Bilkent University
Department of Computer Engineering

Alkahest

An auto code reviewing extension for VSCode

CS 453 Application Lifecycle Management
Term Project Final Report

GROUP 1F

Alper Göçmen 22002948

Sarper Arda Bakır 21902781

Deniz Tuna Onguner 22001788

Section 1

Instructor: Eray Tüzün

Teaching Assistant(s): Yahya Elnouby & Umut Cihan

Final Report - May 13, 2024, Monday

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Application Lifecycle Management course – CS 453.

Table of Contents

Application Overview

Alkahest entails the development of a comprehensive Visual Studio Code (VSCode) plugin meticulously tailored for software practitioners. Unlike existing solutions that often necessitate navigating through many websites and tools, our plugin serves as an all-encompassing toolset within the familiar VSCode environment. It amalgamates crucial features such as code duplication detection, bug identification, and resolution assistance, aiming to streamline the software development process and heighten productivity.

In software development, the fragmentation of existing solutions imposes a significant challenge, compelling developers to switch between disparate platforms, leading to productivity losses and workflow disruptions. The proposed VSCode plugin addresses these challenges by providing a cohesive and integrated solution, consolidating diverse functionalities into a user-friendly interface. Eliminating the need for constant tool-switching enhances workflow continuity and collaboration. While platforms like OpenAI's ChatGPT and GitHub Copilot provide valuable insight and recommendations, they may not be as proficient as SonarQube in the contextual understanding required for precise code analysis and error detection. In practice, while receiving support from SonarQube in the code analysis section, a more detailed solution process is progressed with the Gemini API in specific areas. Similarly, tools like Review Board and Crucible excel in facilitating code review processes but may need more automatic bug detection and integration with development workflows. Our plugin bridges these gaps by seamlessly integrating with VSCode, leveraging AI-driven bug detection, and providing real-time code analysis, thus ensuring reliability, scalability, and performance while minimizing workflow interruptions.

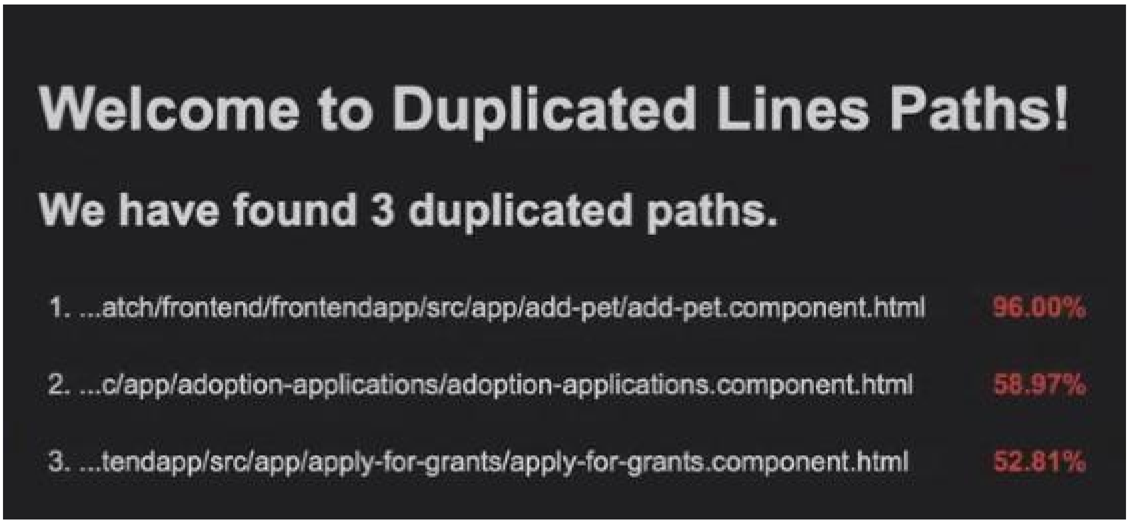
In application, users can utilize three different features: analysis of project, finding duplicate lines, and detecting bugs. When users run the scan command, results are displayed to the user via the secondary sidebar of VSCode. These metrics are several bugs that are coding errors that will break user code and need to be fixed immediately, code smells that is code that is confusing and difficult to maintain, duplicated lines density that is identical lines of code, cognitive complexity that is a measure of how difficult the application is to understand, NCLOC (non-commented lines of code) that is the number of non-commented lines of code in the project, and vulnerabilities that is code that hackers can exploit. If the user has scanned the code before, the application asks the user if they want to scan again. Additionally, once the scan starts, the user can cancel the scan if desired. When the user wants to find duplicated lines, all files containing duplicated lines are listed in a VSCode sidebar, showing what percentage of the file they are. When the file is clicked on the sidebar, it is opened for the user and the duplicated lines are highlighted. As a last feature, when the user wants to detect bugs, a VSCode sidebar provides information on whether the bugs detected are minor, major, blocker or critical. In this file, they are located, their description, and on which line they are located. When the user goes to the file, the line that caused the error is highlighted again, and the user is asked right above the line whether he wants to solve this problem with Gemini AI. When the solution is clicked, the solution from Gemini AI is shown to the user in another sidebar. In this way, thanks to the application, the user can scan his codes, detect duplicated lines, and find bugs via VSCode, the platform on which he wrote his code, thanks to SonarQube, and solve the problems with Gemini AI while remaining on the same platform.

Validation of the Application

Successful Part:

Our application is successful in terms of code duplication reduction. We claimed at least 80% accuracy on finding duplicated lines. In other words, we aim to find at least 80% of the duplicated lines. SonarQube provides the duplicated lines locations, but it does not give the accuracy directly. Thus, to prove our success criteria is met, we prepared the files and tested the results manually.

To prove that the application is successful, we prepared two instances. We compared two files, and we saw the similarities between them.



Welcome to Duplicated Lines Paths!

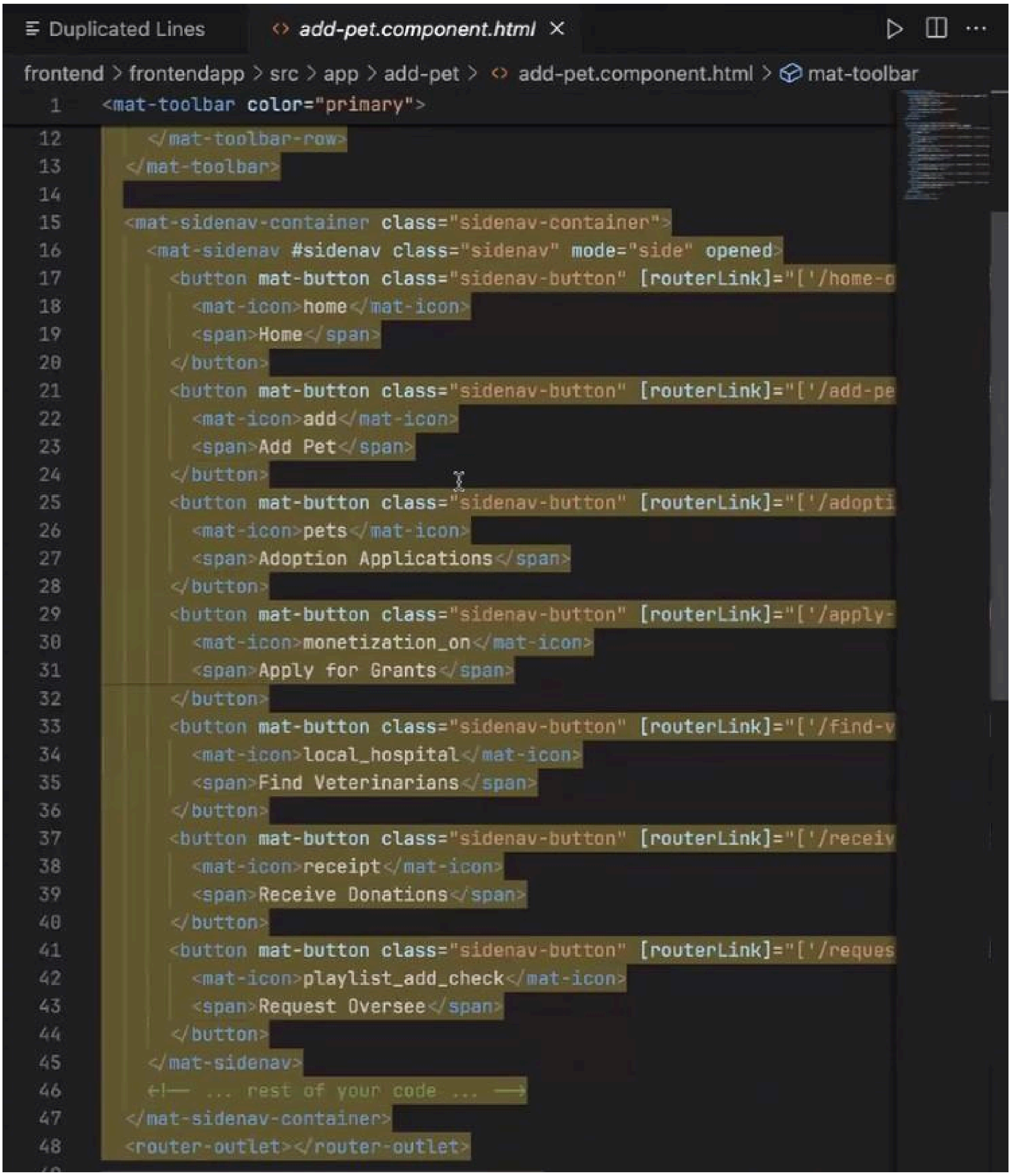
We have found 3 duplicated paths.

```
1. ...atch/frontend/frontendapp/src/app/add-pet/add-pet.component.html       96.00%
2. ...c/app/adoption-applications/adoption-applications.component.html       58.97%
3. ...tendapp/src/app/apply-for-grants/apply-for-grants.component.html       52.81%
```

We have 3 files that consist of repetitive lines. From the SonarQube results, we calculated the density. For instance, the first file has a density of 96%. It means that, if the first file has 100 lines of code, 96 lines of them exist in another file in the scanned folder.

Our first instance is the comparison between File 1 and File 2.
We can understand from the 96%, nearly all lines of file 1 exist in some other file which is file 2, in our case.

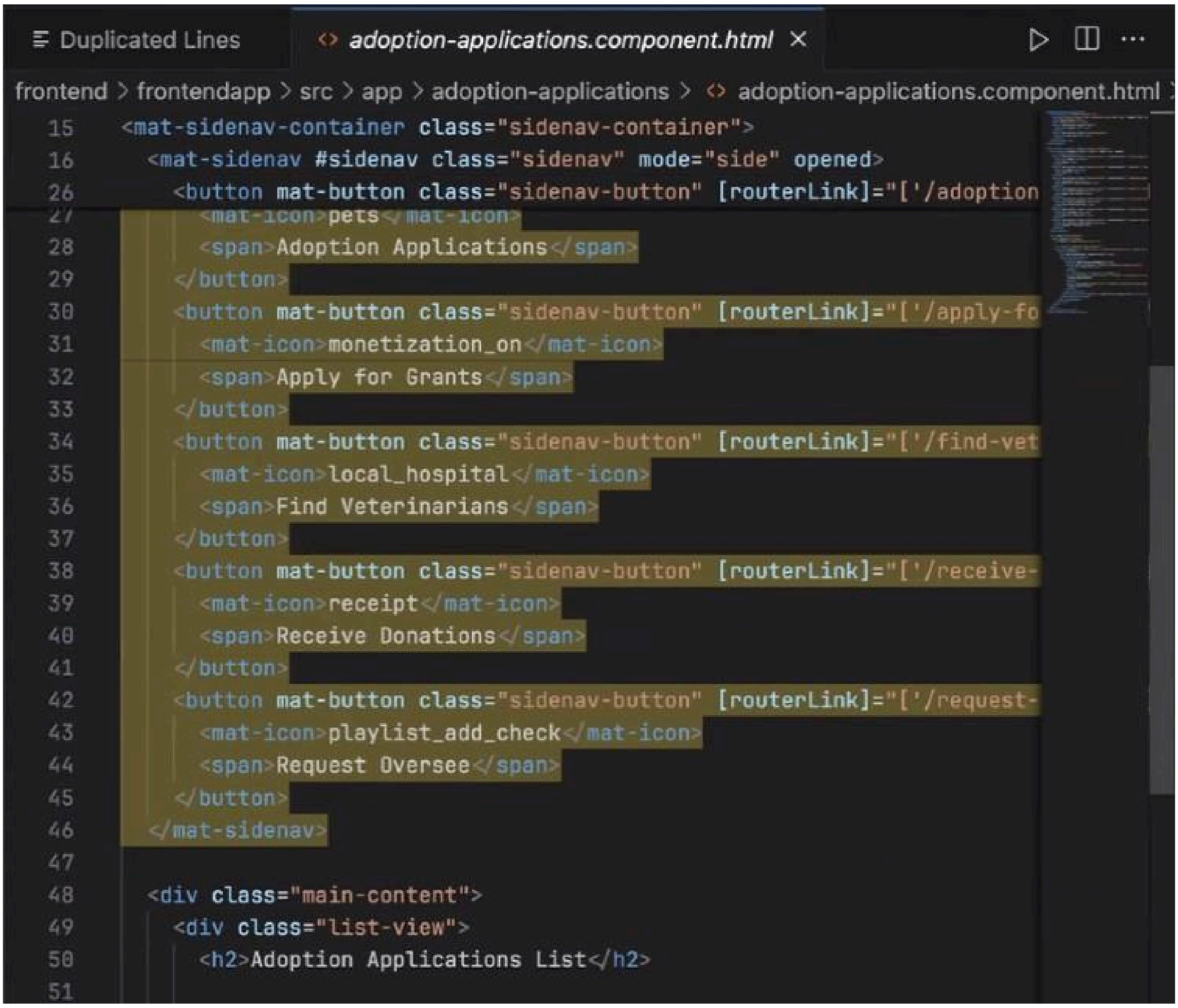
You can see the duplicated lines in the File 1 below:



Duplicated Lines
add-pet.component.html
frontend > frontendapp > src > app > add-pet > add-pet.component.html > mat-toolbar

```html
<mat-toolbar color="primary">
    </mat-toolbar-row>
  </mat-toolbar>

  <mat-sidenav-container class="sidenav-container">
    <mat-sidenav #sidenav class="sidenav" mode="side" opened>
      <button mat-button class="sidenav-button" [routerLink]="['/home-o
        <mat-icon>home</mat-icon>
        <span>Home</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/add-pe
        <mat-icon>add</mat-icon>
        <span>Add Pet</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/adopti
        <mat-icon>pets</mat-icon>
        <span>Adoption Applications</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/apply-
        <mat-icon>monetization_on</mat-icon>
        <span>Apply for Grants</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/find-v
        <mat-icon>local_hospital</mat-icon>
        <span>Find Veterinarians</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/receiv
        <mat-icon>receipt</mat-icon>
        <span>Receive Donations</span>
      </button>
      <button mat-button class="sidenav-button" [routerLink]="['/reques
        <mat-icon>playlist_add_check</mat-icon>
        <span>Request Oversee</span>
      </button>
    </mat-sidenav>
    <!-- ... rest of your code ... -->
  </mat-sidenav-container>
  <router-outlet></router-outlet>
```

You can see the duplicated lines in the File 2, and it has 58.97% duplicated lines.


Duplicated Lines
adoption-applications.component.html
frontend > frontendapp > src > app > adoption-applications > adoption-applications.component.html

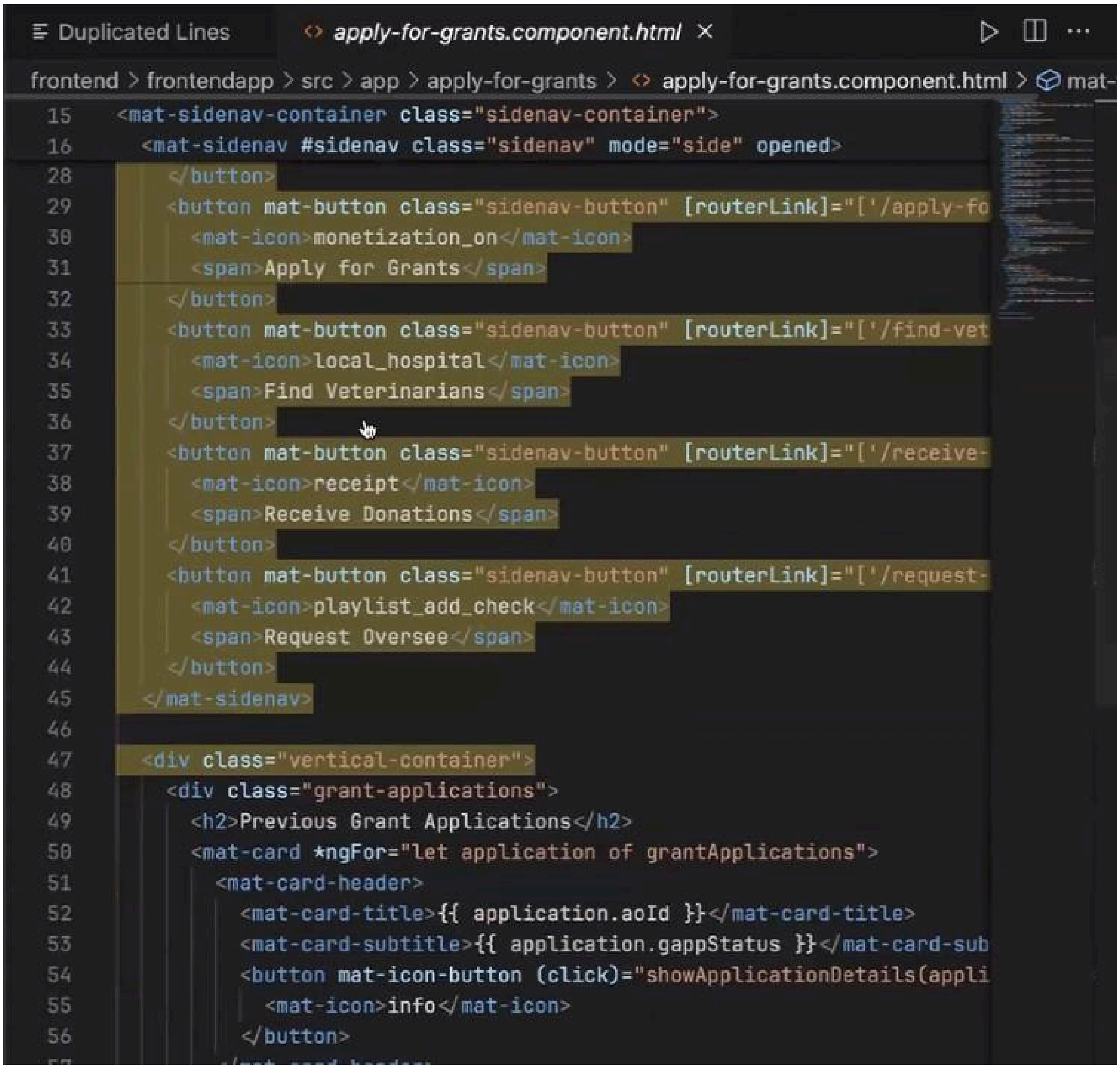
```
<mat-sidenav-container class="sidenav-container">
  <mat-sidenav #sidenav class="sidenav" mode="side" opened>
    <button mat-button class="sidenav-button" [routerLink]="['/adoption
      <mat-icon>pets</mat-icon>
      <span>Adoption Applications</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/apply-fo
      <mat-icon>monetization_on</mat-icon>
      <span>Apply for Grants</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/find-vet
      <mat-icon>local_hospital</mat-icon>
      <span>Find Veterinarians</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/receive-
      <mat-icon>receipt</mat-icon>
      <span>Receive Donations</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/request-
      <mat-icon>playlist_add_check</mat-icon>
      <span>Request Oversee</span>
    </button>
  </mat-sidenav>

  <div class="main-content">
    <div class="list-view">
      <h2>Adoption Applications List</h2>

```

When we compare these two files, we check if the colored lines are the same in the two files. Also, we checked if there exist any repetitive code snippets that are not colored by the extension. There was not any duplicated code other than the colored ones.

So as a result of the first instance, we can say that our application is not missing any similarity between these two files. Thus, the accuracy is 80%+ as we claimed.

Our second instance is the comparison between File 1 and File 3.

You can see the duplicated lines in the File 3, and it has 52.81% duplicated lines.

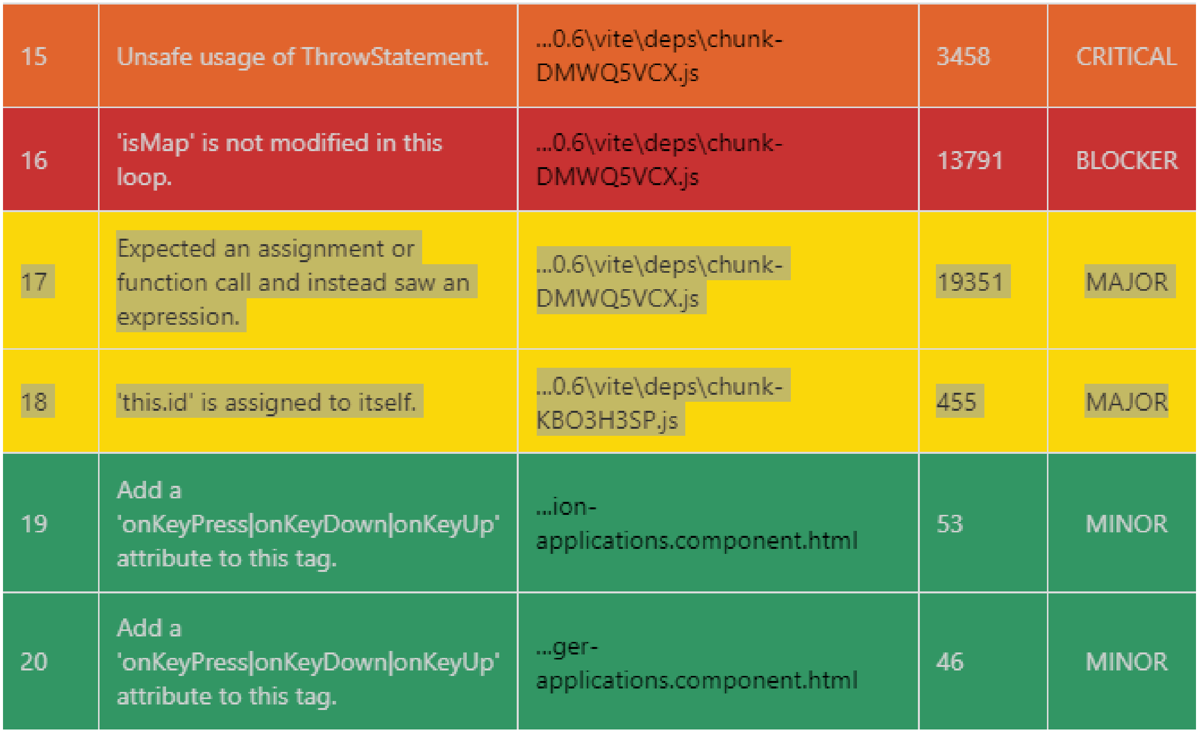

Duplicated Lines
apply-for-grants.component.html
frontend > frontendapp > src > app > apply-for-grants > apply-for-grants.component.html > mat-

```
<mat-sidenav-container class="sidenav-container">
  <mat-sidenav #sidenav class="sidenav" mode="side" opened>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/apply-fo
      <mat-icon>monetization_on</mat-icon>
      <span>Apply for Grants</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/find-vet
      <mat-icon>local_hospital</mat-icon>
      <span>Find Veterinarians</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/receive-
      <mat-icon>receipt</mat-icon>
      <span>Receive Donations</span>
    </button>
    <button mat-button class="sidenav-button" [routerLink]="['/request-
      <mat-icon>playlist_add_check</mat-icon>
      <span>Request Oversee</span>
    </button>
  </mat-sidenav>

  <div class="vertical-container">
    <div class="grant-applications">
      <h2>Previous Grant Applications</h2>
      <mat-card *ngFor="let application of grantApplications">
        <mat-card-header>
          <mat-card-title>{{ application.aoId }}</mat-card-title>
          <mat-card-subtitle>{{ application.gappStatus }}</mat-card-sub
          <button mat-icon-button (click)="showApplicationDetails(appli
            <mat-icon>info</mat-icon>
          </button>
```

Again, we saw that all colored lines are the same as the first one. Not colored ones are the different code snippets. As a result, we can confirm that we are getting 80%+ duplicated lines accuracy in comparison between File 1 and 3.

As an overall result, we can say that we achieved the threshold of 80% in finding duplicated lines. Thus, our success criteria (5.4) is met.

Unsuccessful Part:

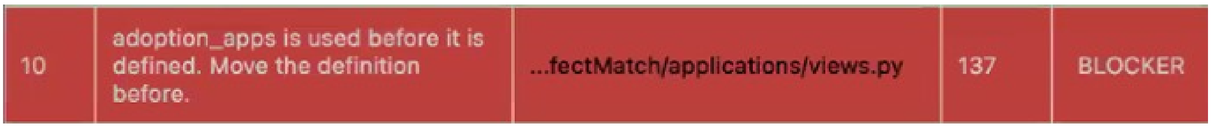


15	Unsafe usage of ThrowStatement.	...0.6\vite\deps\chunk-DMWQ5VCX.js	3458	CRITICAL
16	'isMap' is not modified in this loop.	...0.6\vite\deps\chunk-DMWQ5VCX.js	13791	BLOCKER
17	Expected an assignment or function call and instead saw an expression.	...0.6\vite\deps\chunk-DMWQ5VCX.js	19351	MAJOR
18	'this.id' is assigned to itself.	...0.6\vite\deps\chunk-KBO3H3SP.js	455	MAJOR
19	Add a 'onKeyPress|onKeyDown|onKeyUp' attribute to this tag.	...ion-applications.component.html	53	MINOR
20	Add a 'onKeyPress|onKeyDown|onKeyUp' attribute to this tag.	...ger-applications.component.html	46	MINOR

This table consists of each bug severity type in our application which are minor, major, critical and blocker. So, when we say a bug, it could imply more than one thing. For instance, the minor bugs are not seen as a warning in the VSCode, but it is found as a bug by SonarQube.

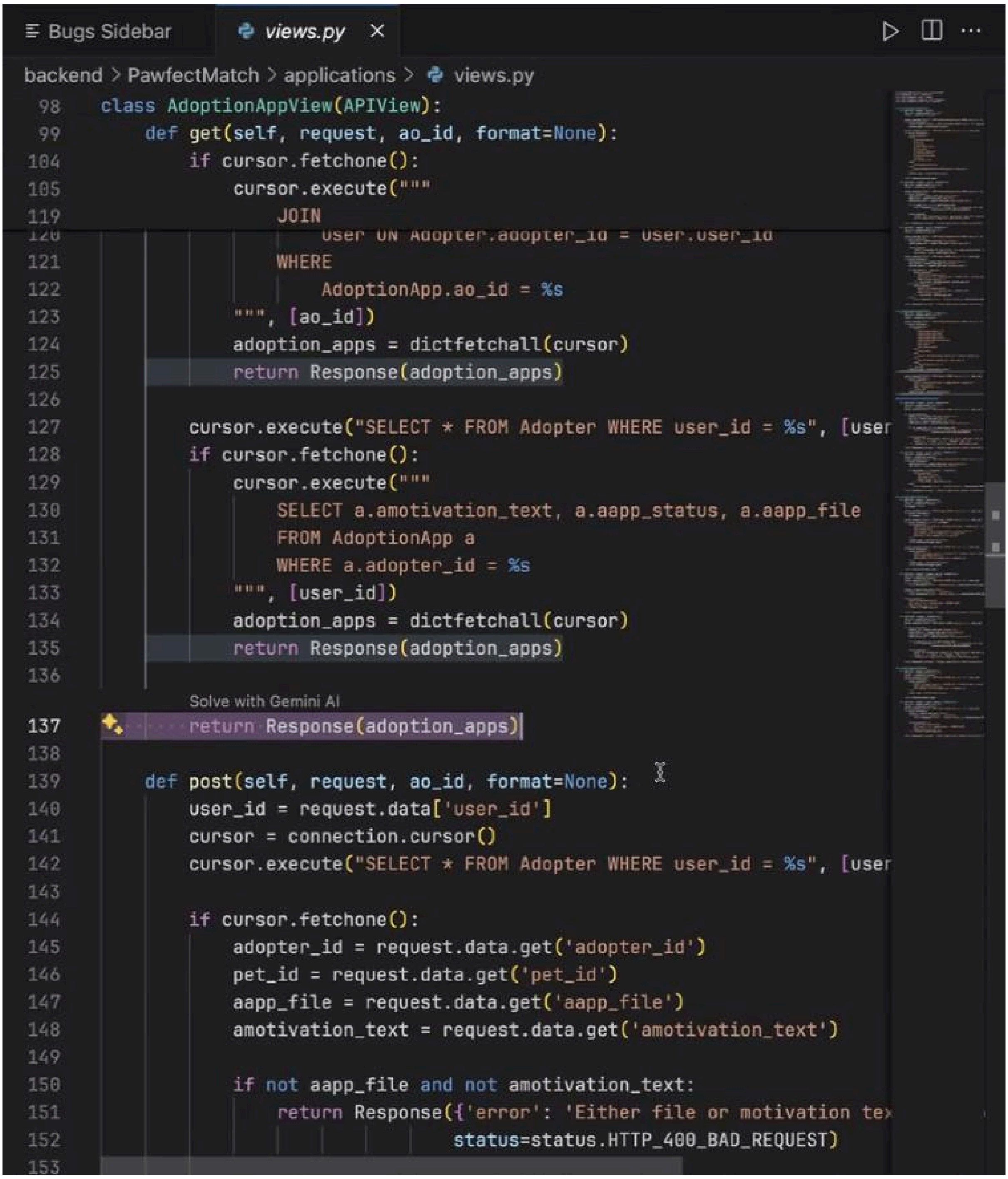
After finding bugs, we offer a solution using Gemini AI in our application. However, our application is unsuccessful in terms of code quality enhancement part, especially in the AI generated response. We prepared two instances to prove that we have not achieved the threshold, and the application does not met the success criteria

The first instance is the bug below. In this file, SonarQube found a bug at line 137. It is a blocker bug, since if it is not fixed, the program will not run.



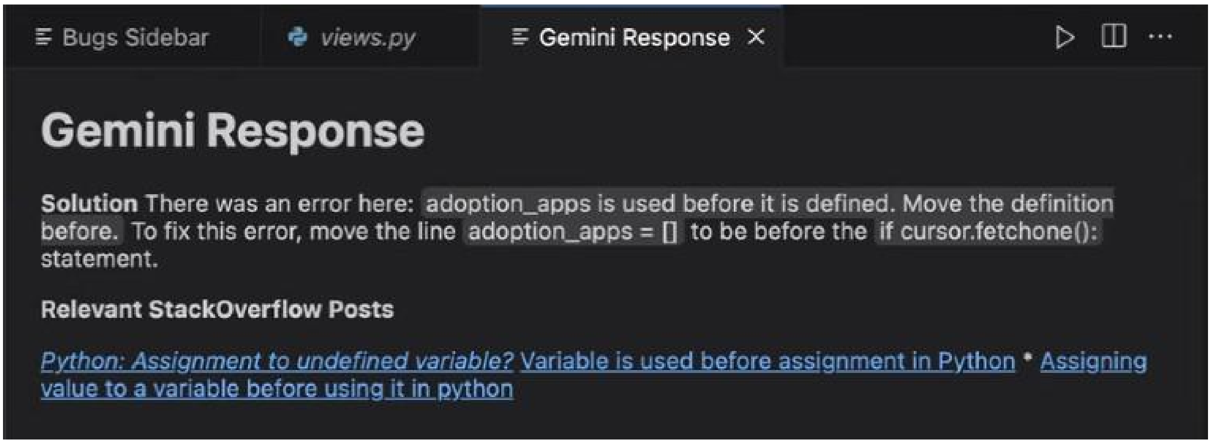
10	adoption_apps is used before it is defined. Move the definition before.	...fectMatch/applications/views.py	137	BLOCKER

The definition says that “adoption_apps” is used before it is defined. SonarQube says to move the definition before, but it does not say exactly where.



```python
class AdoptionAppView(APIView):
    def get(self, request, ao_id, format=None):
        if cursor.fetchone():
            cursor.execute("""
                JOIN
                    user ON Adopter.adopter_id = user.user_id
                WHERE
                    AdoptionApp.ao_id = %s
            """, [ao_id])
            adoption_apps = dictfetchall(cursor)
            return Response(adoption_apps)

        cursor.execute("SELECT * FROM Adopter WHERE user_id = %s", [user
        if cursor.fetchone():
            cursor.execute("""
                SELECT a.amotivation_text, a.aapp_status, a.aapp_file
                FROM AdoptionApp a
                WHERE a.adopter_id = %s
            """, [user_id])
            adoption_apps = dictfetchall(cursor)
            return Response(adoption_apps)

        return Response(adoption_apps)

    def post(self, request, ao_id, format=None):
        user_id = request.data['user_id']
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM Adopter WHERE user_id = %s", [user

        if cursor.fetchone():
            adopter_id = request.data.get('adopter_id')
            pet_id = request.data.get('pet_id')
            aapp_file = request.data.get('aapp_file')
            amotivation_text = request.data.get('amotivation_text')

            if not aapp_file and not amotivation_text:
                return Response({'error': 'Either file or motivation tex
                                status=status.HTTP_400_BAD_REQUEST)
```

When we analyzed that by looking at it, we realized that this bug occurs because of the scope in the related file. Just by looking, we could not find the solution, so we asked to the Gemini.



Gemini Response

Solution There was an error here: adoption_apps is used before it is defined. Move the definition before. To fix this error, move the line adoption_apps = [] to be before the if cursor.fetchone(): statement.

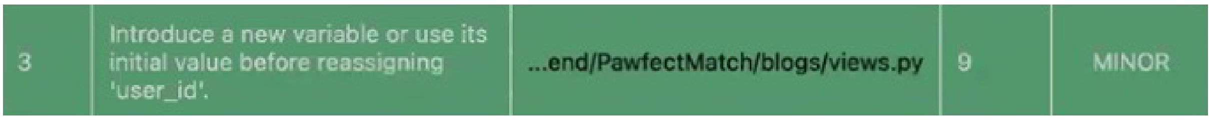
Relevant StackOverflow Posts

Python: Assignment to undefined variable? Variable is used before assignment in Python • Assigning value to a variable before using it in python

This was the response we got. One of the main issues is that, in the response it says that move the line “adoption_apps[“] to be before the “if cursor.fetchone():” statement. However there were more than one “if cursor.fetchone():” statement in that function, so by only looking at the answer of the Gemini, we are still not able to fix the bug.

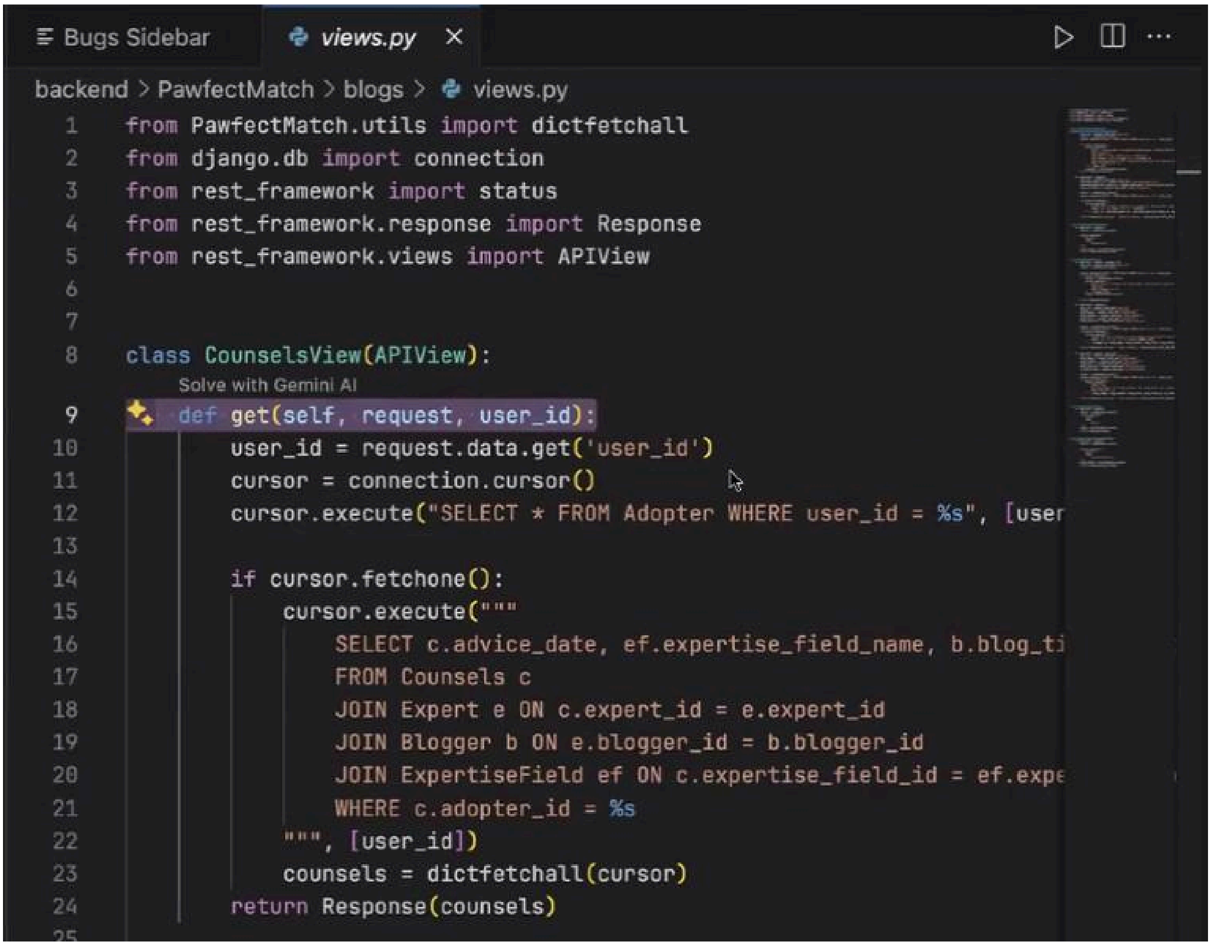
The second issue is the lack of relevant solutions in StackOverFlow. We expect to get at least 3 solutions from the web, as we stated in the success criteria. However, the criteria is not met in this case.

The second instance is the bug below. In this file, SonarQube found a bug at line 9:



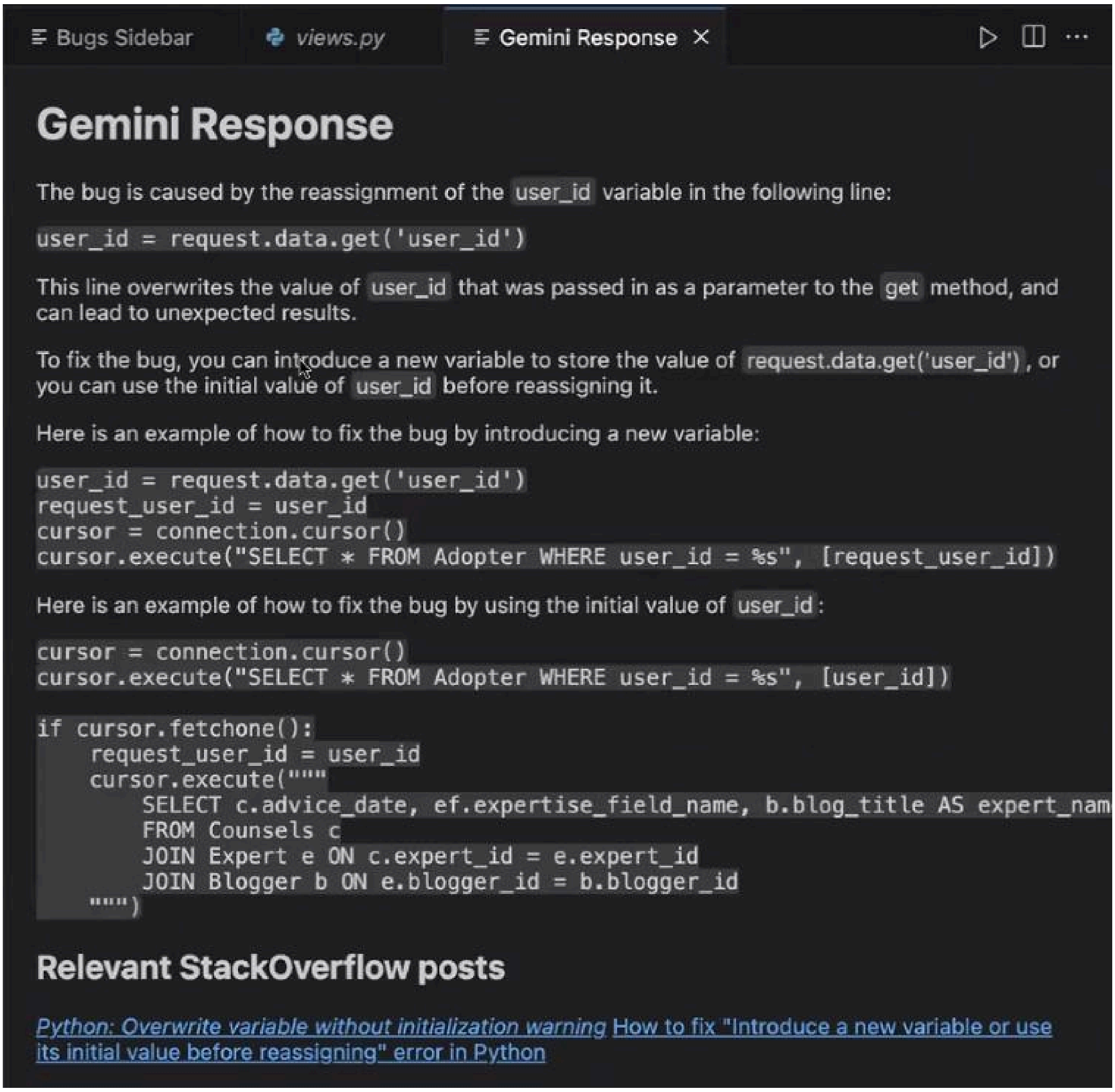
3	Introduce a new variable or use its initial value before reassigning 'user_id'.	...end/PawfectMatch/blogs/views.py	9	MINOR

The definition says “Introduce a new variable or use its initial value before reassigning ‘user_id”, and it is a minor bug.



```python
from PawfectMatch.utils import dictfetchall
from django.db import connection
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView


class CounselsView(APIView):
    def get(self, request, user_id):
        user_id = request.data.get('user_id')
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM Adopter WHERE user_id = %s", [user

        if cursor.fetchone():
            cursor.execute("""
                SELECT c.advice_date, ef.expertise_field_name, b.blog_ti
                FROM Counsels c
                JOIN Expert e ON c.expert_id = e.expert_id
                JOIN Blogger b ON e.blogger_id = b.blogger_id
                JOIN ExpertiseField ef ON c.expertise_field_id = ef.expe
                WHERE c.adopter_id = %s
            """, [user_id])
            counsels = dictfetchall(cursor)
        return Response(counsels)
```

When we analyzed by looking at the code, we saw that we are losing the value passed as a parameter that we are getting from the function. So we can solve it by introducing a new variable before we assign user_id. Even though it is a simple bug and it can be solved by looking at it, we asked Gemini to see if it gives the correct solution.



Gemini Response

The bug is caused by the reassignment of the `user_id` variable in the following line:

```
user_id = request.data.get('user_id')
```

This line overwrites the value of `user_id` that was passed in as a parameter to the `get` method, and can lead to unexpected results.

To fix the bug, you can introduce a new variable to store the value of `request.data.get('user_id')`, or you can use the initial value of `user_id` before reassigning it.

Here is an example of how to fix the bug by introducing a new variable:

```
user_id = request.data.get('user_id')
request_user_id = user_id
cursor = connection.cursor()
cursor.execute("SELECT * FROM Adopter WHERE user_id = %s", [request_user_id])
```

Here is an example of how to fix the bug by using the initial value of `user_id`:

```
cursor = connection.cursor()
cursor.execute("SELECT * FROM Adopter WHERE user_id = %s", [user_id])

if cursor.fetchone():
    request_user_id = user_id
    cursor.execute("""
        SELECT c.advice_date, ef.expertise_field_name, b.blog_title AS expert_nam
        FROM Counsels c
        JOIN Expert e ON c.expert_id = e.expert_id
        JOIN Blogger b ON e.blogger_id = b.blogger_id
    """)
```

Relevant StackOverflow posts

Python: Overwrite variable without initialization warning How to fix "Introduce a new variable or use its initial value before reassigning" error in Python

This was the response we got. In the response we can see that the Gemini understood the problem. It gives two solutions. In the first one, the code snippet it gives to us is nearly the same with the current code. It uses user_id, then it introduces a new variable, but it also loses the initial value. Thus, it does not solve the problem

In the second one, it uses the initial value. However, without changing the “user_id = request.data.get(‘user_id’)” line, we cannot use the initial value. Thus, the second solution is also wrong.

Therefore, the main issue in the response was the solutions were not correct.

Lastly, the same issue with the previous one occurred, which was the lack of relevant solutions in StackOverFlow. We expect to get at least 3 solutions from the web, as we stated in the success criteria. However, the criteria is not met in this case.

In general, we can say that our application does not satisfy the success criteria code quality enhancement (5.3), since our response does not include 50% correct code. Also, if it does not include any code snippet, we are not able to guide the user correctly since we are not able to provide enough related solutions on the web.

Work Allocation

Deniz Tuna Onguner

Project Proposal: Possible technical solutions to use in the application, The mock-up image.
Progress Report: Business Questions, Functional and Non-functional Requirements, Architecture diagram, Activity diagram, Half of the mock-up images.
Final Report: User manual, Build instructions, Screen-recording for the demo video.
Code: Initializing the project and the GitHub repository, Integrating the Web API connections of SonarCloud and Google Generative AI – Gemini, Integrating the SonarQube, Implementing the scanning feature via SonarQube, Implementing the solution(s) generation and finding related links via Gemini features, Designing and implementing the measures sidebar view and the Gemini response sidebar view, Coded the feature of highlighting the bugs found with the color red, Converted Gemini's markdown responses to HTMLs.

Sarper Arda Bakır

Project Proposal: Initial High-Level Requirements, Competitor Applications.
Progress Report: Problem Statement, Application Use Case, Mock-up images.
Final Report: Application Overview, Planned Work and Delivered Work.
Code: Integrating the SonarQube Duplications API, Designing and implementing the duplicated lines side bar, Highlighting duplicated lines in files.

Alper Göçmen

Project Proposal: Application Definition, Brief Description of the Problem, Motivation.
Progress Report: Success Criteria, Sample Data, Technical Benchmark.
Final Report: Validation of the Application.
Code: Integrating the SonarQube Issues API, Designing and implementing the bugs sidebar.

Planned Work and Delivered Work

Planned Work:

- A VSCode plugin application
- Code review feature
- Summarization feature
- Duplication finding feature
  - Enable reviewers to provide feedback on code segment
- Bug identification feature
  - Provide automated or guided solutions for resolving identified issues such as StackOverFlow or SonarQube
- Resolution assistance within the VSCode environment

Delivered Work:

- A VSCode plugin application that multifunctional toolset
- Scan the code and return measures about code from SonarQube
- List files that have duplicated line(s) in VSCode sidebar
- Highlight the duplicated line(s) in the files
- List file that have bug(s) in VSCode sidebar
- Highlight the line(s) that cause bug in the files
- Recommend a solution how user can fix the bug with Gemini AI

User Manual

The user manual is also available in the repository via the following link: User Manual

Phase 1: Scanning

- **Step 1:** After the extension is started, open the folder of the project that you want to auto-review.
- **Step 2:** Then, use the command **`SonarQube Scan`** to start scanning. This may take some time depending on your active connection speed and the size of the code base you are trying to scan. Be patient and wait for the scanning to complete.
- **Step 3:** You will see the overall review of your project which will be automatically displayed once the scanning is completed. This view is static unless you re-scan the project, feel free to close it; you can always bring it back via the command **`SonarQube Get Measures`**.

Phase 2: Evaluation

- **Duplications:** Use the command **`SonarQube Get Duplications`** to see the files with duplicated lines. This view is dynamic, click on the filename to direct to the file; the duplicated lines will be highlighted with the color yellow.
- **Bugs:** Use the command **`SonarQube Get Bugs`** to see the bugs found. In the view, you can see where the bug is, why it is an issue, and how urgently it needs to be solved, i.e. its severity. This view is also dynamic, click on the path to direct to the file, the line causing the bug will be highlighted with the color red. At the top of this line, you will see an option, i.e. VSCode CodeLens, saying **`Solve with Gemini AI`**. Click it to ask Gemini AI's help to solve the bug, this action will launch another sidebar view with Gemini's response once it is received.

Additional Notes

Scan command will automatically initialize a **`sonar-project.properties`** file unless there is already one. Do not delete it, it is required. If you delete it, Alkahest will continue to create a new one again and again.

Other Deliverables

Code: Tuna-Onguner/Alkahest

Build Instructions: alkahest/README.md

Demo Video: Alkahest Demo