Change Impact Detector Feature Survey

A version control system is a must for big software projects. It can help developers to manage the source code by keeping track of all the code changes. They also protect source code from unintentional human-based errors and their potential impacts with the pull request mechanism. Code reviewers can look at the pull request changes and see the obvious errors. However, some errors can be hard to identify, and in-depth analysis may be needed to detect them. Change impact analysis is a way to estimate the possible effects of proposed changes which can also be used to detect such errors. Change impact analysis can significantly reduce the cost and time needed to maintain the project. The analysis can be done on different granularities. This project's analysis granularity is the pull request. Our project calculates the risk score of pull requests which represents the potential impact of a pull request on other parts of the system, by creating a call graph and analyzing each changed pair with predetermined metrics.

This form was created to get your opinion on features of our senior year project, Change Impact Detector.

* Zorunlu soruyu belirtir

1. E-posta *

____

2. How many years of experience do you have in software development? *

Yalnızca bir şıkkı işaretleyin.

○ 1 - 3 years

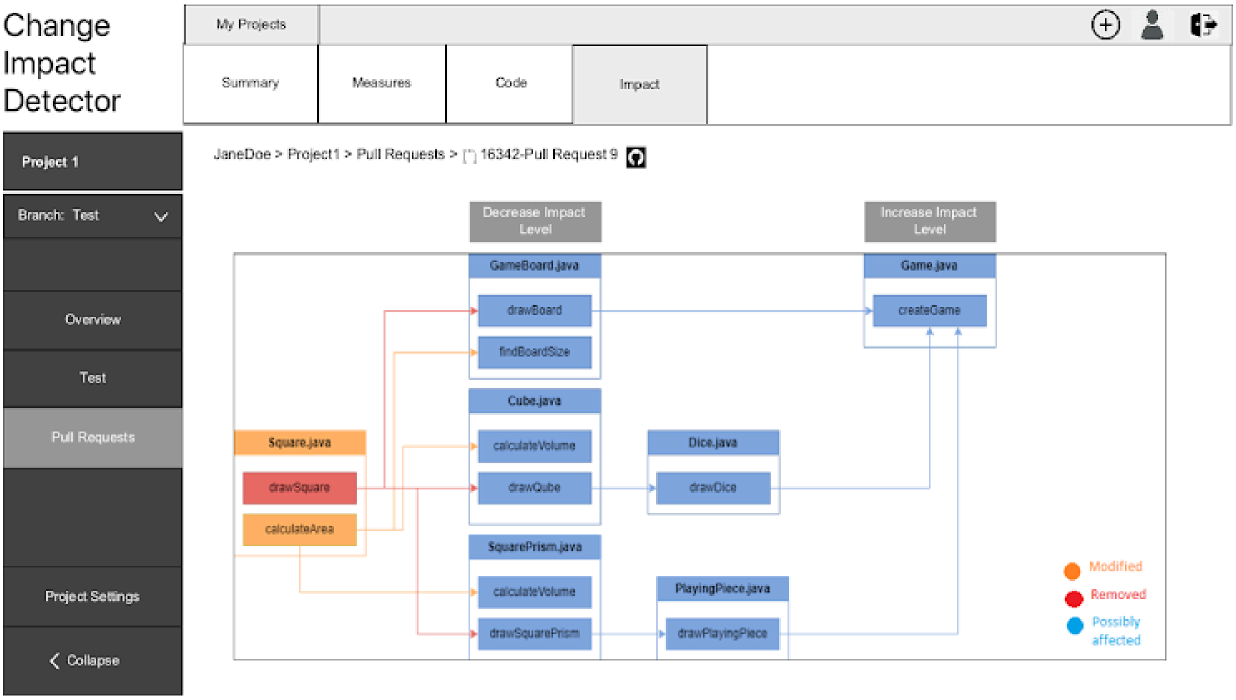
○ 3 - 5 years

○ 5 - 10 years

○ 10 - 15 years

○ 15 + years

3. The call graph feature visualizes the potential effects of the pull request after it is merged. It can guide users by highlighting different layers of change impact with relations between caller-callee functions. We believe it will be beneficial for our users to create insights. Would you want to use such a feature? *

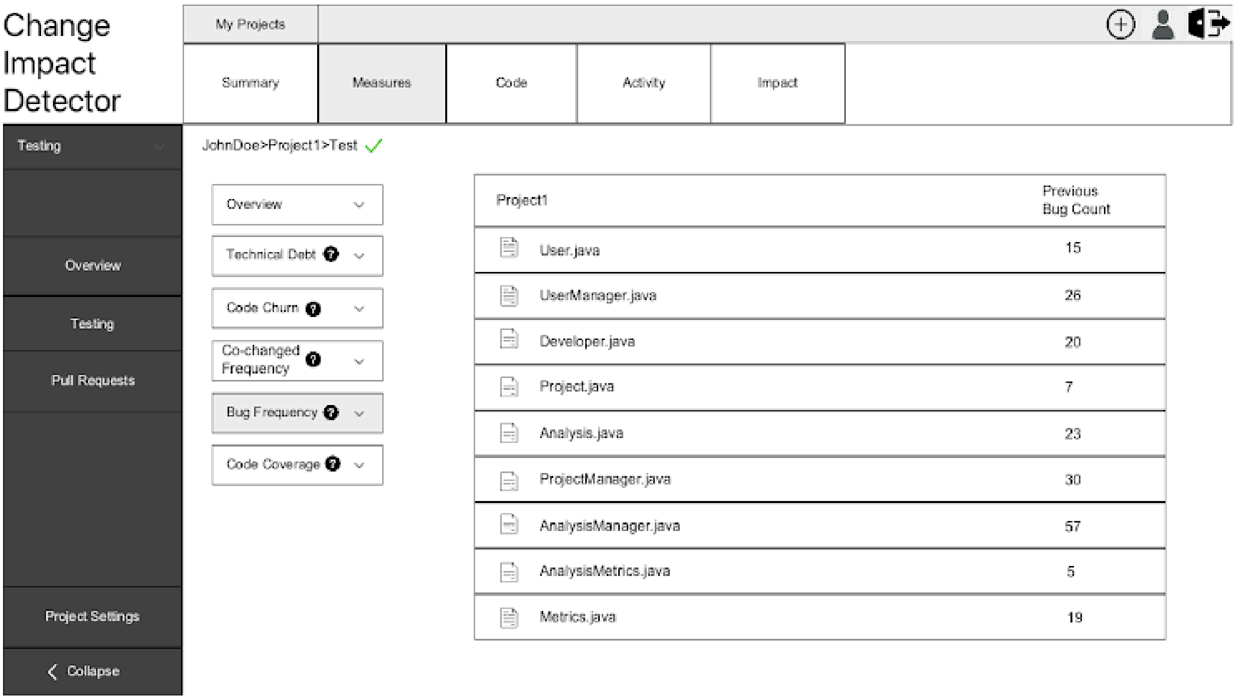


Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

4. What do you think about call graph feature? How can we improve this?

5. The bug frequencies feature calculates associated bug count per file in a pull request since the beginning of the project. It lets users to see bug prone files in the change set so that they can foresee which changes can trigger old bugs. Would you want to use such a feature? *



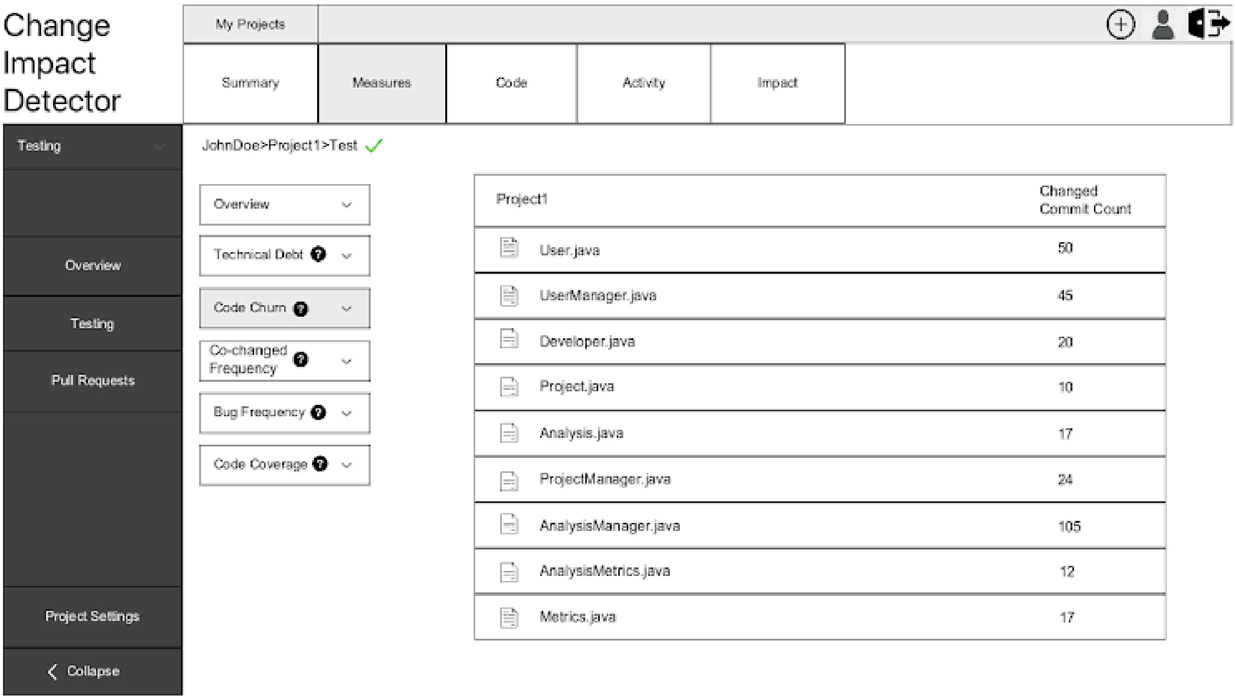
Project1	Previous Bug Count
User.java	15
UserManager.java	26
Developer.java	20
Project.java	7
Analysis.java	23
ProjectManager.java	30
AnalysisManager.java	57
AnalysisMetrics.java	5
Metrics.java	19

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

6. What do you think about bug frequencies feature? How can we improve this?

7. The code churn feature represents how many times a code piece is edited. If a code is changed a lot of times, it is more susceptible to bugs. Would you want to use such a feature? *



Project1	Changed Commit Count
User.java	50
UserManager.java	45
Developer.java	20
Project.java	10
Analysis.java	17
ProjectManager.java	24
AnalysisManager.java	105
AnalysisMetrics.java	12
Metrics.java	17

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

8. What do you think about code churn feature? How can we improve this?

____

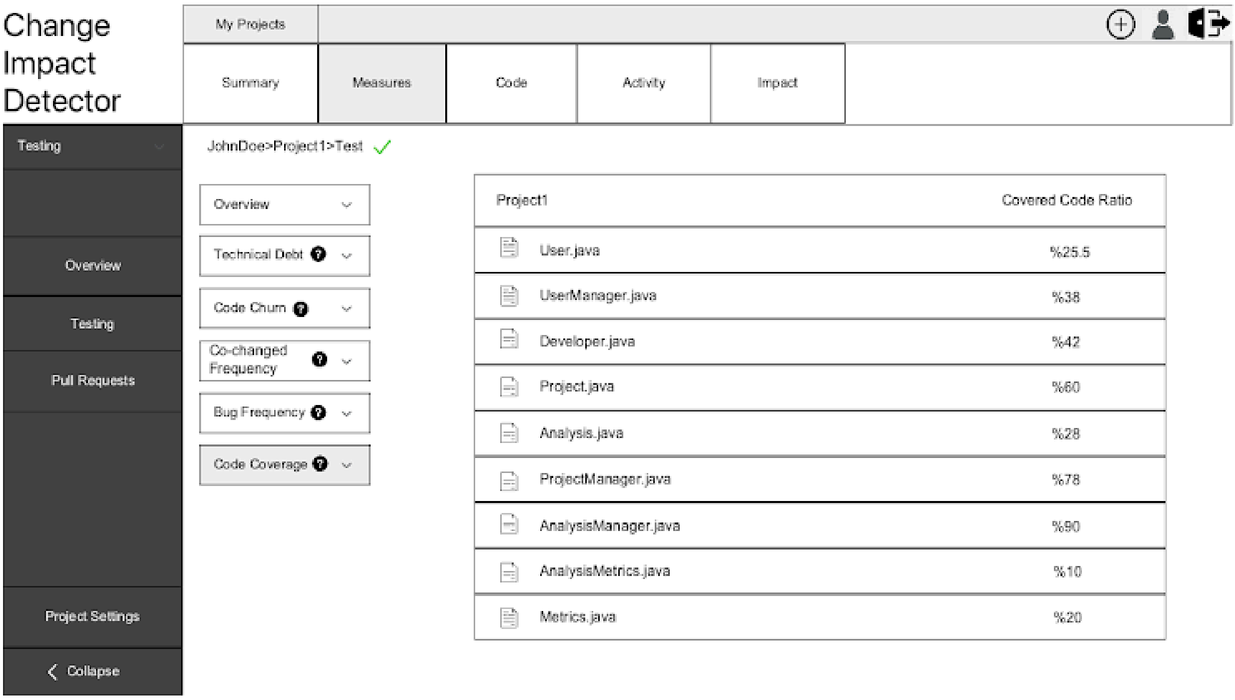
____

____

____

____

9. Code coverage is a software testing metric that determines the number of lines of code that is validated under a test procedure. Would you want to use such a feature? *



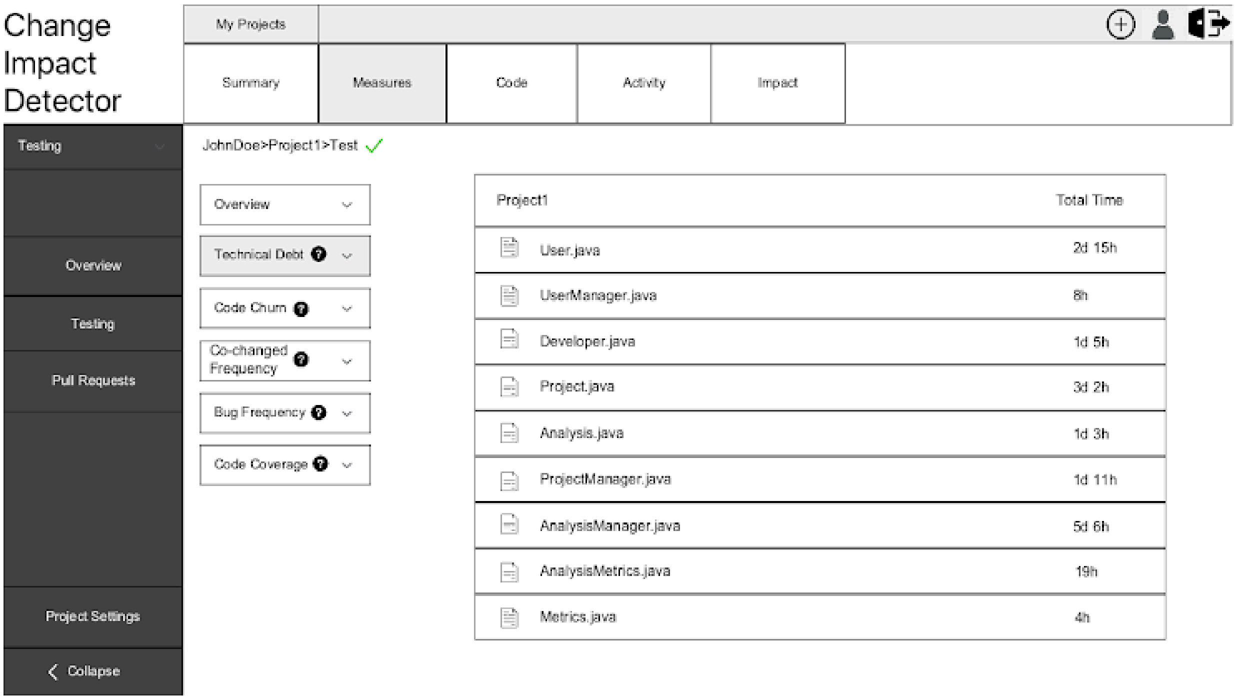
Project1	Covered Code Ratio
User.java	%25.5
UserManager.java	%38
Developer.java	%42
Project.java	%60
Analysis.java	%28
ProjectManager.java	%78
AnalysisManager.java	%90
AnalysisMetrics.java	%10
Metrics.java	%20

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

10. What do you think about code coverage? How can we improve this?

11. Technical debt feature is the measure of compromise in the quality of code done to make up for quick delivery schedules. This lets user to see the amount of time needed to solve code smells. It's related to maintainability rather than security or cost. Would you want to use such a feature? *



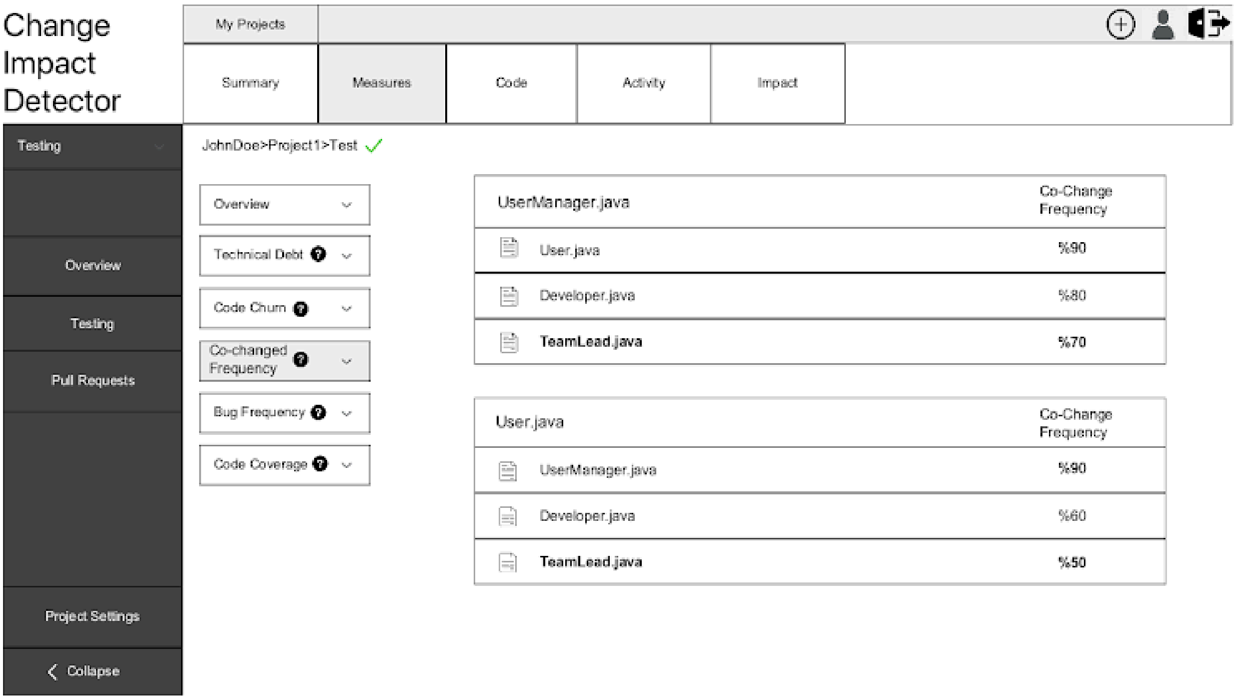
Project1	Total Time
User.java	2d 15h
UserManager.java	8h
Developer.java	1d 5h
Project.java	3d 2h
Analysis.java	1d 3h
ProjectManager.java	1d 11h
AnalysisManager.java	5d 6h
AnalysisMetrics.java	19h
Metrics.java	4h

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

12. What do you think about technical debt feature? How can we improve this?

13. Some files can be logically coupled so that if one of them changes, the other one should also change. These are called co-chancing files. Would you want to see such relationships? *



UserManager.java	Co-Change Frequency
User.java	%90
Developer.java	%80
TeamLead.java	%70

User.java	Co-Change Frequency
UserManager.java	%90
Developer.java	%60
TeamLead.java	%50

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

14. What do you think about co-changing files feature? How can we improve this?

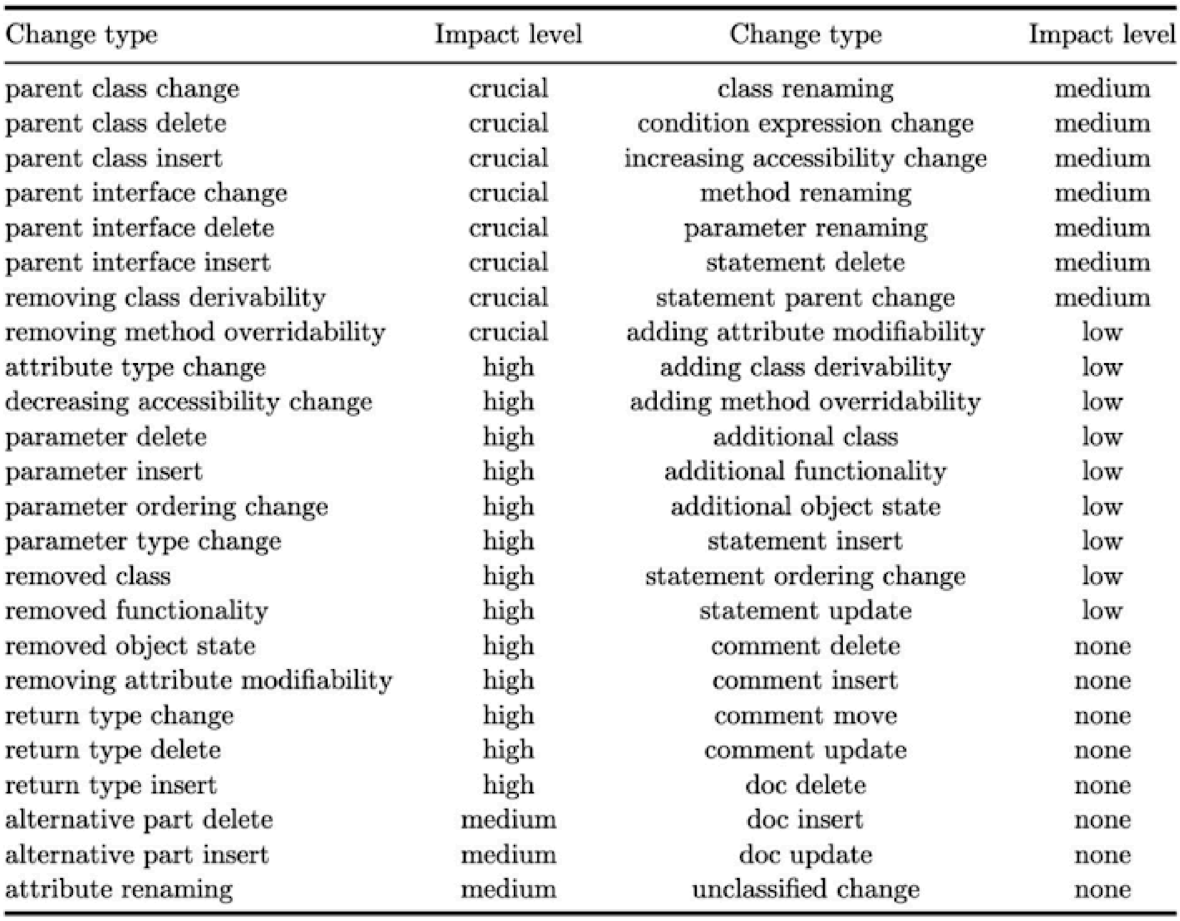
15. Risk score represents the magnitude of the change impact of a pull request. What type of metrics should be considered for risk score calculation? *

Uygun olanların tümünü işaretleyin.

□ Code coverage
□ Technical debt
□ Bug frequencies of the changed files
□ Co-chancing files
□ Code churn of the changed files
□ Complexity of the changed files
□ Inheritance between changed files and the other files
□ Coupling between changed files and the other files
□ Cohesion between changed files and the other files

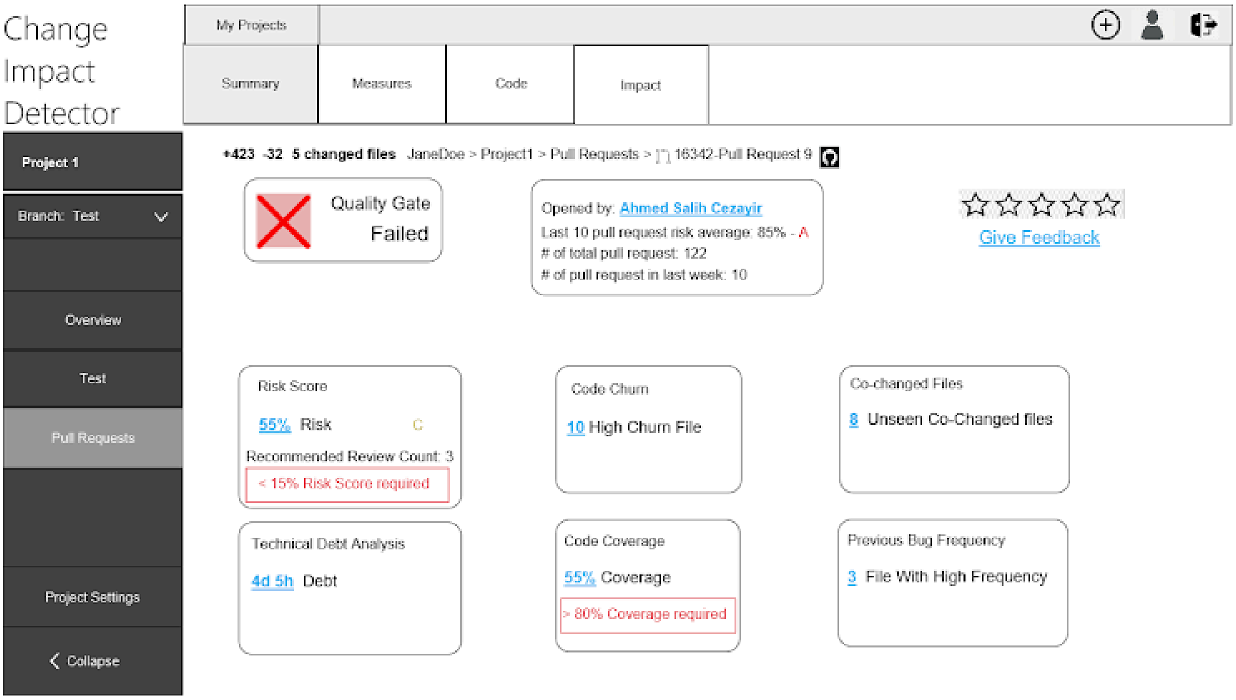
□ Diğer: ______________________________

16. Alternatively, change types can be determined to calculate change impact. Do you think it should be considered as a metric? If yes, is it more important than previous ones? *



Change type	Impact level	Change type	Impact level
parent class change	crucial	class renaming	medium
parent class delete	crucial	condition expression change	medium
parent class insert	crucial	increasing accessibility change	medium
parent interface change	crucial	method renaming	medium
parent interface delete	crucial	parameter renaming	medium
parent interface insert	crucial	statement delete	medium
removing class derivability	crucial	statement parent change	medium
removing method overridability	crucial	adding attribute modifiability	low
attribute type change	high	adding class derivability	low
decreasing accessibility change	high	adding method overridability	low
parameter delete	high	additional class	low
parameter insert	high	additional functionality	low
parameter ordering change	high	additional object state	low
parameter type change	high	statement insert	low
removed class	high	statement ordering change	low
removed functionality	high	statement update	low
removed object state	high	comment delete	none
removing attribute modifiability	high	comment insert	none
return type change	high	comment move	none
return type delete	high	comment update	none
return type insert	high	doc delete	none
alternative part delete	medium	doc insert	none
alternative part insert	medium	doc update	none
attribute renaming	medium	unclassified change	none

17. Recommended reviewer count is determined by the risk score of a pull request. The threshold for the metric can be determined before the analysis. Would you want to see such a recommendation? *



+423 -32 5 changed files JaneDoe > Project1 > Pull Requests > 16342-Pull Request 9

Quality Gate Failed

Opened by: Ahmed Salih Cezayir
Last 10 pull request risk average: 85% - A
# of total pull request: 122
# of pull request in last week: 10

Risk Score
55% Risk C
Recommended Review Count: 3
< 15% Risk Score required

Code Churn
10 High Churn File

Co-changed Files
8 Unseen Co-Changed files

Technical Debt Analysis
4d 5h Debt

Code Coverage
55% Coverage
> 80% Coverage required

Previous Bug Frequency
3 File With High Frequency

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Defi	○	○	○	○	○	Totally would

18. What do you think about recommended reviewer count feature? How can we improve this?

19. Do you currently analyze pull request's change impact in your projects? If yes, how do you analyze it? *

20. All of the above features are related to change impact analysis. So, do you think change impact analysis for pull requests is important? *

Yalnızca bir şıkkı işaretleyin.

	1	2	3	4	5	
Not	○	○	○	○	○	Pretty important