

# SE Project-1 Bonus Report

The code for the bonus implementation can be found here:

[https://github.com/serc-courses/se-project-1--\\_14/tree/master/BONUS](https://github.com/serc-courses/se-project-1--_14/tree/master/BONUS)

---

## Description:

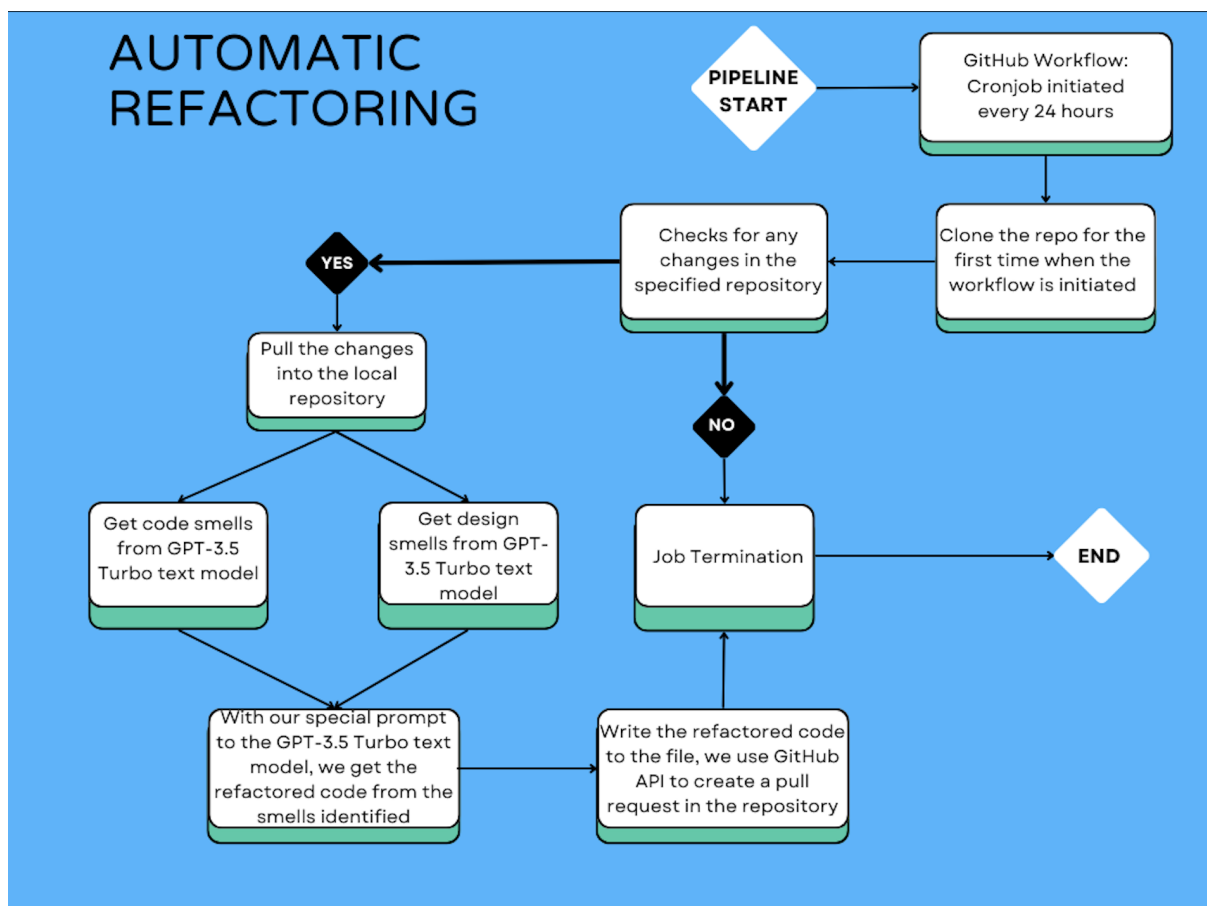
- We've implemented a **GitHub workflow that runs a CRON job every 24 hours**. The YML file for it can be found at `.github/workflows/design-smell-check.yml`
- Our current pipeline just uses the `scripts/scripts.py` as a monolithic codebase.
- The repository is cloned if not already done under which case it is pulled from GitHub.
- We checkout to the `Main` or `Master` branch.
- A new branch is created based on seconds passed since UTC.
- For every `.java` file in the repository,
  - We find the code smells using a curated prompt for GPT3.5 that has descriptions of code metrics.
  - We find the design smells using the code smells again using a curated prompt for the current code.
  - Finally, we get the refactored code using, *cue drumrolls*, a final prompt that returns only the code. We write this refactored code into original file.
- After all files have been refactored we create a pull request to the original repository using the GitHub API, completing our pipeline.

## Notes:

- One thing to note is that since we are using OpenAI library's `0.28.0 version`, we cannot use the codex models like **code-davinci-002** which are specially curated for an excellent code analysis and suggestions. The reason why we cannot use this model is because *OpenAI has deprecated these codex models :(*

- We are using the **Text GPT-3.5 Turbo model** for our analysis throughout. We were not expect this text model to output code alone with a generic prompt specifying the smells which we identified, but we wrote a very specific prompt which surprisingly achieves the same, so we were not required to do some text processing on the output to specifically extract the code snippet from it.
- Since there is only a limited usage of the API supported, we only consider the first five files to limit it and also limit the code length because of the token size limit like: `code = file.read()[:3000]`
- We also tried using CheckStyle to detect code smells to pass to GPT but were unable to integrate it into GitHub workflow. The sample code for it can be found in `scripts/get_code_smells.ipynb` under `utilities.run_checkstyle()`

## Flowchart to explain our approach:



Note that we create a branch first before doing a pull request, we specify the branch name via the timestamp since the UTC.

Active branches					
Branch		Updated	Check status	Behind   Ahead	Pull request
<a href="#">branch_1708710108</a>		3 minutes ago		0   1	#9  ...
<a href="#">branch_1708709504</a>		13 minutes ago		0   1	#8  ...
<a href="#">branch_1708709457</a>		14 minutes ago		0   1	#7  ...
<a href="#">branch_1708709366</a>		15 minutes ago		0   1	#6  ...
<a href="#">branch_1708709197</a>		18 minutes ago		0   1	#5  ...
<a href="#">View more branches &gt;</a>					

## Example output:

```
Repository already cloned. Pulling latest changes...
<git.repo.base.Repo '/home/vineeth/Desktop/III-II/SE/project/se-project-1--_14/BONUS/automated-refactoring-main/temp/.git'>
Repository cloned successfully
Branch 'branch_1708709504' created and checked out successfully!
Formed new branch branch_1708709504
Analyzing code in file: temp/books-core/src/main/java/com/sismics/util/jpa/DbOpenHelper.java
Code Smells Found: Overall, the code seems to be well-structured and follows some good practices. However, there are a few areas that could be improved:

1. **Cyclomatic Complexity**: The 'open()' method has a high cyclomatic complexity due to multiple try-catch blocks and nested logic. It could be refactored to reduce complexity and improve readability.
2. **Class Data Abstraction Coupling**: The 'DbOpenHelper' class has a dependency on concrete classes like 'SuppliedConnectionProviderConnectionHelper' and 'FormatStyle'. This could lead to higher coupling and reduced flexibility. Consider using interfaces or abstractions instead.
3. **Class Fan-Out Complexity**: The 'DbOpenHelper' class directly interacts with multiple classes such as 'ServiceRegistry', 'ConnectionHelper', 'SqlStatementLogger', etc. This could increase the complexity of the class and make it harder to maintain.
4. **Boolean Expression Complexity**: There are boolean expressions in the code that could be simplified or extracted into separate methods for better readability.
5. **JavaNCSS**: The code has a mix of Java and SQL statements within the 'open()' method, which could make it harder to maintain and test.
6. **NPath Complexity**: The 'open()' method has nested try-catch blocks and conditional statements that could lead to high NPath complexity. Consider refactoring these to simplify the logic.
7. **Error Handling**: The error handling in the code is not consistent. Some exceptions are caught and logged, while others are not properly handled.
8. **Resource Management**: Resources like 'Connection', 'Statement', and 'ResultSet' should be properly closed in a 'finally' block to avoid resource leaks.

Overall, the code could benefit from refactoring to improve readability, maintainability, and reduce complexity.

Code smells Analysis Complete
Design Smells Found: Based on the provided code snippet and the identified design smells, here are some specific observations and recommendations:

1. **Cyclomatic Complexity**:
   - The 'open()' method has nested try-catch blocks and multiple levels of logic, leading to high complexity. Consider refactoring this method to break down the logic into smaller, more manageable pieces.



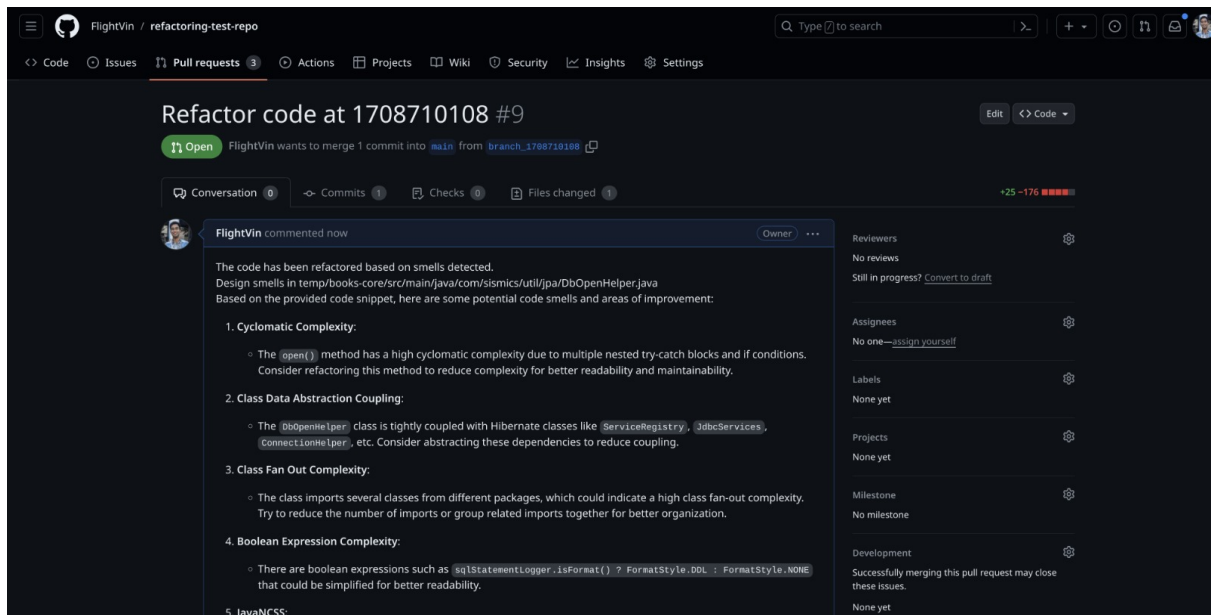
```

    }
    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            log.error("Error closing connection", e);
        }
    }
    if (outputFileWriter != null) {
        try {
            outputFileWriter.close();
        } catch (IOException e) {
            log.error("Error closing output file writer", e);
        }
    }
}

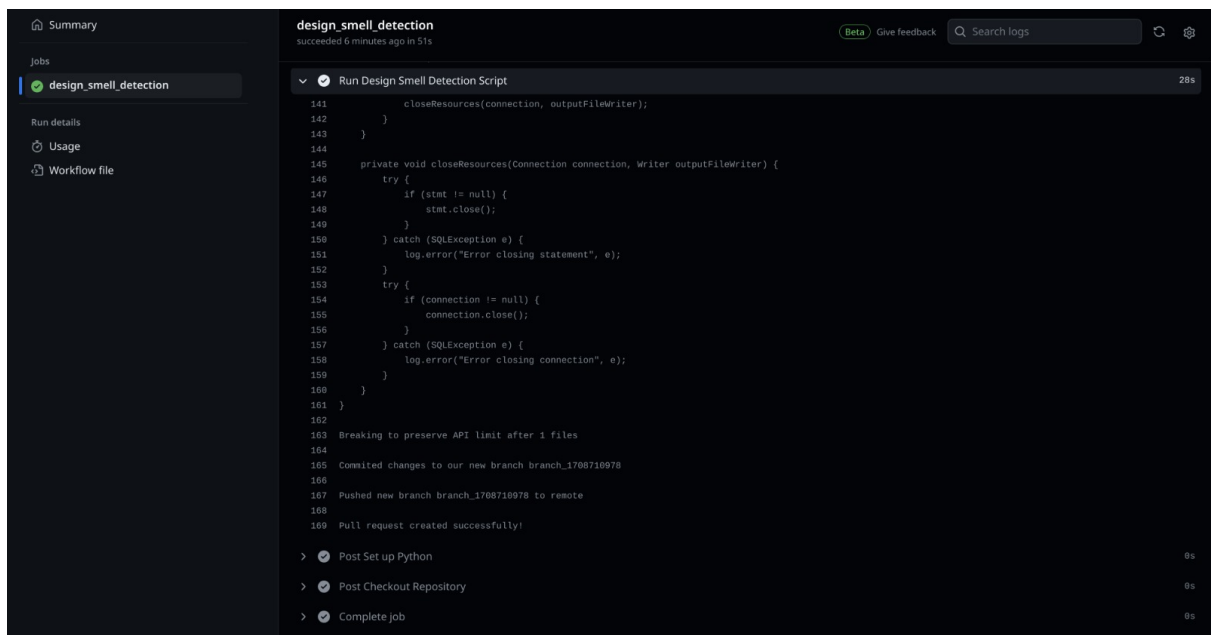
```


Breaking to preserve API limit after 1 files
Committed changes to our new branch branch_1708709504
Pushed new branch branch_1708709504 to remote
Pull request created successfully!
```

## Pull Request:



**Our GitHub Action specified CRON Job successfully executes and completes the pipeline.**



**We specify the file name which is being refactored, and the smells also in the PR. Everything is automated by our script :)**

You can check out the PRs and all their descriptions here:

<https://github.com/FlightVin/refactoring-test-repo>

**This marks the end of our implementation of the bonus section.**