TDT4242 :Advance Software Engineering

Spring 2025

Assignment 4: Code Smell and Code Refactoring

Group 18

Ana Barrera Novas

Gargi Ketan Chauhan

Garima Ketan Chauhan

Task 1: Code Review and Refactoring

Code Smell ID	Source Link	Type	Code snippet	Manual check result	Explanation
S01	backend/secfit /settings.py	Long Method	All file	Incorrect	It is true that it is a large file with many settings defined in a single place, but it makes sense that it is this way as it is the settings file of the application.
S02	backend/secfit /settings.py	Duplicate Code	lines 124-137	Correct	The pattern of 'NAME': 'django.contrib.auth.password_validation …' is repeated multiple times. Instead of a loop or a reusable structure, it duplicates dictionary objects for similar purposes. Not critical, only could affect extensibility.
S03	backend/secfit /settings.py	Comments	All file	Incorrect	The AI considered it smell as comments can be over explanatory and easily ignored, but in this case all comments ease readability and remind the developer what should be deleted or modified in the long run.

S04	backend/users /views.py	Long Method	UserDetail.get()
			
			OfferDetail.put()
				Correct	These methods contain many responsibilities (query logic, data handling, SQL, conditions branching, serialization…). This can affect readability, testability and maintainability.

S05	backend/users /views	Blob	UserDetail.get()
			
			OfferDetail.put()
				Correct	For the same reasons as the above, these methods could grow into “god objects” handling too much logic.

S06	backend/users /views	Shotgun Surgery	(...) if offer.status == 'a': (…) status='p' (…)	Correct	Based only on the view, it could look like ‘a’ and ‘p’ were magic literals used without central definition. We can see in models.py that actually, they are defined, yet they are not being referenced correctly. The smell led to something that was actually correct, but it is still misused.

S07	backend/users/models.py	Data Clumps	athlete = models.ForeignKey(get_user_model(), ...) owner = models.ForeignKey(get_user_model(), ...) owner = models.ForeignKey(get_user_model(), ...) recipient = models.ForeignKey(get_user_model(), ...)	Correct	All these pairings have a conceptual relationship, yet they are treated as unrelated, leading to duplication and, potentially, to inconsistent handling through models and views. Some of this behavior could be extracted into shared logic
S08	backend/workouts/views.py	Duplicate Code	All view classes	Correct	The classes all repeat similar logic: get(), post(), put(), delete(), permissions declarations… This could be abstracted into base classes to not make this repetitions continuous.

S09	backend/workouts/views.py	Data Clumps	def get(self, request, *args, **kwargs)
			
			def post(self, request, *args, **kwargs
			
			(…)
				Incorrect	The AI has probably detected this as a smell as the same group of parameters appears across many methods, yet in this case it is standard for Django class-based views and it shouldn't be problematic.

S10	backend/workouts/views.py and backend/comments/views.py	Primitive Obsession	Q(workout__visibility="PU")
			
			Q(workout__visibility="CO")
				Correct	"PU" and "CO" are raw strings to represent domain concepts: public and coach. As they are used across different views, they should be defined elsewhere as constants or enums.

Task 2 : Reflective Analysis

ChatGPT scanned through large files and recognised common code patterns associated with code smells like repeated logic and long methods. This is the same as a human reviewer would do manually, but much faster. It was able to recognise many code smells giving the correct prompt, which is very useful. Yet it is still necessary to have a human reviewer involved. We can see that the AI over-detected some code smells, mostly because of the lack of context. It doesn’t always distinguish between code smells and standard framework practices. And it seems hard for the AI to recognise that something can be free of smells, since when you tell it to search for something, it makes its way into finding it.

Still, it is a very useful tool. When handed to someone who can then manually review the smells it can save up a lot of time and effort. In the end, smells are just that, an indication that there might be a deeper problem, which doesn’t always occur.

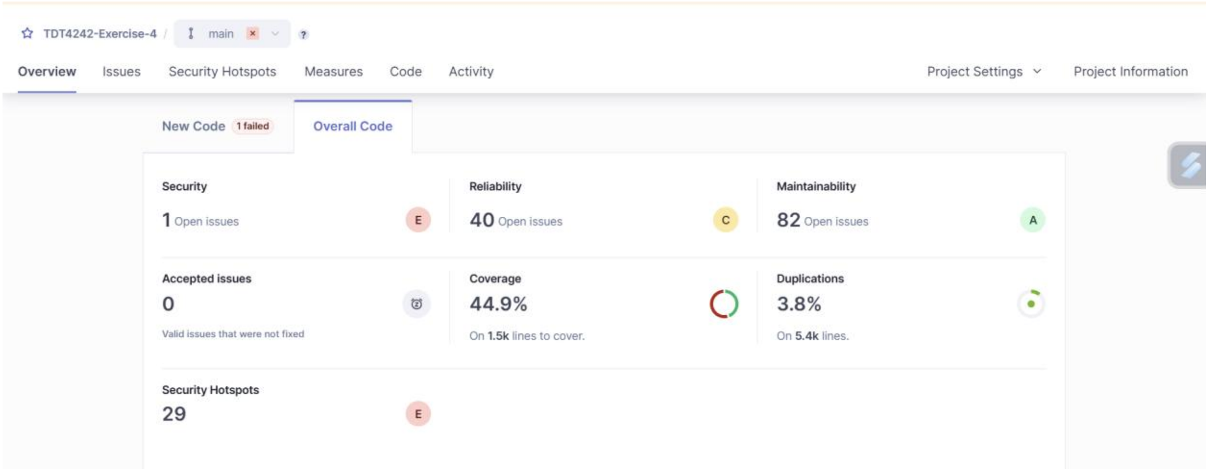
AI tools in general can boost the productivity of an individual or a team in many different ways. For individual software engineering, a tool like ChatGPT can be used as a second reviewer, a second “pair of eyes”. It can help you catch problems when developing, and take into consideration implications that you would not think about in the first place. It can also help you make correct structural decisions even before starting the development, or recommend cleanups or strategies after identifying the correct smells.

And in a team, it can also be a valuable helper. It can give architecture feedback when planning, to help find high level problems or give recommendations in areas to be more careful. It can also be implemented as a documentation checker, flagging missing or redundant documentation, which could improve a lot the maintainability of the project. And lastly, to help the individual developer to be in line with the team, integrated AI in CI/CD could flag smells, identify security problems and suggest improvements before merging.

Task 3 : Code Smell Scanning

We first installed SonarQube and SonarScanner, created a profile and generated the key for our project. We had to create a new file called sonar-project.properties in our Secfit folder and it contained the project key, name, hosting url and the sources. We also set the duplication criteria to a minimum of 10 lines. These are the results obtained:

1. This is the overview of our code. We notice certain Security, Reliability and Maintainability issues. The coverage is 44.9% (It is 67.5% for our backend directory and 0% for the frontend one since our tests do not include any frontend component.) There are a few duplications and Security Hotspots.



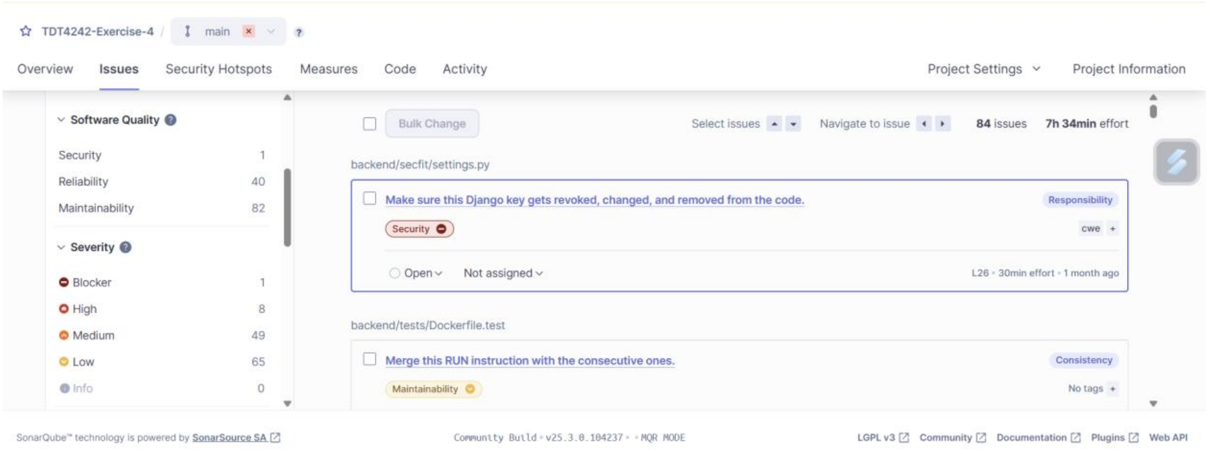
TDT4242-Exercise-4 / main

Overview | Issues | Security Hotspots | Measures | Code | Activity | Project Settings | Project Information

New Code 1 failed | Overall Code

Security: 1 Open issues (E)	Reliability: 40 Open issues (C)	Maintainability: 82 Open issues (A)
Accepted issues: 0 — Valid issues that were not fixed	Coverage: 44.9% — On 1.5k lines to cover.	Duplications: 3.8% — On 5.4k lines.
Security Hotspots: 29 (E)		

2. This is a screenshot of the issues detected.



TDT4242-Exercise-4 / main

Overview | Issues | Security Hotspots | Measures | Code | Activity | Project Settings | Project Information

Software Quality

Security	1
Reliability	40
Maintainability	82

Severity

Blocker	1
High	8
Medium	49
Low	65
Info	0

Bulk Change — Select issues — Navigate to issue — 84 issues — 7h 34min effort

backend/secfit/settings.py

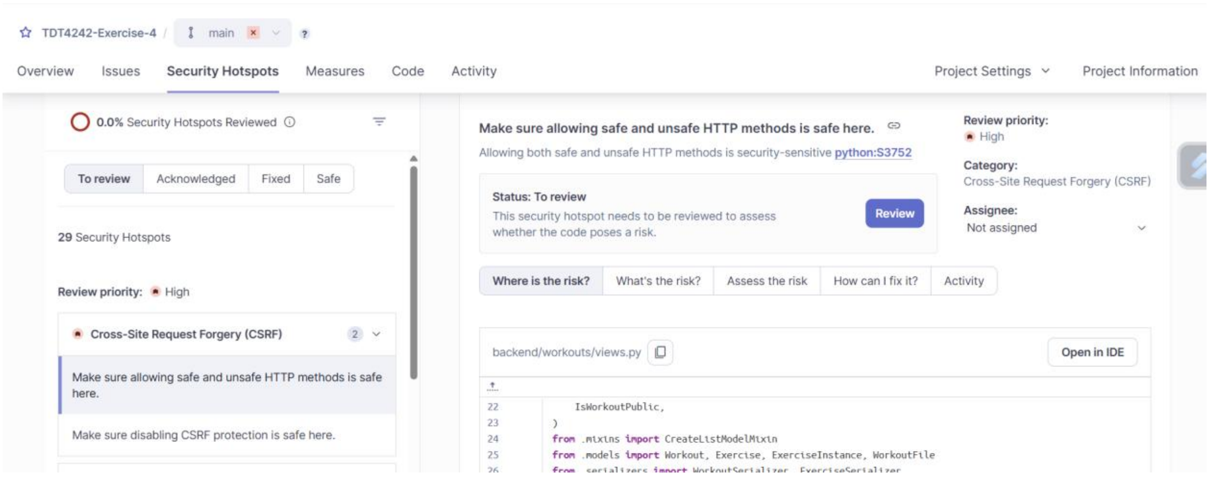
Make sure this Django key gets revoked, changed, and removed from the code. — Responsibility — Security — cwe — Open — Not assigned — L26 · 30min effort · 1 month ago

backend/tests/Dockerfile.test

Merge this RUN instruction with the consecutive ones. — Consistency — Maintainability — No tags

SonarQube™ technology is powered by SonarSource SA — Community Build · v25.3.0.104237 · · MQR MODE — LGPL v3 — Community — Documentation — Plugins — Web API

3. This is a screenshot of the security hotspots detected.



0.0% Security Hotspots Reviewed

29 Security Hotspots

Review priority: High

Cross-Site Request Forgery (CSRF) 2

Make sure allowing safe and unsafe HTTP methods is safe here.

Make sure disabling CSRF protection is safe here.

Make sure allowing safe and unsafe HTTP methods is safe here.

Allowing both safe and unsafe HTTP methods is security-sensitive python:S3752

Status: To review

This security hotspot needs to be reviewed to assess whether the code poses a risk.

Review priority: High

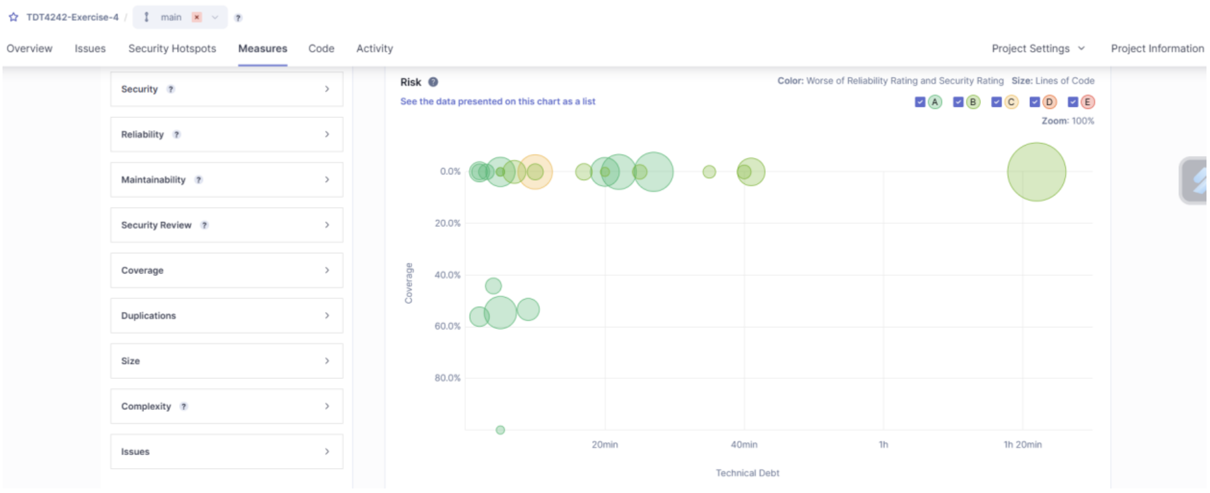
Category: Cross-Site Request Forgery (CSRF)

Assignee: Not assigned

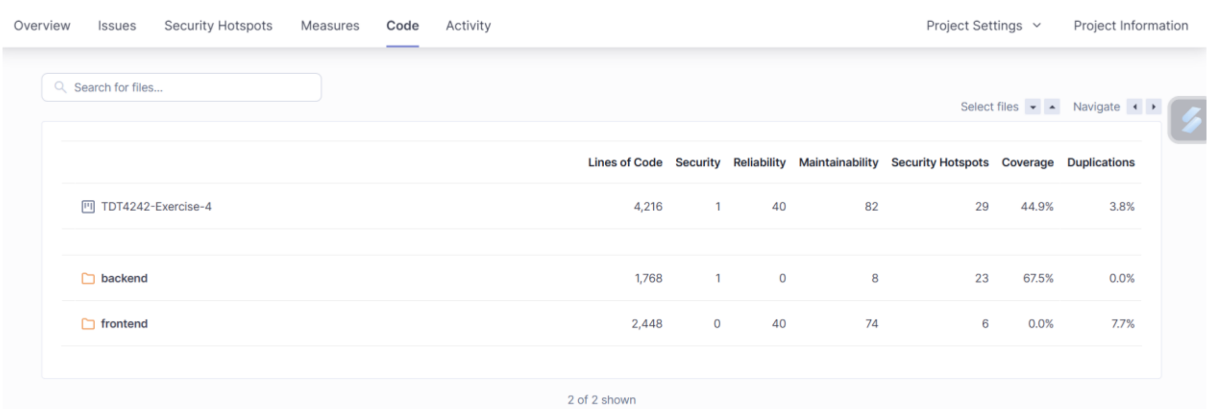
backend/workouts/views.py

```
    IsWorkoutPublic,
)
from .mixins import CreateListModelMixin
from .models import Workout, Exercise, ExerciseInstance, WorkoutFile
from .serializers import WorkoutSerializer, ExerciseSerializer
```

4. Screenshot of the Measures section:



5. It has mainly checked the code under backend and frontend directories and excluded the Dockerfiles and docs, nginx folders. This section gives a directory-level summary.



	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
TDT4242-Exercise-4	4,216	1	40	82	29	44.9%	3.8%
backend	1,768	1	0	8	23	67.5%	0.0%
frontend	2,448	0	40	74	6	0.0%	7.7%

2 of 2 shown

The End

Thank You 😊