Git, GitLab, CI, Code Quality, Security and SonarQube

David Parter,
Computer Sciences Department

UW IT Professionals Conference
June 6, 2019

Continuous Integration

Continuous Deployment

Traditional Release Cycle

- One or more scheduled major and minor releases
- Each release/deployment is an ***event***, and a cause for ***dread***:

Traditional Release Cycle Dread:

Developers and vendors:

- Deadlines!!
- Long delays in getting features & fixes to customers
- Long delays in getting feedback from customers
- Difficulty in understanding all the changes

Customers:

- What changed? What else did it break?
- How long will the upgrade take? How much down time?
- When can we schedule the upgrade?
- Long wait for fixes and features, don’t know if they will work

Continuous Deployment

- Continuous roll-out of new features and fixes (“small batch”)
- Automated deployment
- Managed and Monitored
- Immediate feedback
- Well-suited for Software as a Service providers
- Netflix, Facebook, ...

Use for smaller-scale tools and services too

Continuous Integration

Integrate (commit/check-in) code frequently

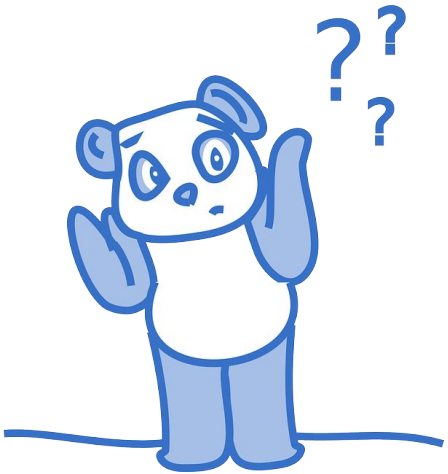
- No massive commits (“small batch”)

Automated tests on every commit

- Automation is good
- Consistent
- Easy (once it is setup)

Catch errors and problems immediately

- While you still remember the code
- Before you have to change lots of code



Continuous Integration Challenges

Culture

Infrastructure and Tools

Project management

CI Challenges: Culture

Frequent commits:

- Many developers are used to long coding sessions, and a check-in when it is “done”
- Each “commit” is an *integration*
  - Needs to build and work
  - Less branching, merging and code conflicts?

Immediate automated testing:

- Functional tests
- Regression tests
- Code and security standards
- No one likes tests

Continuous Integration at the CSL

Application-specific testing

- Unit tests, functional tests, regression tests, etc

One-shot/single-purpose quality and basic checks:

- Syntax checks, enforce the basics

Code Inspection and Analysis: SonarQube

Continuous Integration: Puppet

Yamllint with custom configuration for Puppet node files

Puppet-lint checks puppet configuration

Run on every commit

Won’t deploy configuration to Production unless passes with no errors

Simple Code is Better Code

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

– Brian W. Kernighan

The Elements of Programming Style, 2nd edition, Chapter 2

Python: Pre-Commit hooks

Pre-commit-hooks framework:

- trailing-whitespace
- end-of-file-fixer
- check-docstring-first
- check-json
- check-yaml
- debug-statements
- requirements-txt-fixer

Additional hooks:

- Flake8
- autopep8
- pyupgrade
- reorder_python_imports
- Add-trailing-comma
- bandit
- check-hooks-apply
- check-useless-excludes

SonarQube: Continuous Code Quality

- 25+ Languages
- Integrates with git and other SCMs
- Database stores quality snapshots, history
- **Issues and issue status remembered between analysis runs**
- Adjust Rules to fit your project, organization, etc
- Project Dashboard
- Community edition

SonarQube: Code Analysis and Issues

- **Code Smells**: Code quality and maintainability
- **Bugs**: Reliability
- **Vulnerabilities**: Security issues
- **Security Hotspots**: Security-sensitive code, require review by designated staff

SonarQube: Issue Lifecycle

- Open
- Confirmed
- Resolved
- Reopened
- Closed

SonarQube: Issue Resolution

Closed (Automatic):

- Fixed: when issue not found in later analysis
- Removed: if rule removed from quality profile

Resolved (Manual):

- False Positive
- Won’t FIx

SonarQube Example: Cognitive Complexity

Cognitive Complexity Score: How hard it is to **understand** the code's control flow

- Count breaks in the linear flow of the code
- Count nesting of flow-breaking structures
- Don’t penalize code that is a good practice (shorthand and structures that make the code more readable)

SonarQube Example: Cognitive Complexity

- Code flagged for Cognitive Complexity
- Developer said that what it had to do was complicated, so complexity was expected and OK
- Reviewed the code ...

SonarQube Example: Cognitive Complexity

Reviewed complex code:

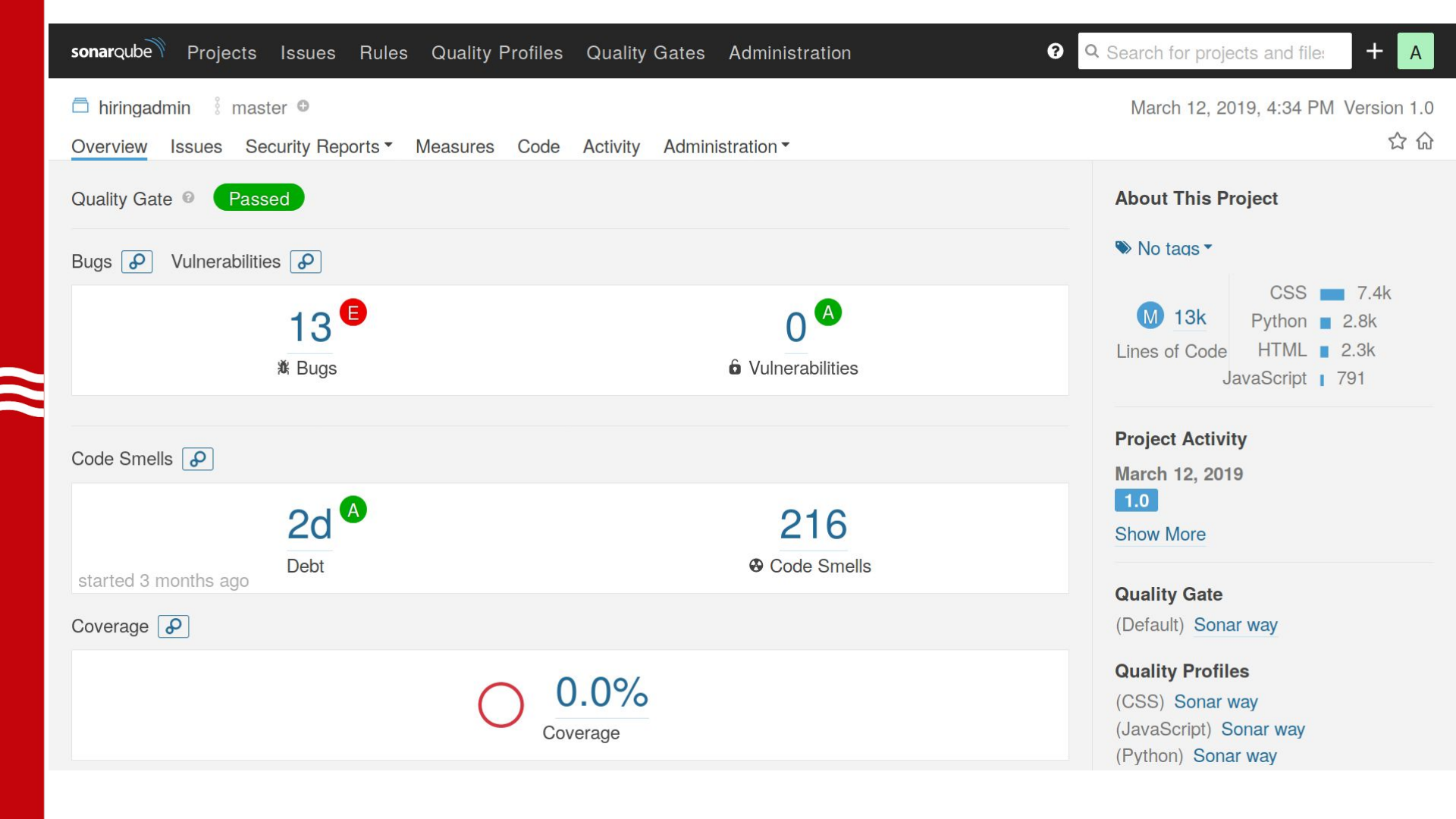
- Creating/Provisioning a new user
- 7 distinct steps/tasks
- Lots of duplicated code, complicated error handling

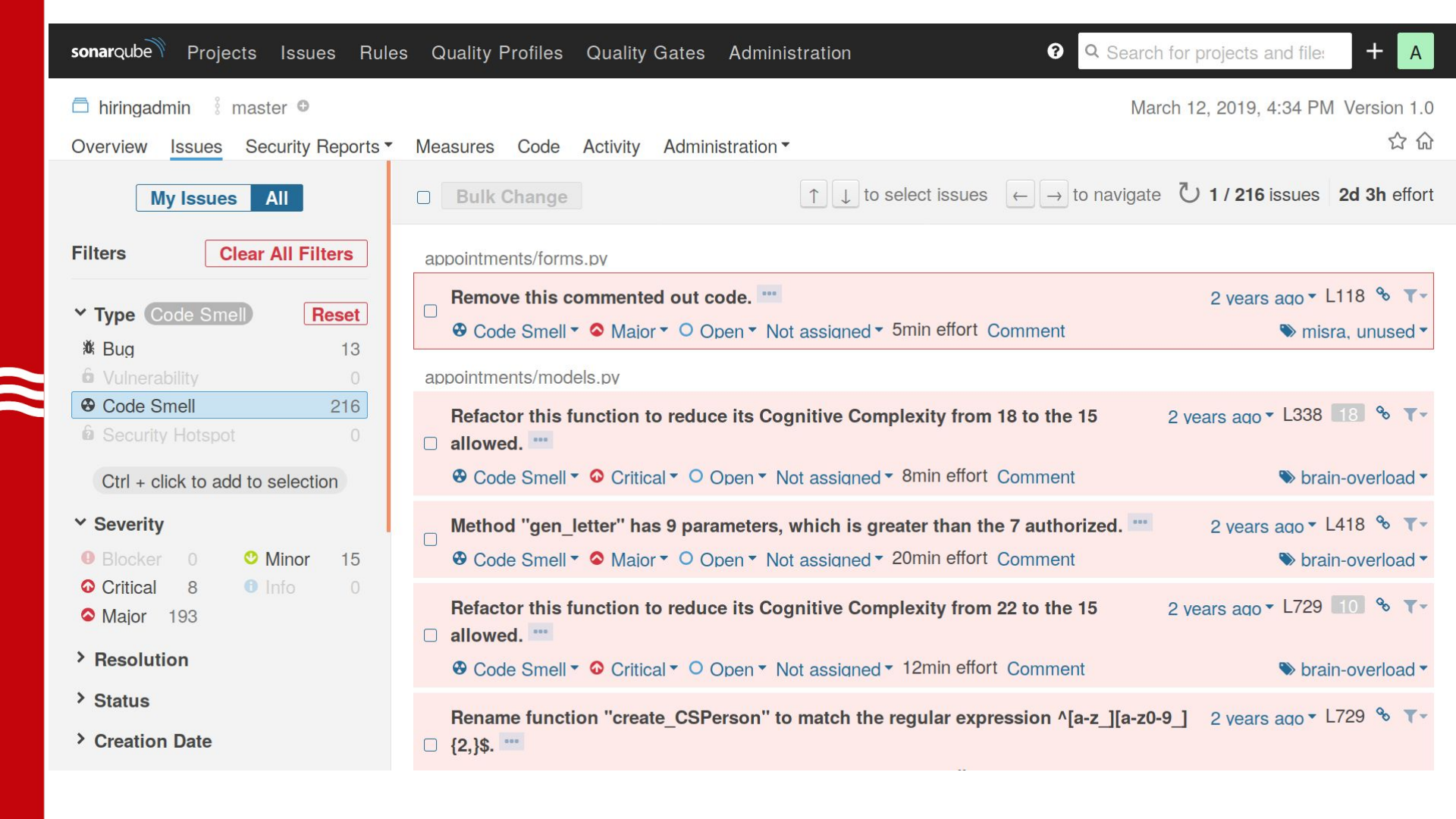
Refactored code:

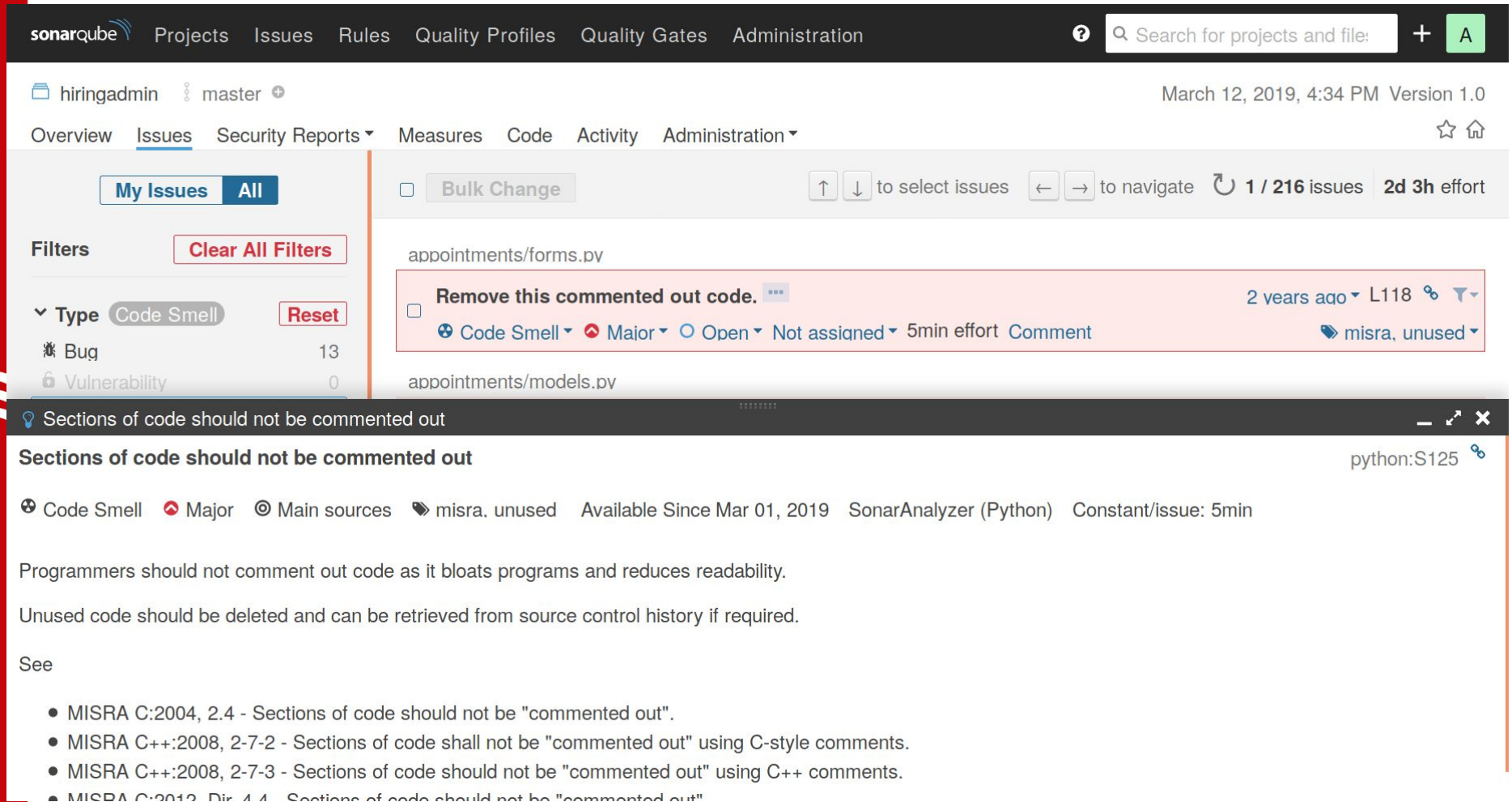
- Simplified code into one Python `try:` clause
- Only two exception cases to handle: Fatal and Retry

Significant reduction in Cognitive Complexity score

Significant improvement and simplification of the code


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects and file
A
hiringadmin
master
March 12, 2019, 4:34 PM Version 1.0
Overview
Issues
Security Reports
Measures
Code
Activity
Administration
Quality Gate
Passed
Bugs
Vulnerabilities
13
E
Bugs
0
A
Vulnerabilities
Code Smells
2d
A
Debt
started 3 months ago
216
Code Smells
Coverage
0.0%
Coverage
About This Project
No tags
M 13k
Lines of Code
CSS 7.4k
Python 2.8k
HTML 2.3k
JavaScript 791
Project Activity
March 12, 2019
1.0
Show More
Quality Gate
(Default) Sonar way
Quality Profiles
(CSS) Sonar way
(JavaScript) Sonar way
(Python) Sonar way


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects and file
A
hiringadmin
master
March 12, 2019, 4:34 PM Version 1.0
Overview
Issues
Security Reports
Measures
Code
Activity
Administration
My Issues
All
Filters
Clear All Filters
Type
Code Smell
Reset
Bug 13
Vulnerability 0
Code Smell 216
Security Hotspot 0
Ctrl + click to add to selection
Severity
Blocker 0
Minor 15
Critical 8
Info 0
Major 193
Resolution
Status
Creation Date
Bulk Change
to select issues
to navigate
1 / 216 issues
2d 3h effort
appointments/forms.py
Remove this commented out code.
2 years ago L118
Code Smell Major Open Not assigned 5min effort Comment
misra, unused
appointments/models.py
Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
2 years ago L338 18
Code Smell Critical Open Not assigned 8min effort Comment
brain-overload
Method "gen_letter" has 9 parameters, which is greater than the 7 authorized.
2 years ago L418
Code Smell Major Open Not assigned 20min effort Comment
brain-overload
Refactor this function to reduce its Cognitive Complexity from 22 to the 15 allowed.
2 years ago L729 10
Code Smell Critical Open Not assigned 12min effort Comment
brain-overload
Rename function "create_CSPerson" to match the regular expression ^[a-z_][a-z0-9_]{2,}$.
2 years ago L729


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects and file
A
hiringadmin
master
March 12, 2019, 4:34 PM Version 1.0
Overview
Issues
Security Reports
Measures
Code
Activity
Administration
My Issues
All
Bulk Change
to select issues
to navigate
1 / 216 issues
2d 3h effort
Filters
Clear All Filters
Type
Code Smell
Reset
Bug
13
Vulnerability
0
appointments/forms.py
Remove this commented out code.
2 years ago
L118
Code Smell
Major
Open
Not assigned
5min effort
Comment
misra, unused
appointments/models.py
Sections of code should not be commented out
Sections of code should not be commented out
python:S125
Code Smell
Major
Main sources
misra, unused
Available Since Mar 01, 2019
SonarAnalyzer (Python)
Constant/issue: 5min
Programmers should not comment out code as it bloats programs and reduces readability.
Unused code should be deleted and can be retrieved from source control history if required.
See
MISRA C:2004, 2.4 - Sections of code should not be "commented out".
MISRA C++:2008, 2-7-2 - Sections of code shall not be "commented out" using C-style comments.
MISRA C++:2008, 2-7-3 - Sections of code should not be "commented out" using C++ comments.

In Conclusion...

Continuous Integration takes some setup, but it’s worth it

Improve code quality and consistency

Allows for Continuous Deployment, or at least faster deployment

Linters and other single-purpose checkers are great

SonarQube brings a lot of tools and more sophisticated analysis

Questions?

Thanks to former CSL student employee Marco Carini for CI setup and deploying as many CI hooks as he could find.

Also thanks to the CSL staff for adopting CI and assisting with this presentation.