Release Summary

Product Name: Code Performance Analyzer

Team Member	**Role**
Philip Pesic	Backend Developer
Adwaith Madadi	Backend Developer
Rohit Mandal	Backend Developer
Michael Pimentel	Frontend Developer
Juan Alvarez Sanchez	Frontend Developer

User Story 1.1 (Backend)

- **As a user, I want to run a simple model that can analyze Python code through a local script so that I can get an initial Big-O complexity estimate.**
- **Acceptance Criteria: Running complexity inference with a small code snippet successfully returns a time complexity**.

User Story 1.2 (Frontend)

- **As a user, I want to see a mocked complexity annotation inside the editor UI so I can preview how the extension will eventually present feedback.**
- **Acceptance Criteria: The extension environment is capable of displaying a non-selectable annotation next to a highlighted chunk of code**

User Story 2.1 (Backend)

- **As a user, I want the extension to return a relatively accurate time-complexity estimate so that the extension is useful.**
- **Acceptance Criteria: The model scores at least 90% on an accuracy comparison with the teacher.**

User Story 2.2 (Frontend)

- **As a user, I want the extension to clearly display complexity in an editor annotation, terminal, and sidebar.**
- **Acceptance Criteria: The frontend shows both an editor annotation and extra relevant info in the terminal, and the sidebar successfully displays with space for the test suite and performance chart.**

User Story 3.1 (Backend)

- **As a user, I want to be able to generate a test script to profile for real performance metrics like execution time and memory usage.**
- **Acceptance Criteria: Generating a test through the extension generates a test file, which successfully returns time and space metrics of the input function.**

User Story 3.2 (Frontend)

- **As a user, I want to submit code through the extension and have the locally hosted model instantly return real complexity results, so the extension becomes functional.**
- **Acceptance Criteria: Upon spinning the model, the "Analyze Code Complexity" becomes functional and displays the real complexity.**

User Story 4.1 (Backend)

- **As a user, I want to be able to use the model without having to download it and use my computer's resources.**
- **Acceptance Criteria: The Kubernetes cluster successfully deploys and scales, and both complexity analysis and test generation work as expected.**

User Story 4.2 (Frontend)

- **As a user, I want the interface to highlight inefficient patterns and show a visual performance chart so I can understand and improve my code's behavior over time.**
- **Acceptance Criteria: The performance chart displays performance results relevant to the actual output of the model in the sidebar.**

Known Bugs

- Running on ARM64 or without CUDA fallback causes PyTorch deadlock

- Starting Server or Port Forwarding sometimes fails on both dev container and cluster launch. Just requires restart.

- CSV Export sometimes provides zero for execution time

- Sometimes the extension falls back to local analysis even when the server is up