AI-Assisted Testing for Transport Management

Introduction:

The client for the project was constantly facing challenges with their legacy code that required frequent changes, and subsequently regression testing that was time-consuming. They sought an efficient solution by way of better unit testing. We reviewed their system using Java tools like JaCoCo for code coverage and JUnit for tests, providing a preliminary estimate.

The client requested a more precise estimate and faster delivery. After brainstorming with them we decided to leverage generative AI technology, specifically tools like OpenAI's Codex, Codiumate, and GPT-3. This approach promised to automate and accelerate the process, achieving thoroughness and accuracy in half the estimated time. Using JaCoCo, we identified gaps in test coverage, set up the testing framework with JUnit, and employed generative AI to write test cases. This reduced manual effort and improved test coverage, significantly enhancing efficiency and reducing turnaround times. Our client was pleased with the rapid delivery and improved testing precision.

Client Details:

Name: Confidential | **Location**: UK | **Industry**: Transportation

Technologies:

Java 8, Maven, JUnit 4, Mockito, PowerMockito, JaCoCo (Reporting)

AI-Assisted Testing for Transport Management

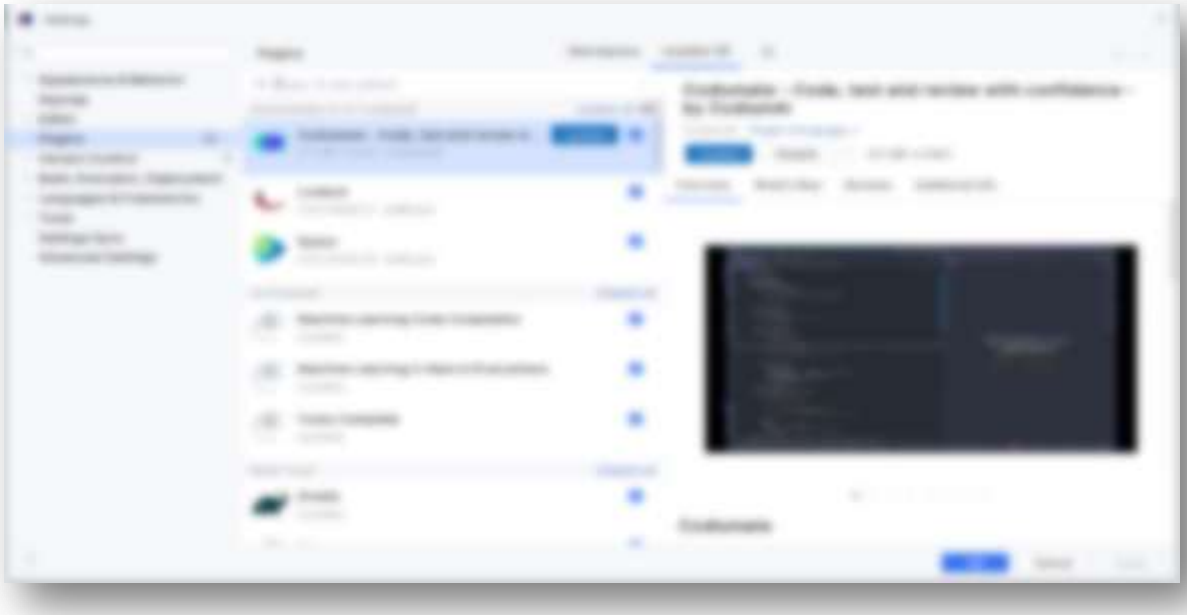
Project Description:

- **Project Scope**: The project is a comprehensive multi-module system, with each module containing between 3,000 and 39,000 lines of code. The primary goal was to enhance unit test coverage across all modules to ensure system reliability and maintainability.
- **Coverage Focus**: We concentrated on covering the Utility classes and Service layer classes, which are critical for the core functionality and performance of the system.
- **Initial State**: Initially, the unit test coverage for each project module was alarmingly low, ranging between just 1-2%. This demanded a significant need for improvement to ensure robust code quality.
- **Gap Identification**: Our team conducted a thorough analysis to identify gaps in the existing test coverage. We developed strategic plans to address these gaps, ensuring a more comprehensive and effective testing framework.
- **AI Tools Utilized**: To accelerate the process of creating unit tests, we leveraged multiple generative AI tools, including Codex, Diffblue Cover, and Codiumate. These tools were instrumental in automating and enhancing the test generation process.
- **Performance of Codiumate**: Among the tools used, Codiumate proved to be particularly effective. It was able to generate numerous test cases covering happy paths, edge cases, and various other scenarios, providing a solid foundation for thorough testing.
- **Coverage Achievements**: Codiumate significantly boosted code coverage, managing to cover 40-50% of complex code lines. When used in combination with GPT-3, the coverage increased substantially, reaching 80-90% for some modules.
- **JaCoCo Implementation**: We implemented JaCoCo to measure unit test coverage across all modules. The results were consolidated into a single module named code-coverage, facilitating easy tracking and analysis.
- **Plugin Versioning**: To maintain compatibility with the legacy code, we aligned the versions of various plugins in accordance. This step was essential to ensure the smooth and effective operation of the testing framework.
- **GitLab Pipeline**: To enhance transparency and ease of monitoring, we created a GitLab pipeline. This allowed the client to directly view the percentage of lines covered by unit tests, providing clear insights into the progress and effectiveness of our testing efforts.

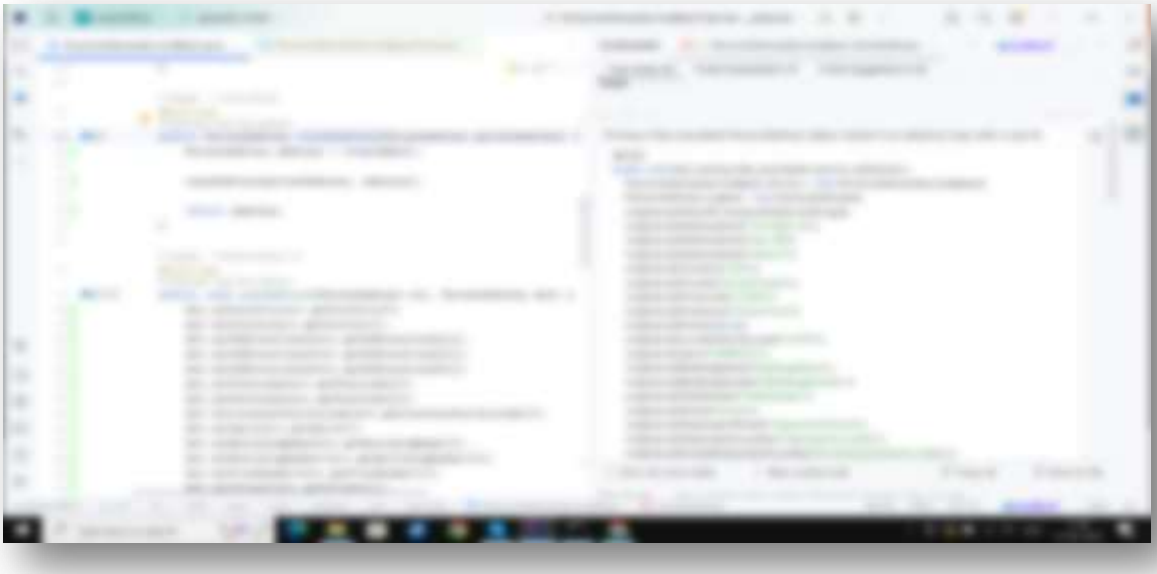
AI-Assisted Testing for Transport Management

Screenshots:

1) Codiumate Plugin



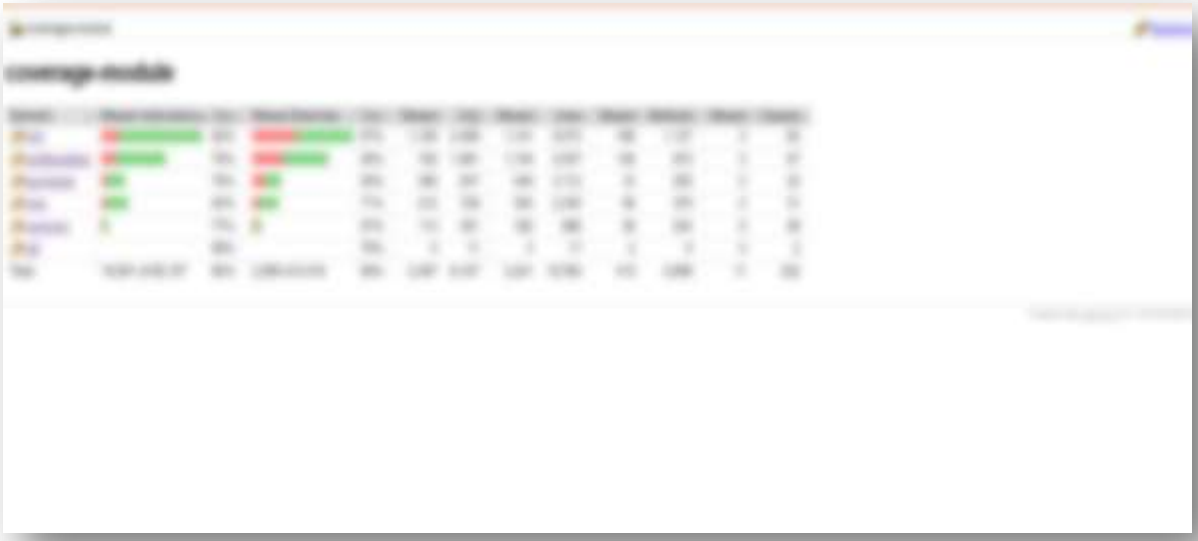
2) Code generation using Codiumate



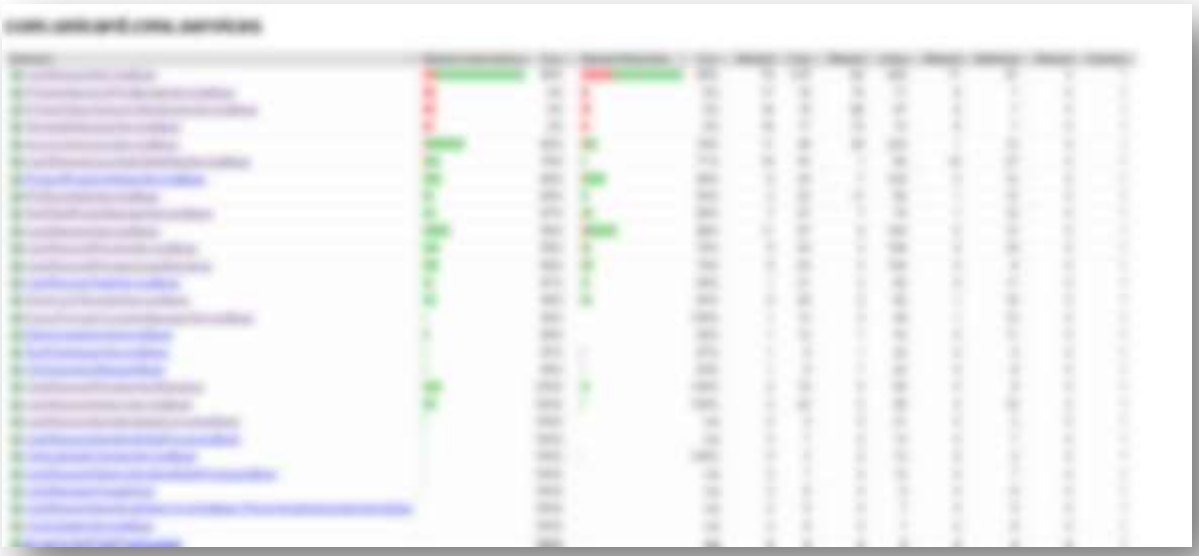
AI-Assisted Testing for Transport Management

3) ChatGPT Code Enhancement

4) JaCoCo Report



AI-Assisted Testing for Transport Management



5) GitLab Pipeline