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

Description:

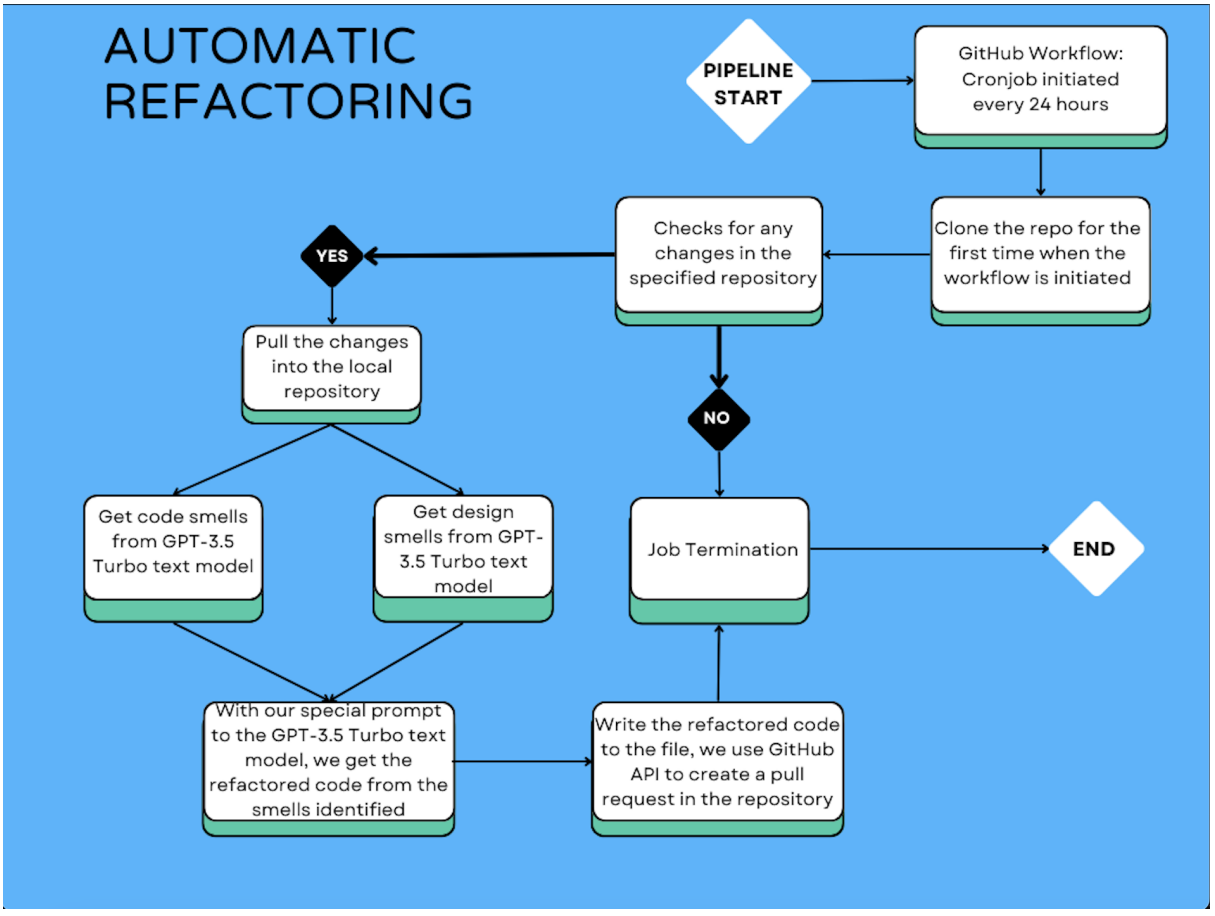
- We've implemented a **GitHub workflow that runs a CRON job every 24 hours**. The YML file for it can be found at `.github/workflows/design-smell-check.yml`
- Our current pipeline just uses the `scripts/scripts.py` as a monolithic codebase.
- The repository is cloned if not already done under which 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:

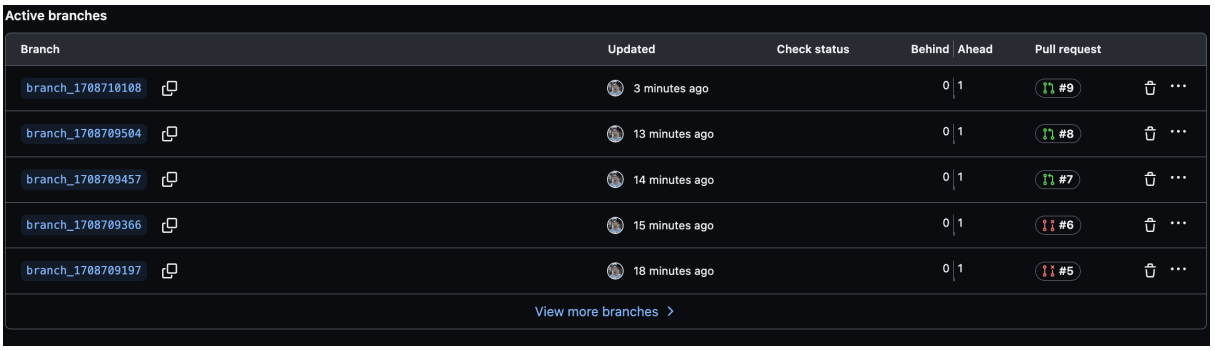
- One thing to note is that since we are using OpenAI library's `0.28.0 version`, we cannot use the codex models like **code-davinci-002** which are specially curated for an excellent code analysis and suggestions. The reason why we cannot use this model is because *OpenAI has deprecated these codex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

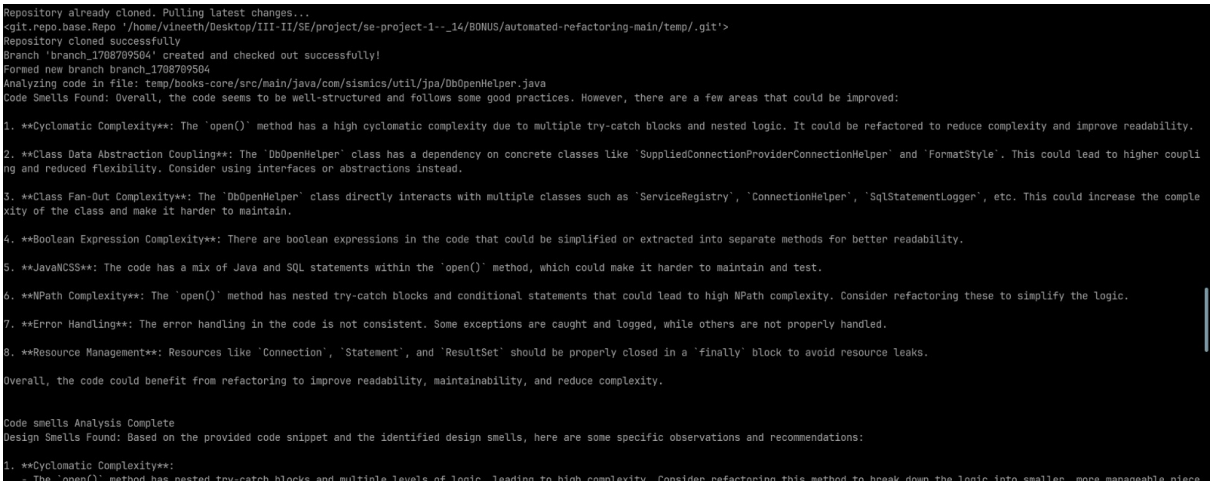
Flowchart to explain our approach:



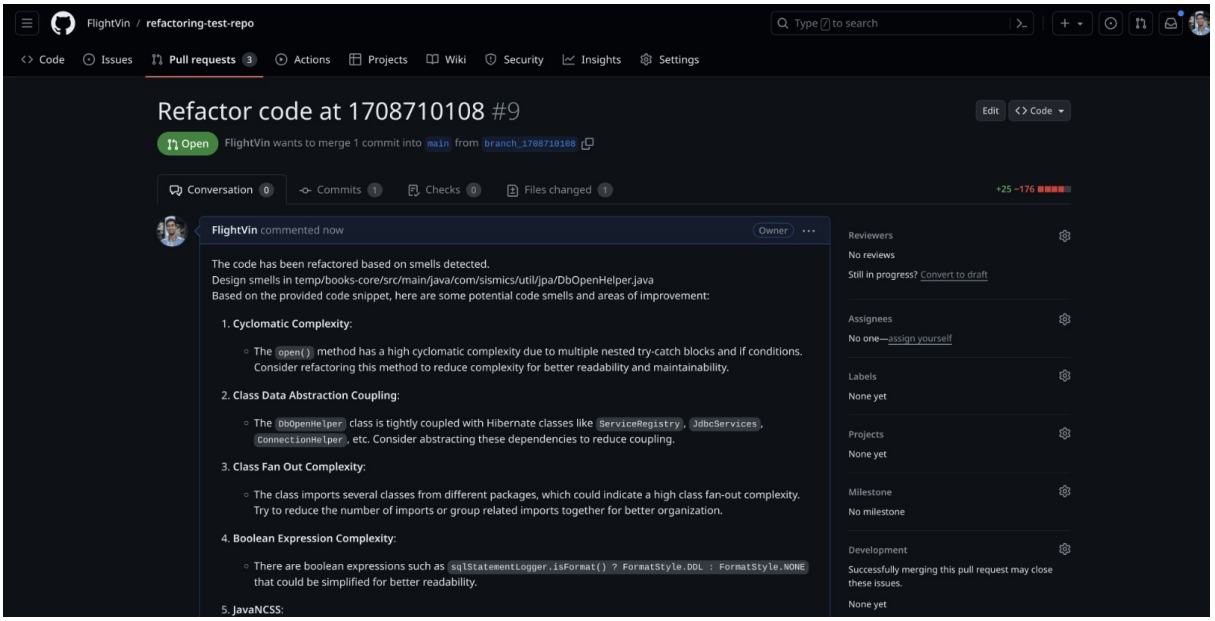
Note that we create a branch first before doing a pull request, we specify the branch name via the timestamp since the UTC.



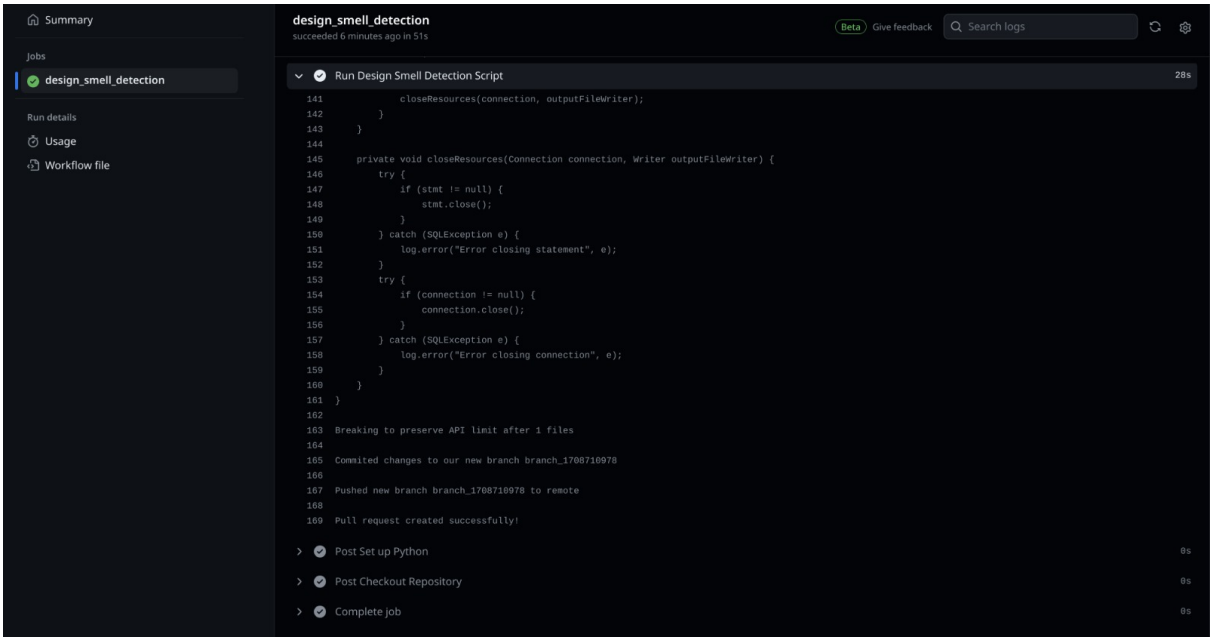
Example output:



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.