustering, anomaly detection
Accuracy	Higher (with good data)	Exploratory, less precise
Example	Predicting bug severity	Grouping similar error logs

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

- ✓ Bias in personalization algorithms can lead to exclusion, stereotyping, or unfair treatment of users. For example, underrepresented groups may receive irrelevant or harmful recommendations.
- ✓ Mitigation is critical because**:**
  - (i) It ensures fairness and inclusivity in digital experiences.
  - (ii) It protects against algorithmic discrimination.
  - (iii)It builds trust in AI systems.
- ✓ Tools like IBM AI Fairness 360 help detect and correct bias by analyzing datasets and model outputs for disparities

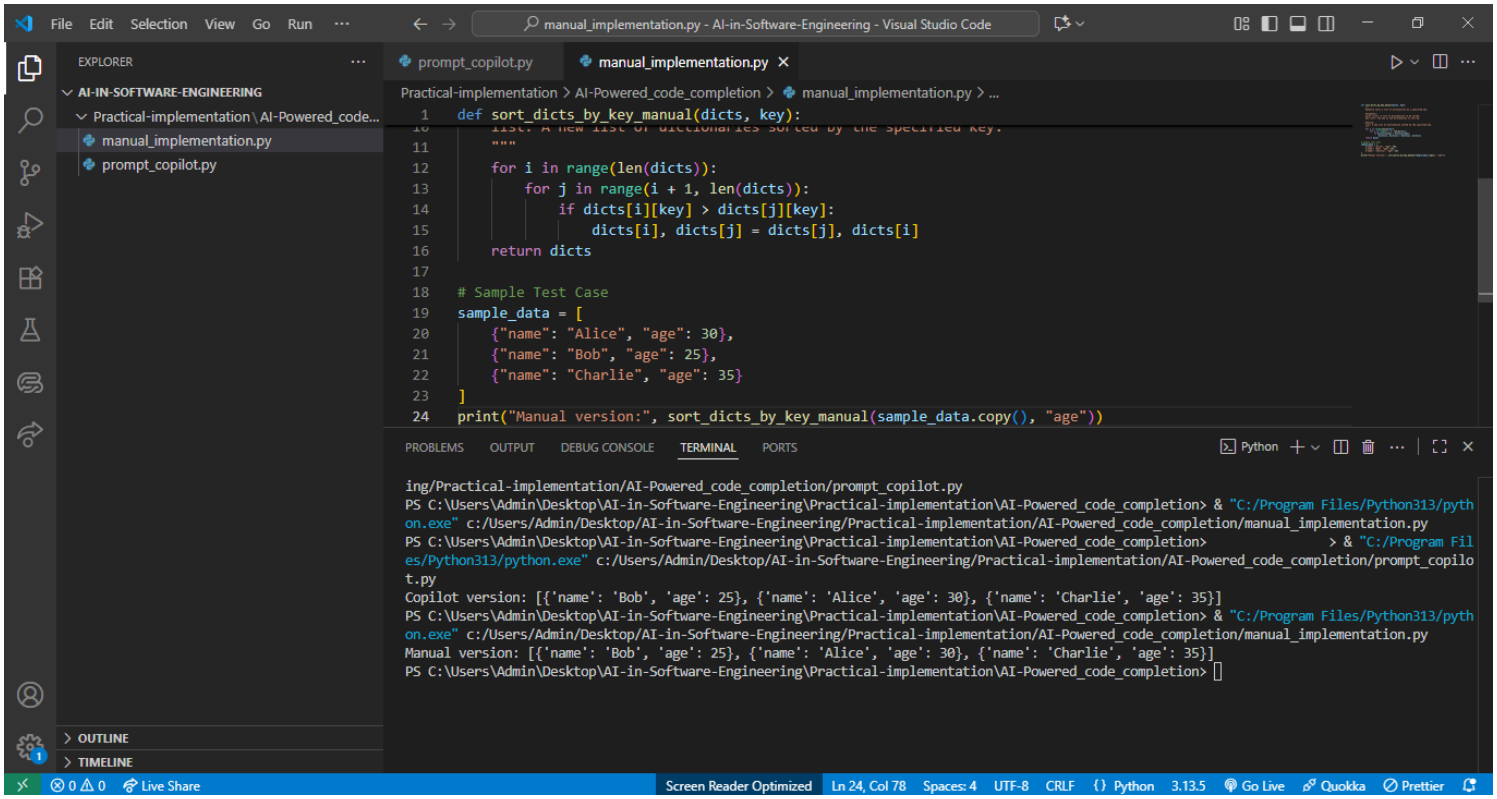
2. Case Study Analysis

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

- ✓ AIOps enhances software deployment efficiency by automating decision-making, reducing manual intervention, and enabling predictive responses across the CI/CD pipeline. It leverages historical data, machine learning, and real-time analytics to streamline operations and minimize downtime.
- ✓ Examples include:
  - (i) Predictive rollbacks and optimized CI/CD workflows - Tools like Harness and CircleCI use AI to analyze historical test data and predict build failures before deployment. Harness automatically rolls back failed deployments, reducing human effort and downtime. CircleCI prioritizes test cases based on success/failure rates, accelerating feedback loops, and improving developer productivity.
  - (ii) AI-driven monitoring and incident response - Platforms like New Relic and Datadog apply AI to detect anomalies in logs and metrics before they impact users. AI-powered chatbots assist DevOps engineers by recommending solutions based on past incidents, enabling faster resolution and reducing system outages.

Part 2: Practical Implementation

Task 1: AI-Powered Code Completion



GitHub Copilot's function uses Python's built-in sorted() method with a lambda expression, making it concise, readable, and efficient. It leverages optimized internal sorting algorithms with O(n log n) complexity, suitable for large datasets. The autogenerated docstring adds clarity, demonstrating Copilot's ability to produce well-documented code with minimal input.

The manual version uses a nested loop structure, resembling bubble sort. While it correctly sorts the list, it's less efficient ($O(n^2)$) and harder to maintain. This approach is useful for understanding sorting logic but not ideal for production environments.

Copilot's suggestion was context-aware and required no edits. It saved development time and aligned with Pythonic best practices. However, it assumes the key exists in all dictionaries and doesn't handle exceptions or edge cases like missing keys or mixed types. This highlights a limitation of AI-generated code: it often lacks defensive programming.

Overall, Copilot accelerated the coding process and provided a clean solution. The manual version offered deeper control but at the cost of performance and clarity. In real-world scenarios, Copilot's output is preferable for speed and maintainability, provided developers review and test its suggestions to ensure robustness and correctness.

Task 2: Automated Testing with AI

Test Scripts

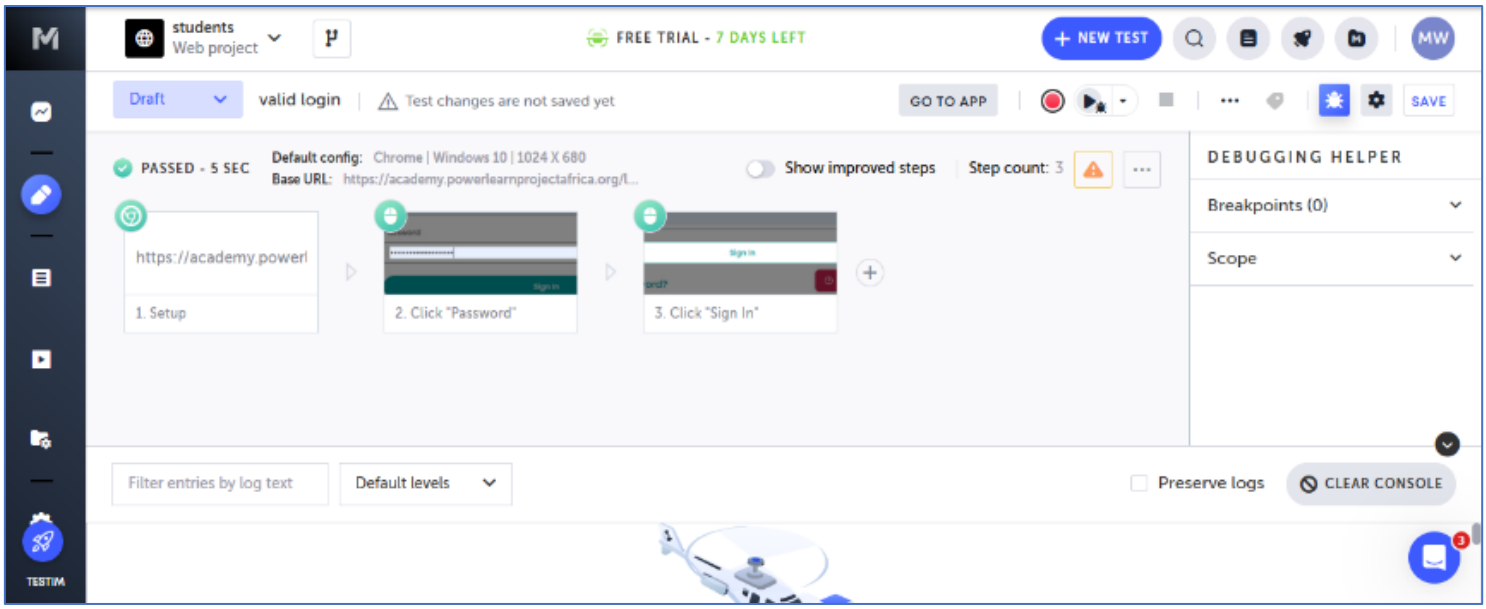
- ✓ *Tool used:* Testim.io
- ✓ *Test cases:*

  (i) Valid login with correct credentials
  (ii) Invalid login with incorrect credentials

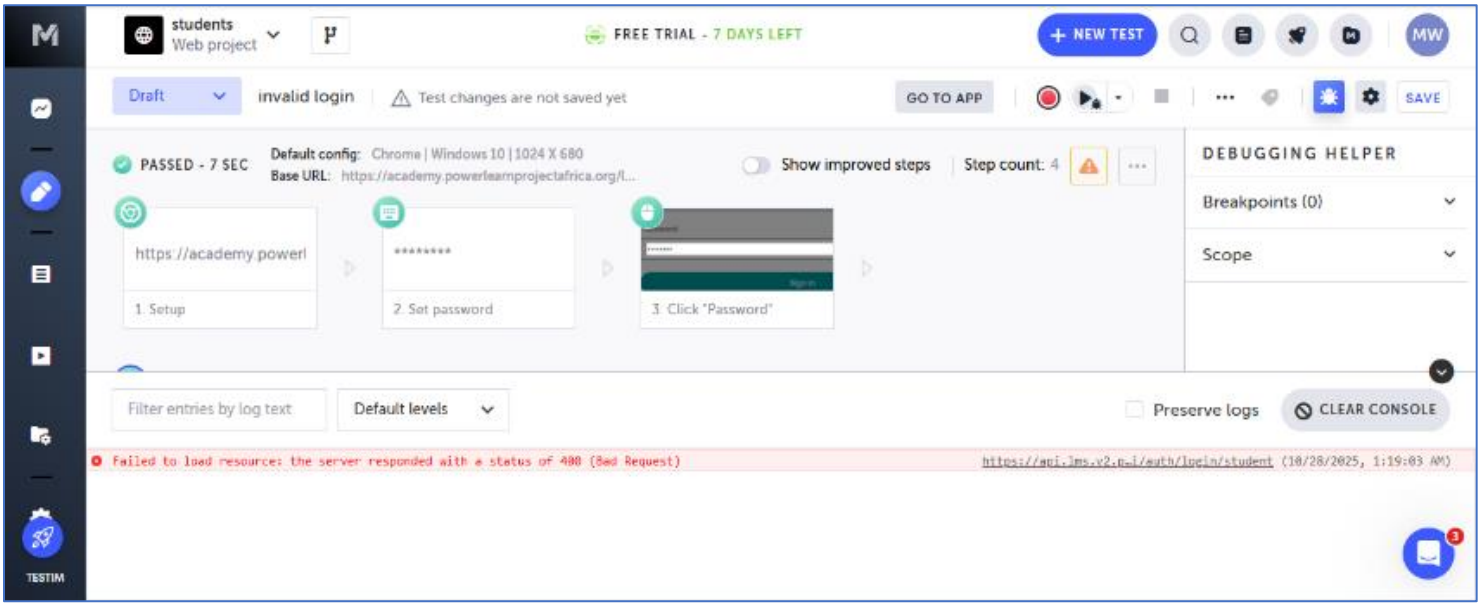
- ✓ *Base URL*: https://academy.powerlearnprojectafrica.org/login

Screenshots

- ✓ Screenshot of **valid login test** (passed)



- ✓ Screenshot of **invalid login test** (passed with error handling)



Summary

- ✓ AI-powered testing tools like Testim.io significantly enhance test coverage and efficiency compared to manual testing. In this task, I automated login validation for both valid and invalid credentials using Testim's visual editor and AI-assisted step suggestions. The platform automatically identified UI elements, optimized locators, and grouped reusable components, reducing setup time and improving test reliability.
- ✓ Unlike manual testing, which is prone to human error and requires repetitive effort, AI tools streamline workflows and adapt to UI changes. The ability to run tests across multiple configurations (e.g., browsers, resolutions) with minimal effort ensures broader coverage. Additionally, Testim's dashboard provides instant feedback on pass/fail rates, helping teams iterate faster.

Task 3: Predictive Analytics for Resource Allocation

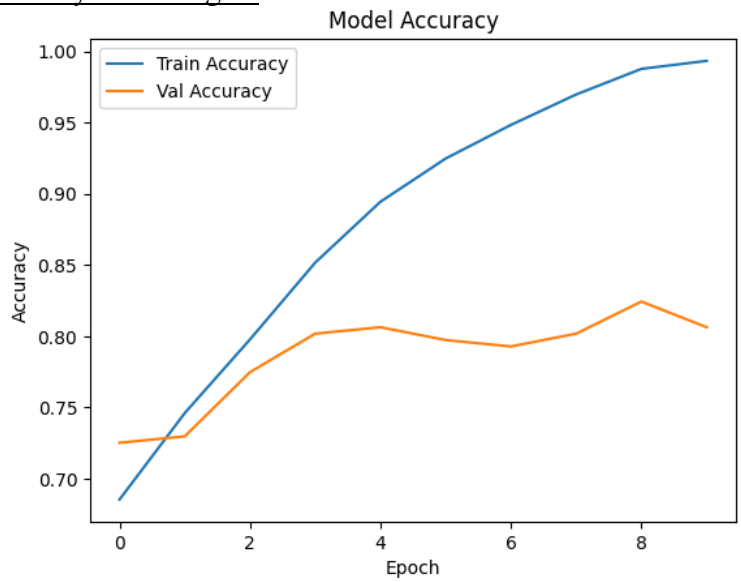
Model Architecture

- ✓ *Type:* CNN with 2 convolutional layers, dropout, and dense layers
- ✓ *Dataset:* IUSS 23–24 Breast Cancer Diagnosis (PNG images)
- ✓ *Training samples:* 890 images (benign/malignant)
- ✓ *Validation samples:* 222 images
- ✓ *Test samples:* Unlabeled images used for inference
- ✓ *Total parameters*: 7.39 million
- ✓ *Input shape:* 128×128×3 (RGB images)

Training Results

- ✓ *Training accuracy:* Increased steadily to 99.6% by epoch 10
- ✓ *Validation accuracy:* Peaked at 82.4% (epoch 9), but fluctuated between 72–82%
- ✓ *Validation loss:* Increased after epoch 5, indicating overfitting

Accuracy Plot Insights



- The **blue line** (Train Accuracy) shows near-perfect learning
- The **orange line** (Val Accuracy) plateaus and dips slightly, confirming that the model is memorizing training data more than generalizing

Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	0.72	0.83	0.77	158
Malignant	0.31	0.19	0.23	64
Overall accuracy	_	_	0.64	222

- ✓ *Weighted F1-Score:* 0.61
- ✓ *Macro F1-Score:* 0.50
- ✓ *Issue Priority Mapping:* Malignant = High, Benign = Medium

Interpretation

- ✓ The model is highly confident on clear cases but struggles with ambiguous or minority class (malignant).
- ✓ Recall for malignant is low (0.19), which is ethically concerning in medical contexts. False negatives could delay treatment.
- ✓ Overfitting is evident after epoch 5; consider early stopping or regularization in future iterations.

Part 3: Ethical Reflection

Bias and fairness in deployed predictive models

- ✓ When deploying a predictive model like the CNN trained in Task 3, it's critical to examine potential biases in the dataset. In this case, the breast cancer image dataset may suffer from class imbalance, with more benign than malignant samples, leading to a model that underperforms on malignant cases. This was evident in the low recall (0.19) for malignant predictions, which could result in false negatives and delayed treatment in real-world scenarios.
- ✓ Bias can also emerge from underrepresented subgroups in the image data, for example, if images disproportionately represent certain demographics (e.g., age, ethnicity, imaging equipment), the model may generalize poorly across diverse populations.
- ✓ Fairness tools like IBM AI Fairness 360 can help by:
  - (i) Auditing model performance across subgroups
  - (ii) Quantifying bias using metrics like disparate impact or equal opportunity
  - (iii)Applying bias mitigation techniques (e.g., reweighting, adversarial debiasing)