Static Code Analysis Tool ——sonarLint

Advised by Prof.Qin Liu

Group 16:
2151409 Yuntao Hu
2152085 Yifei Sun
2054099 Jieying Ye
2152193 Yixin Li

CONTENTS

01

Background

01Background

SonarLint

- SonarLint was created by SonarSource in 2014 as an open-source code analysis tool for multiple programming languages. It aimed to provide configurable and extensible capabilities, unlike existing tools at the time.The tool was designed to integrate with IDEs, offering real-time feedback on code quality and potential issues. Developers can define custom rules or use pre-configured rulesets for their projects.
- Today, SonarLint is widely adopted, supporting various languages and providing comprehensive code analysis, including checks for code smells, security vulnerabilities, and duplication.

01Background

SonarQube

- SonarQube was created by SonarSource in 2008 as an open-source platform for continuous code quality management. It aimed to provide a comprehensive solution for analyzing code quality, tracking technical debt, and improving software project health.
- SonarQube offers centralized code analysis, supporting multiple programming languages and allowing organizations to define custom rules and quality gates. Today, it is widely used, supporting over 25 languages and integrating with various development tools and CI/CD pipelines.

02

Objectives

02Objectives

01

Code Quality Inspection

02

Coding Standards Enforcement

03

Security Vulnerability Detection

04

Technical Debt Management

05

Team Collaboration and Visibility

03

Functionalities

03Functionalities

Code Analysis

- Both tools provide comprehensive code analysis capabilities, supporting a wide range of programming languages.
- They can identify code issues, such as bugs, code smells, security vulnerabilities, and code duplications.

Rule Management

- Both tools allow users to define and customize their own set of rules for code quality and coding standards.
- Users can also use pre-configured rule sets provided by the tools or the community.

Real-time Feedback

- Sonarlint integrates directly with the developer's IDE, providing real-time feedback on code quality issues.
- SonarQube can be configured to analyze code changes and provide feedback within the development workflow.

03Functionalities

Reporting and Dashboards

- SonarQube offers detailed reporting and dashboards that provide visibility into the overall code quality and technical debt of a project.
- These reports can be used to track progress, identify trends, and make informed decisions about code quality improvements.

Integration with Development Tools

- Both Sonarlint and SonarQube integrate with a wide range of development tools, such as version control systems, build tools, and IDEs.
- This allows for seamless integration of code analysis into the development lifecycle.

04

Features

04Features

SonarLint

- Custom Rules: Similar to ESLint, SonarLint allows users to customize rules according to their own needs to ensure code compliance with specific coding standards and best practices. Users can add, modify, and delete rules, and customize the check method and content.
- Extensibility: SonarLint provides a pluggable plugin system that allows developers to customize the functionality and rules of the tool according to their needs. This enables users to adapt the tool to specific development requirements and code conventions.
- Supported Languages: SonarLint supports a wide range of programming languages, including JavaScript, TypeScript, JSX, Vue, and the latest ECMAScript standards. This allows developers to use the tool to analyze code written in the latest language features.

SonarQube

- Reporting and Output Formats: Like ESLint, SonarQube supports multiple output formats, including HTML, JSON, and others. This allows developers to integrate the tool with other systems or present the analysis results in a format that is easily understandable by team members.
- Centralized Code Analysis: SonarQube provides a centralized platform for continuous code quality management, offering comprehensive code analysis, issue tracking, and reporting capabilities.
- Integration with Development Workflow: SonarQube is designed to integrate with various development tools and CI/CD pipelines, enabling seamless integration of code analysis into the development lifecycle.

05

Usage&Experiment &Analysis

05Usage&Experiment&Analysis

(1)IntelliJ IDEA 2023/Java

Usage

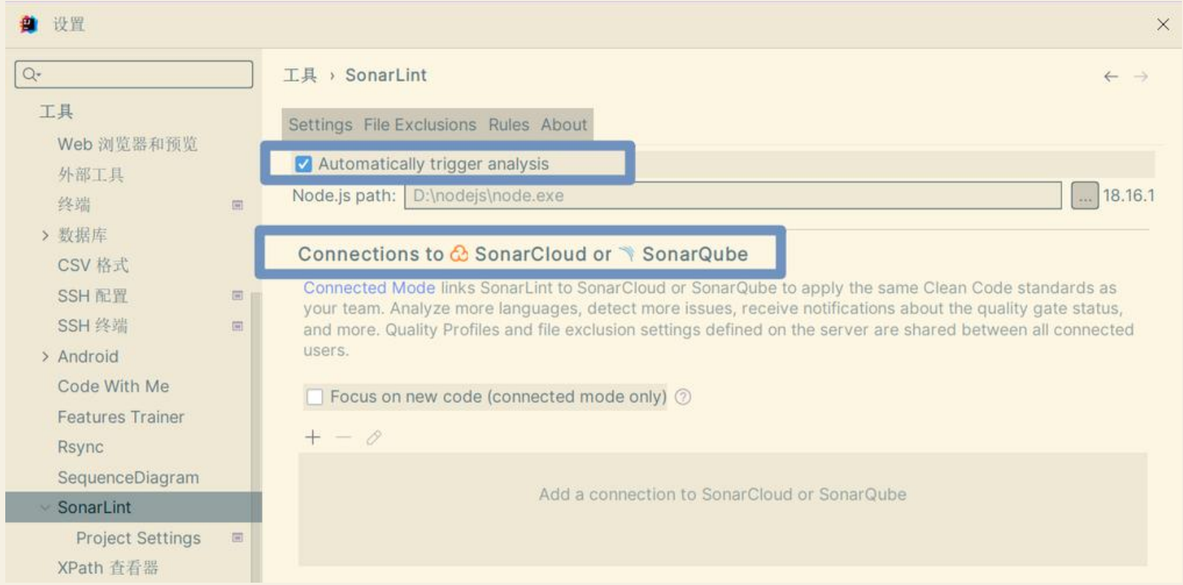
Experiments

Result Analysis

1. Install the SonarLint plugin.

2.Configure the overall settings. We can set it to automatically trigger code checks.

We also recommend connecting to SonarQube.



[图片]

05Usage&Experiment&Analysis

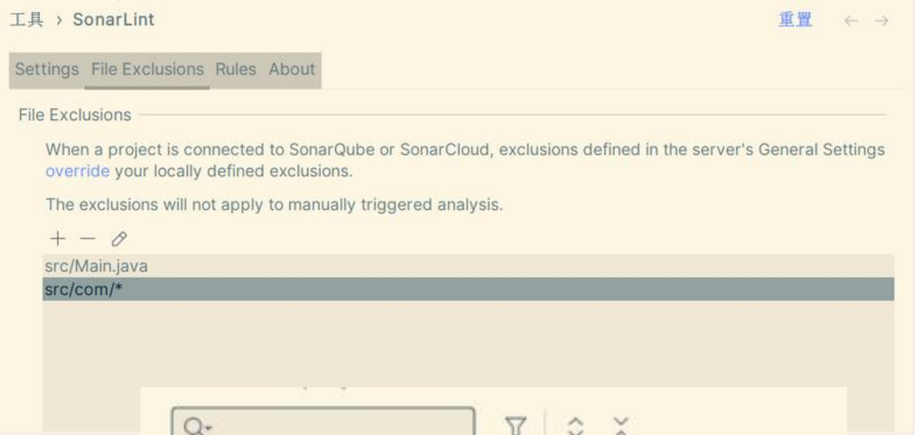
(1)IntelliJ IDEA 2023/Java

Usage

Experiments

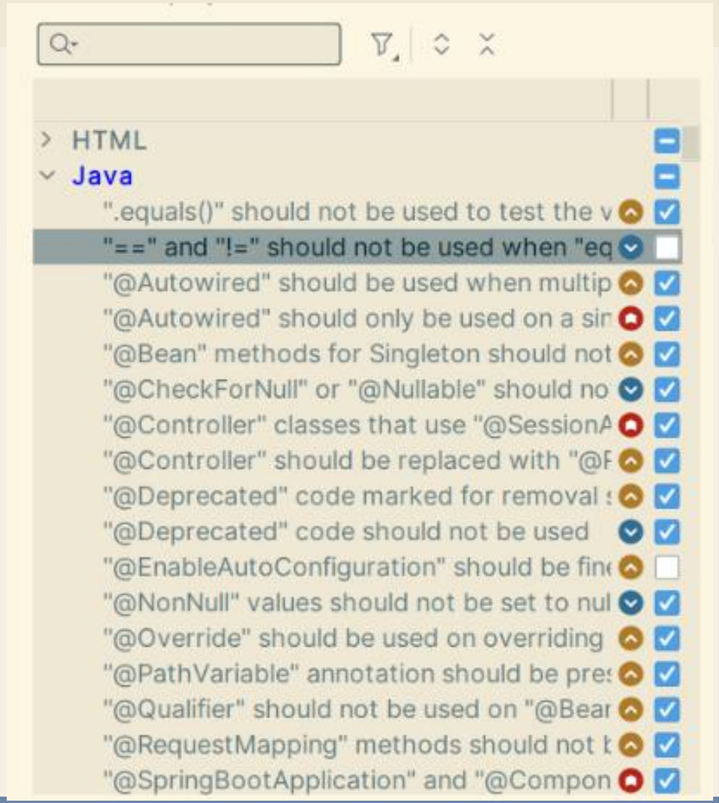
Result Analysis

- We have the option to exclude certain files from code checks.



- We can enable or disable specific rules.

- Additionally, each rule is categorized by levels that describe their impact on code reliability, with red indicating the highest impact.



[图片]

05Usage&Experiment&Analysis

(1)IntelliJ IDEA 2023/Java

Usage

Experiments

Result Analysis

We can utilize the shortcut Ctrl+Shift+S to activate this tool.

SonarLint has found 394 issues in this project.

05Usage&Experiment&Analysis

(1)IntelliJ IDEA 2023/Java

Usage

Experiments

Result Analysis

SonarLint facilitates pinpointing the location of issues and provides detailed explanations.

Upon code modification, this warning turns to grey, indicating real-time updates.

05Usage&Experiment&Analysis

(1)IntelliJ IDEA 2023/Java

Usage

Experiments

Result Analysis

SonarLint can provide useful recommendations for each issue.

Rule Locations

String literals should not be duplicated

Adaptability issue | not distinct Maintainability java:S1192 Learn more

Why is this an issue? How can I fix it?

Compliant solution

```
private static final String ACTION_1 = "action1";  // Compliant

public void run() {
  prepare(ACTION_1);                                // Compliant
  execute(ACTION_1);
  release(ACTION_1);
}
```

Parameters

threshold 3

Parameter values can be set in Rule Settings. In connected mode, server side configuration overrides local settings.

05Usage&Experiment&Analysis

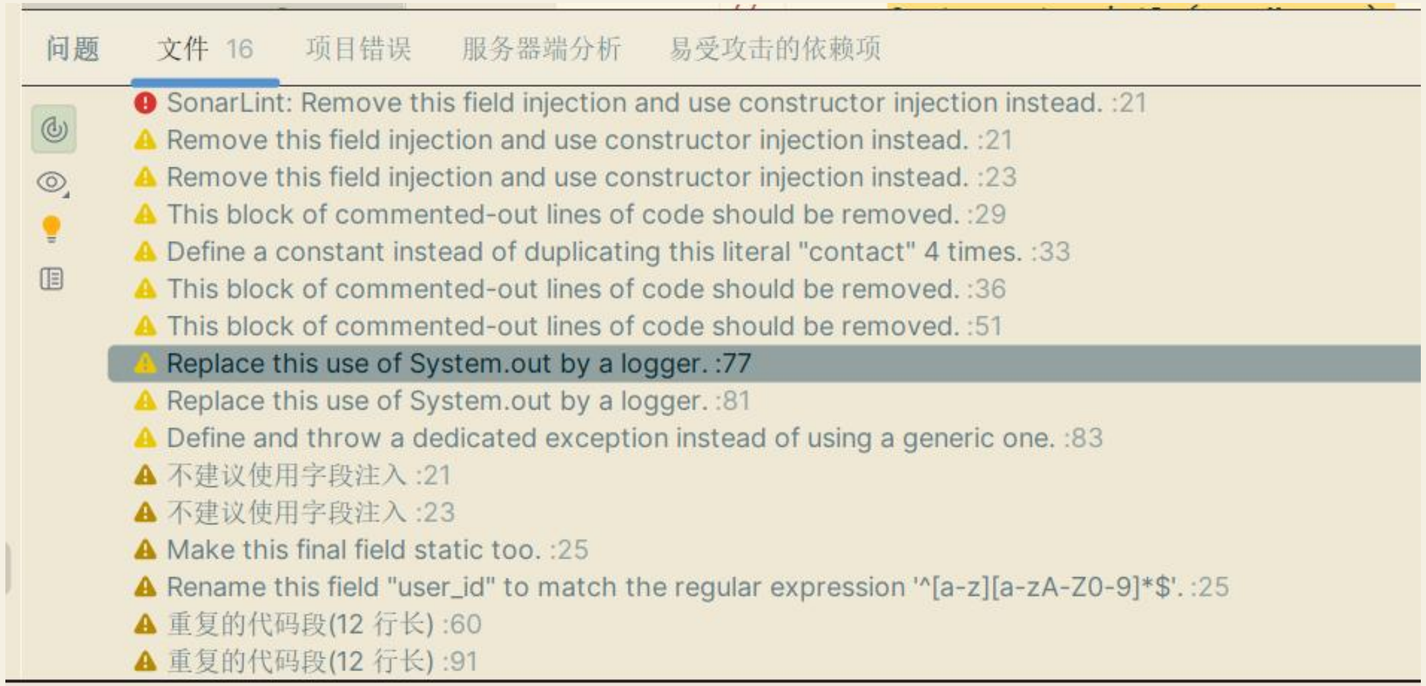
(1)IntelliJ IDEA 2023/Java

Usage

Experiments

Result Analysis

When comparing SonarLint with the analysis tool built into IDEA, we found there are many common points. For instance, they both suggested changing 'System.out' to log output and emphasized the proper use of '@Autowired'.



05Usage&Experiment&Analysis

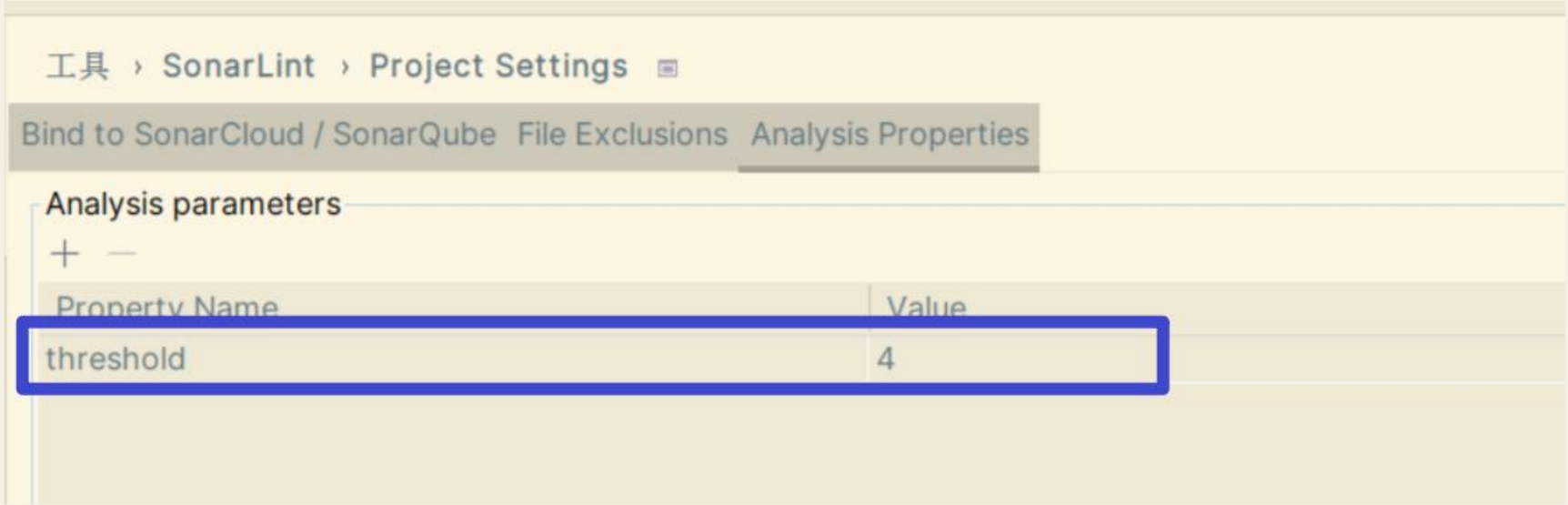
(1)IntelliJ IDEA 2023/Java

Usage

Experiments

Result Analysis

We think the advantages of SonarLint lie in its customizable analysis parameters, users have the flexibility to choose the granularity of error detection.



05Usage&Experiment&Analysis

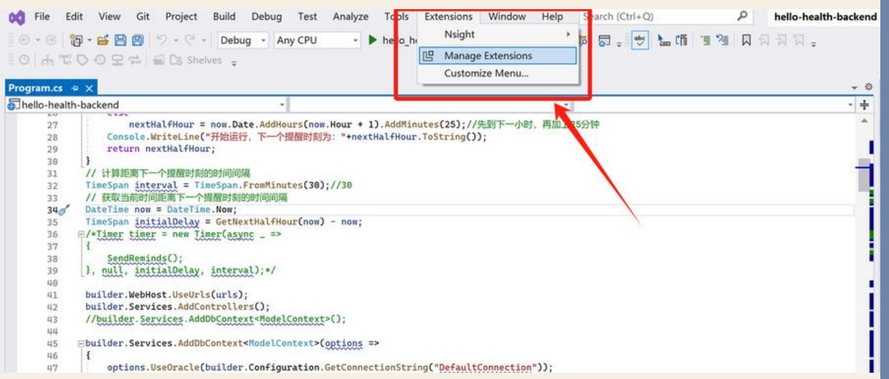
(2)Visual Studio 2022/C# ASP.NET

Usage

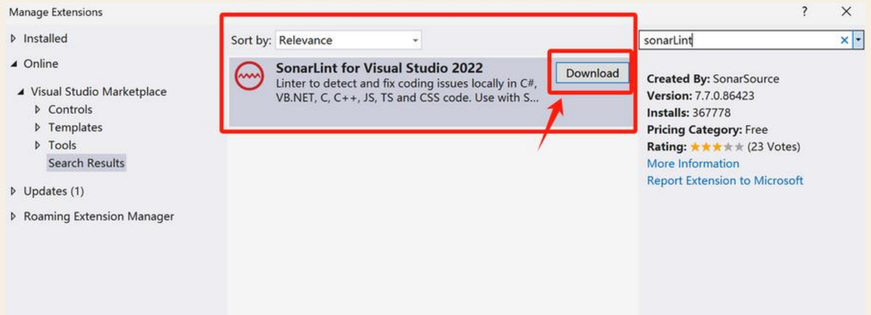
Experiments

Result Analysis

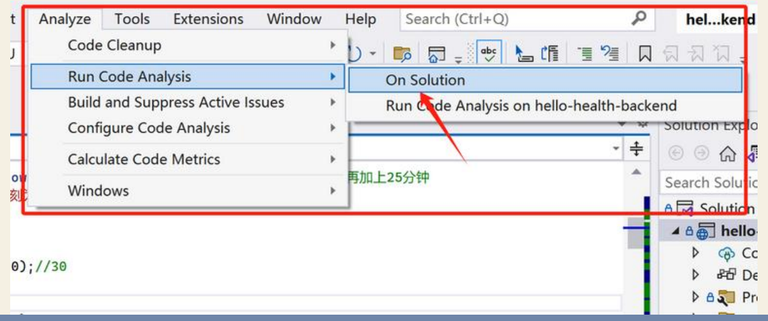
1.Open the VS2022 development tool, click "Extensions" -> "Manage Extensions".



2.Click "Online" -> "Visual Studio Marketplace", enter SonarLint in the search box, and then click download.



3.Click "Run Code Analysis" -> "On solution" and it begins to analyze.



05Usage&Experiment&Analysis

(2)Visual Studio 2022/C# ASP.NET

Usage

Experiments

Result Analysis

This project involves developing the backend of a database-driven application using C# ,ASP.NET within the Visual Studio IDE.

In this code, ASP.NET Core is utilized to create API controllers and route requests.

Entity Framework Core is an Object-Relational Mapping (ORM) framework used for interacting with databases.
In this code, Entity Framework Core is employed to access the database and perform database-related operations.

Static Code Analysis Focus Areas

- Code Quality and Maintainability
- Security(Authentication and Authorization,Data Exposure)
- Code Style and Conventions
- Exception Handling

05Usage&Experiment&Analysis

(2)Visual Studio 2022/C# ASP.NET

Usage

Experiments

Result Analysis

SonarLint lists 355 warnings.

```
namespace hello_health_backend.Controller
{
    4 references
    public class administrator
    {
        //管理员ID
        2 references
        public int? id { get; set; }
```

S101 Rename class 'administrator' to match pascal case naming rules, consider using 'Administrator'. hello-health-back

Naming conventions

It emphasizes that we should follow Pascal case naming rules.It is reasonable.

It is essential to ensure code follows consistent naming conventions.

```
//创建一个类，负责返回药品信息
11 references
public class Medicine_data
{
    4 references
    public string medicine_id { get; set; }//批准文号
    4 references
    public string? medicine_ch_name { get; set; }//中文名称
    4 references
    public string? medicine_en_name { get; set; }//英文名称
    12 references
    public string? medicine_category { get; set; }//分类
```

CS8618 Non-nullable property 'medicine_id' must contain a non-null value when exiting constructor. Consider declaring the property as hello-health-backend

Complete declarations, about Non-nullable and nullable

However, I think this suggestion is unreasonable.

The purpose of declaring properties as non-nullable is to ensure data integrity and logical consistency, enforcing this rule to avoid runtime errors due to null references. Thus, the suggestion to declare such properties as nullable seems to contradict the original intention of maintaining strong type safety and ensuring data completeness.

05Usage&Experiment&Analysis

(2)Visual Studio 2022/C# ASP.NET

SonarLint lists 355 warnings.

Usage

Experiments

Result Analysis

```
if (now.Minute < 25)//25
    nextHalfHour = new DateTime(n
else if (now.Minute < 55)//55
    nextHalfHour = new DateTime(n
else
    nextHalfHour = now.Date.AddHo
```

nullable.

S6562 Provide the "DateTimeKind" when creating this object. hello-health

Overlooked coding standards

Provide the "DateTimeKind" when creating this object "Datetime".

It is reasonable, and very easy to overlook!

Specifying the DateTimeKind is crucial for correctly handling time zone conversions and for ensuring that the date and time values are interpreted correctly according to their intended time zone context.

```
[HttpPost("sendFlash")]
0 references
public async Task<string> Creat
{
```

CS1998 This async method lacks 'await' operators and will run synchronously. Consider using the 'await' operator to await non-blocking API calls, or 'await Task.Run(...)' to do CPU-bound work on a background thread.

Asynchronous programming

This warning highlights a common oversight in asynchronous programming where an async method does not actually await any asynchronous operations, causing it to execute synchronously instead. The reminder suggests using the await operator for non-blocking API calls or employing await Task.Run(...) for CPU-intensive tasks to ensure they run on a background thread.

It is very useful!

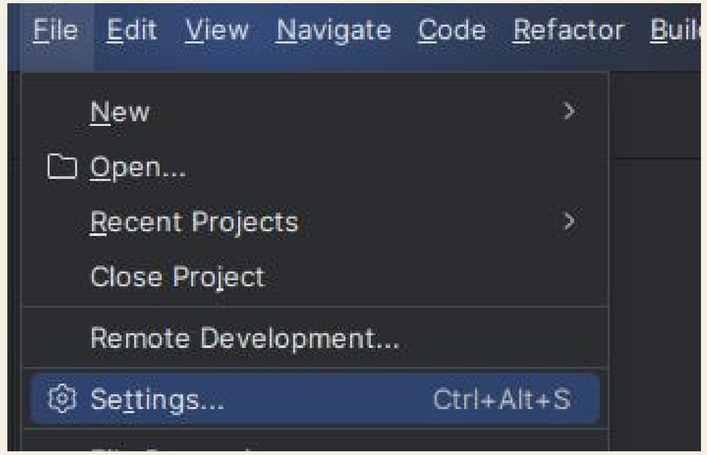
05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

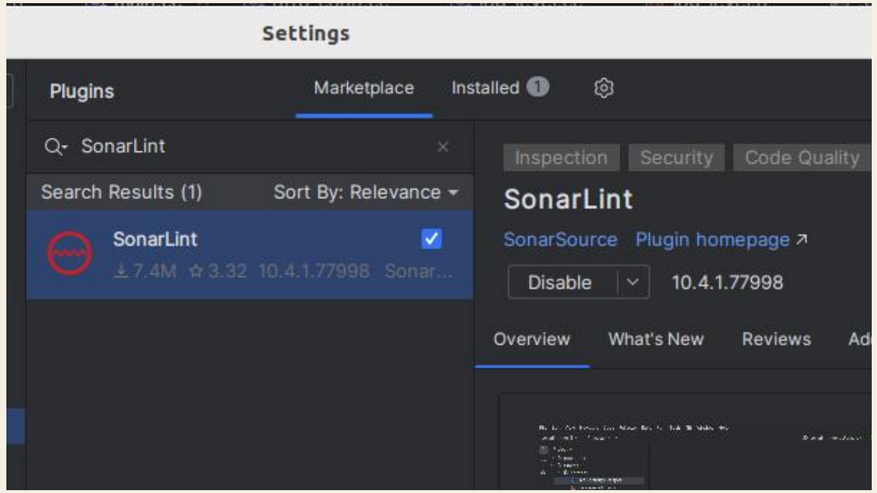
Usage

Experiments

Result Analysis



1. Open the CLion IDE,then click File->Settings->Plugin



2. In the setting dialog ,find the Plugin ,then In the Marketplace menu,search SonarLint. Eventually,you will find the plugin,then click install.

你的段落文字

05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

Usage

Experiments

Result Analysis

Initially,there're 168 issues in the project.

```
} 21 else if( ( sockfd == pipefd[0] ) 22 && ( events[i].events &

    int sig;
    char signals[1024];
    ret = recv( fd: pipefd[0], buf: signals, n: sizeof( signals ),
     23 if( ret == -1 ) {
        continue;
    }  24 else if( ret == 0 ) {
        continue;
    }  25 else  {
         26 for( int i = 0; i < ret; ++i ) {
             27 switch( signals[i] )  {
                case SIGALRM:
```

Too many nest and highly cognitive

- Cognitive complexity is incremented each time the code breaks
- Each nesting level adds a malus to the breaking call.
- Method calls are free

```
(5, 0) Global pointers should be const at every level. 27 minutes ago
(6, 0) Global pointers should be const at every level. 27 minutes ago
(7, 0) Global pointers should be const at every level. 27 minutes ago
(8, 0) Global pointers should be const at every level. 27 minutes ago
(9, 0) Global pointers should be const at every level. 27 minutes ago
(506, 25) implicit conversion loses integer precision: 'size_t' (aka 'uns
(513, 25) implicit conversion loses integer precision: 'size_t' (aka 'uns
(520, 24) implicit conversion loses integer precision: 'size_t' (aka 'uns
(539, 32) implicit conversion loses integer precision: '__off_t' (aka 'lor
(542, 32) implicit conversion loses integer precision: '__off_t' (aka 'lor
```

Global variables and type cast issues account for most

- Convenient but hard to understand.
- We'd better reason locally about the code and the variable.
- Global variables are often subject to race conditions in multi-threaded environments.

05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

Solution:Encapsulated as functions and classes

Usage

Experiments

Result Analysis

```
handle_listener() : int
handle_sig() : int
handle_in() : void
configure(int, char **) const : int
init(int, char **) : void
process(const int &) : void
```

1.Extract the code blocks as classes and functions

2. Pass the value by reference

3. Adhere to the const semantics if possible

05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

New Issue:global and static variables of a class

Usage

Experiments

Result Analysis

1. Global variables must be const
2. Function pointers require static function
3. static function requires static members in a class
4. New issue introduced: static members inaccessible when compiling

```
/usr/bin/ld: main.o: in function `epoll_util::run()':
main.cc:(.text._ZN10epoll_util3runEv[_ZN10epoll_util3runEv]+0x29): undefined ref
erence to `fd_util::epollfd'
/usr/bin/ld: main.o: in function `epoll_util::clear()':
main.cc:(.text._ZN10epoll_util5clearEv[_ZN10epoll_util5clearEv]+0x21): undefined
 reference to `fd_util::pipefd'
/usr/bin/ld: main.cc:(.text._ZN10epoll_util5clearEv[_ZN10epoll_util5clearEv]+0x2
e): undefined reference to `fd_util::pipefd'
/usr/bin/ld: warning: creating DT_TEXTREL in a PIE
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:59: webserver] Error 1
make[1]: Leaving directory '/home/victor/Web-Server'
make: *** [Makefile:29: all] Error 2
```

05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

Usage

Experiments

Result Analysis

Solution:define static members after declaration globally

```
class fd_util{
public:
    static int pipefd[2];
    static heap_util_timer timer_lst;
    static int epollfd;
    static int cnt;
    static void addfd( int epfd, int fd ) {...}
    static void sig_handler( int sig ) {...}
    static void addsig(int sig, void( handler )(int)=sig_handler){...}
    static void timer_handler() {...}
    static void cb_func( const client_data* user_data ) {...}
};
int fd_util::pipefd[2]={ [0]: 0, [1]: 0};
heap_util_timer fd_util::timer_lst;
int fd_util::epollfd=0;
int fd_util::cnt=0;
```

05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

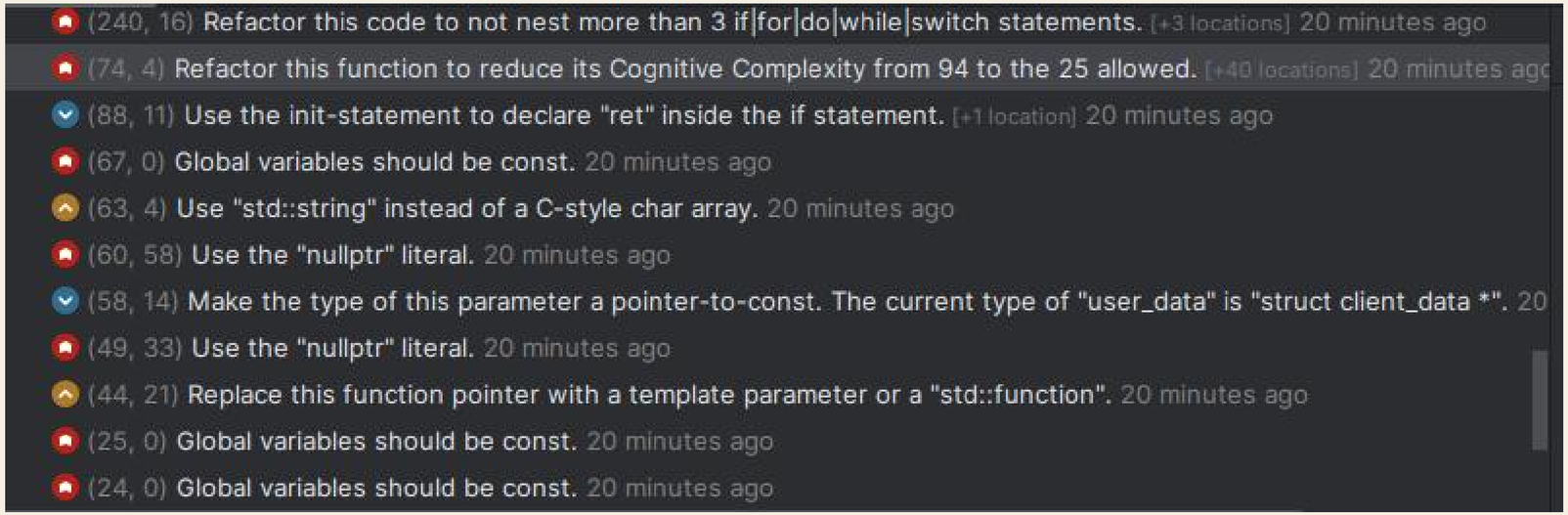
Usage

Experiments

Result Analysis

Miscellaneous:Other types of issue

- Preferring enumclass not enum
- constexpr not MACRO for const variables
- Specific types of exception not general
- Forbidden or explicitly write the copy and assign constructor
- Pass const reference if possible
- Make the function const if it doesn't modify any parameter



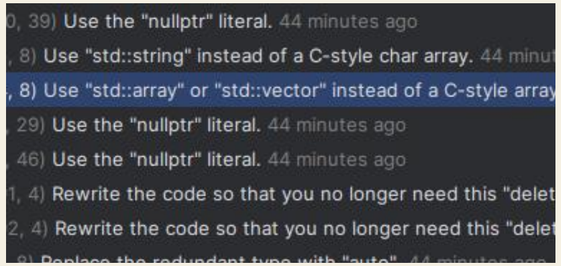
05Usage&Experiment&Analysis

(3)JetBrains CLion/C++

Usage

Experiments

Result Analysis



Not all of the review suggestions are rational.

- STL Containers and string may be unsafe in multi-threading context
- Substitute macro with enum class may compromise other macros related to whom.

```
int main( int argc, char* argv[] ) {
    epoll_util util(argc,argv);
    if(!util.get_init_state()) {
        util.clear();
        return 0;
    }
    //Configuration
    while( !util.get_stopserver())
    {
        util.run();
    }
    util.clear();
    return 0;
}
```

More C++ paradigm than C

After struggling through the static analysis of code, my code becomes more maintainable and understandable.I also find that C++ is considerably different from C,which means that I need to throw so many bad habits taken from C and write a more organized and awecome C++ code.

06

Extension--SonarQube

06Extension--SonarQube

SonarQube is an open-source platform for continuous inspection of code quality and static code analysis, supporting multiple programming languages. It can be integrated into CI/CD pipelines to provide comprehensive assessments of code quality and security.

SonarLint is primarily used for immediate feedback during development, while SonarQube is used for in-depth analysis and long-term quality tracking of codebases.

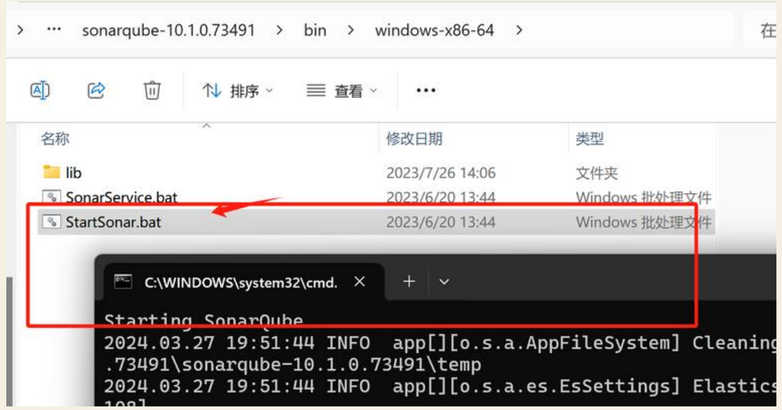
06Extension--SonarQube

Usage

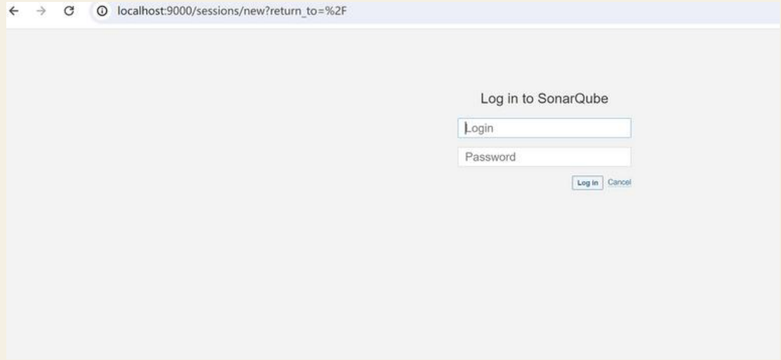
Experiments

Result Analysis

1.Download the sonarqube software , click and run StartSonar.bat.



2.And the Web interface is on the localhost:9000, input the Login_number and password.



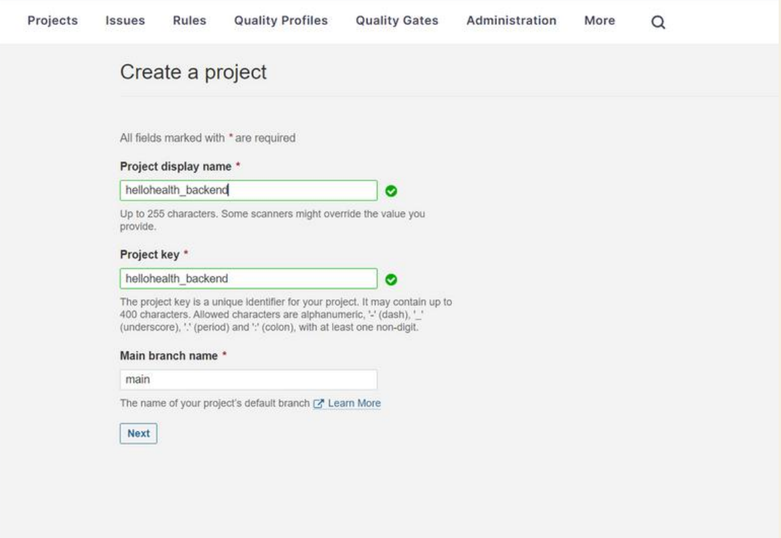
06Extension--SonarQube

Usage

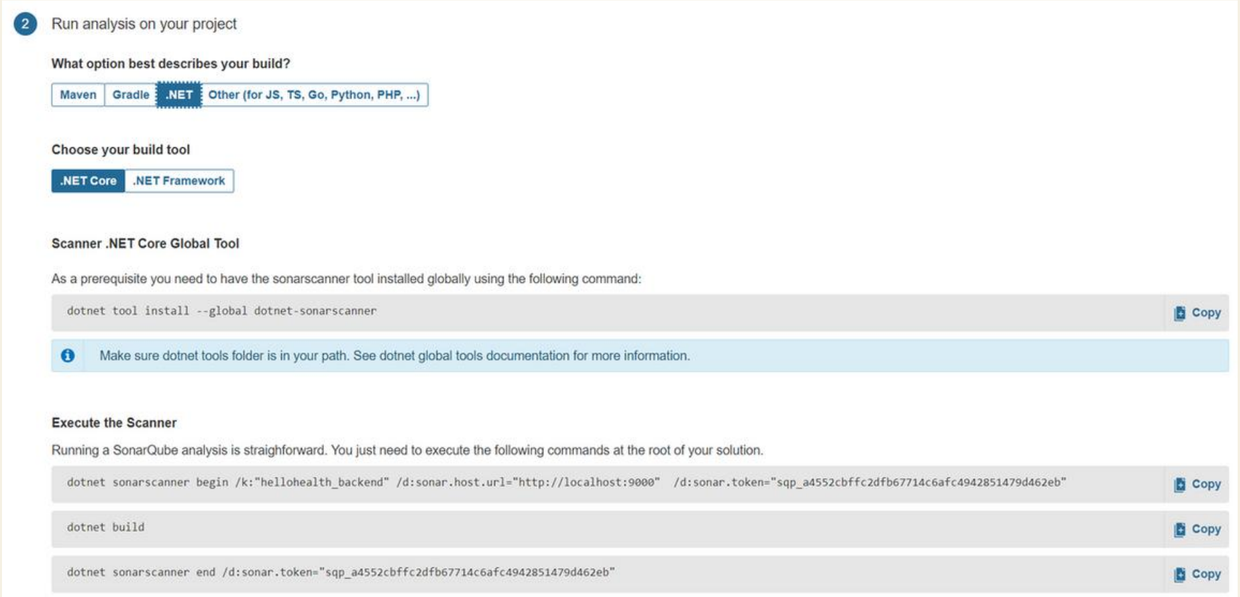
Experiments

Result Analysis

3.Create a project, generate the project token .



4.Run the command according to the instructions.



06Extension--SonarQube

Usage

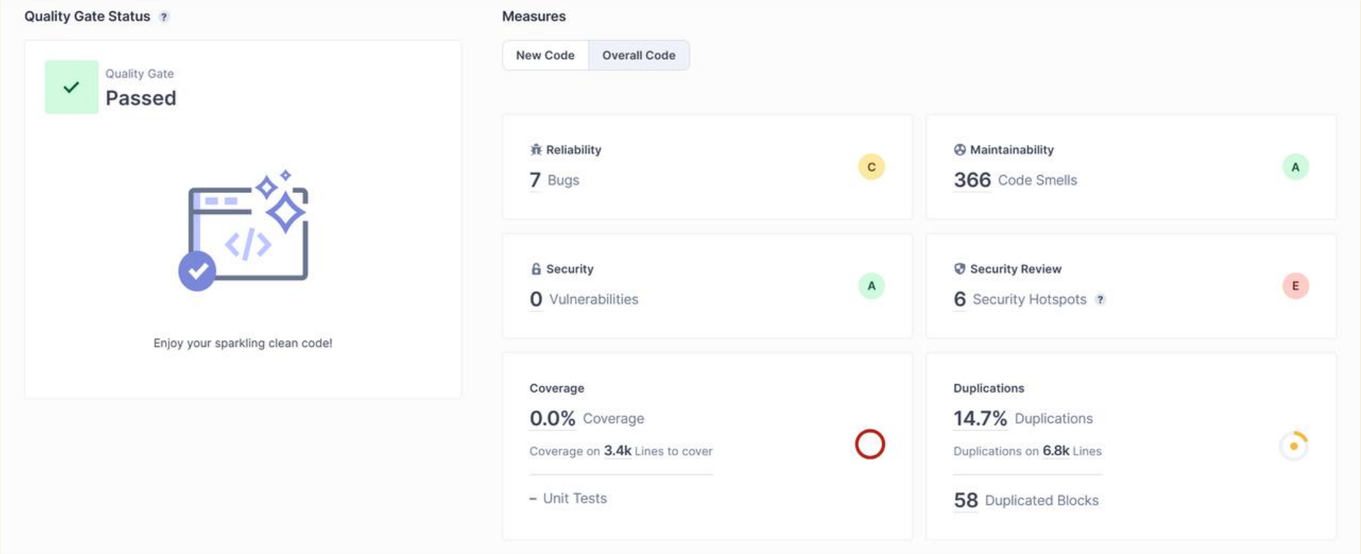
Experiments

Result Analysis

This project involves developing the backend of a database-driven application using C# ,ASP.NET within the Visual Studio IDE.

We use the same project mentioned before.

The result is on the http://localhost:9000/dashboard?id=hellohealth_backend.



06Extension--SonarQube

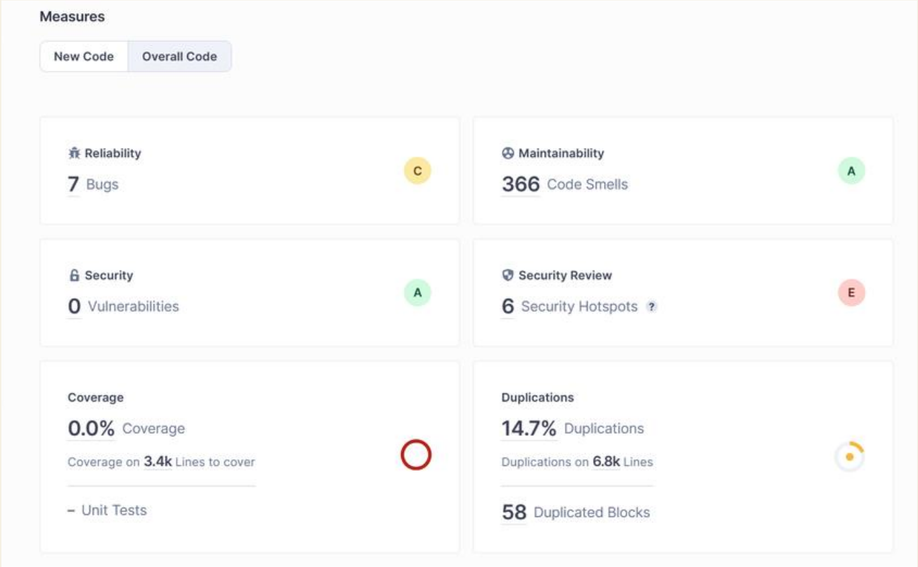
Usage

Experiments

Result Analysis

It offers 6 main metrics:

- Maintainability: How easy it is to maintain and update the code. Code smells.
- Reliability: It identifies critical issues that could cause system crashes or unexpected behavior, bugs.
- Security: Vulnerabilities that could be exploited by attackers to gain unauthorized access, disrupt services, or steal sensitive information.
- Security Review: It is an in-depth analysis aimed at reinforcing application security.Security hotspots.



- Coverage: Quantifies the extent to which the source code is executed by automated tests.
- Duplications: This metric identifies portions of the code that are duplicated elsewhere within the codebase.

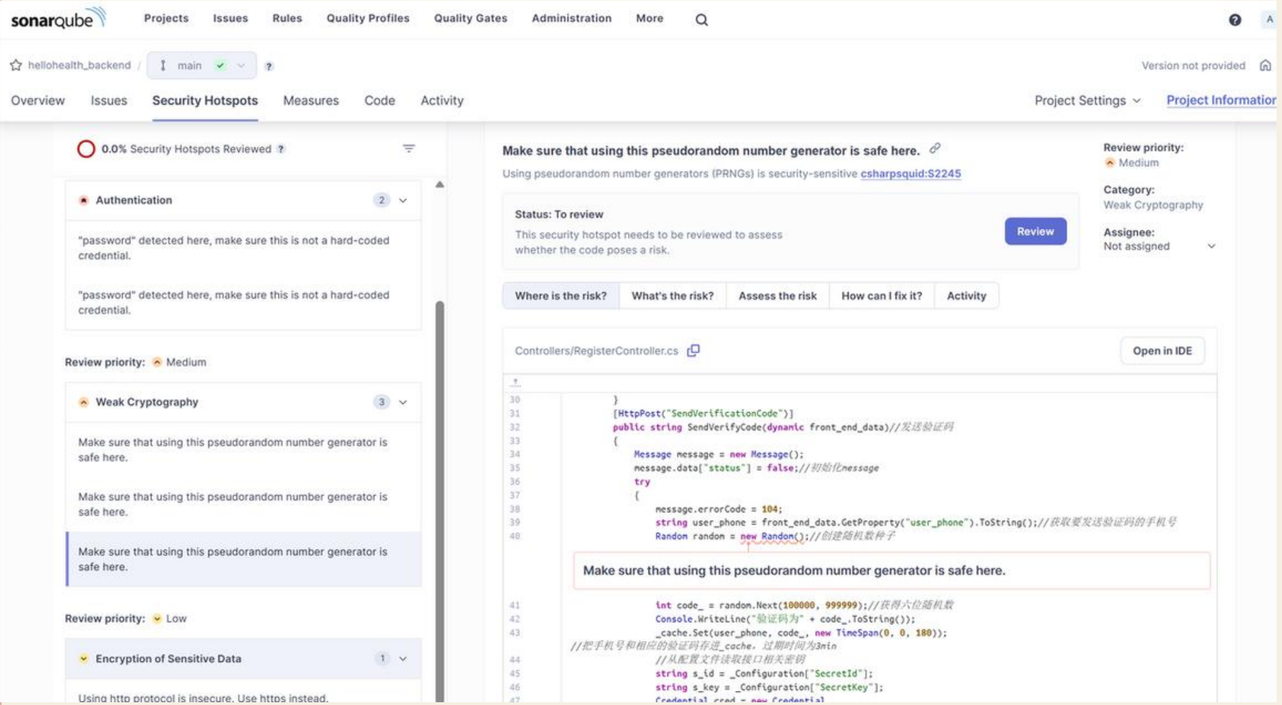
06Extension--SonarQube

SonarQube has valueble suggestions abont security hotspots.

Usage

Experiments

Result Analysis



- Different Level of review priority.

- It points out the problem with Password in the setting file. It is certainly of high severity.

- It points out the problem with random number.We should make sure it is safe here.

06Extension--SonarQube

Usage

Experiments

Result Analysis

Differences between SonarQube & SonarLint.

	SonarLint	SonarQube
Analysis Scope	Limited	Comprehensive
Feedback	Real-time	Complex integration
Use	Easy to install as an IDE extension	Diffucult to set up
Cost	Free	Cost for Enterprise Features

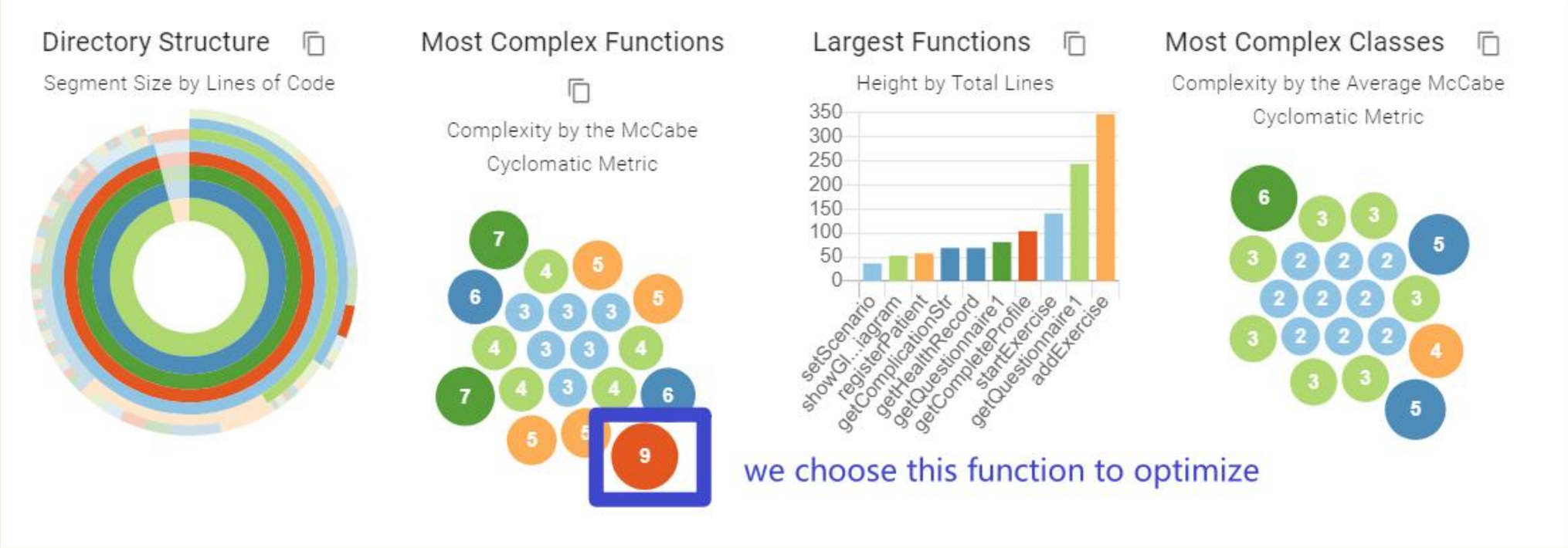
07

Control Flow Graph

07Control Flow Graph

Original code

This is an overview provided by ‘Understand’. It can assess code complexity based on cyclomatic complexity.



07Control Flow Graph

The graph comprises 22 vertices and 29 edges, indicating the presence of 9 cycles.

Original code

Therefore, we carefully examined the code and find many issues.

Variables are declared using primitive types and are not initialized.

```
Double HYPER_THRESHOLD,EU_THRESHOLD,AFTERLUNCH_HYPER_THRESHOLD,AFTERDINNER_HYPER_THRESHOLD;
```

The variable 'AfterDinner' is defined with a logical flaw; its value will always be 'false'.

Furthermore, due to 'AfterDinner' always being false, there are unreachable segments of code.

```
Boolean AfterDinner=(date.getHour()>18&&date.getHour()<19);//根据时间判断是否在餐后
Boolean AfterLunch=(date.getHour()>12&&date.getHour()<14);
```

07Control Flow Graph

We identified redundant conditions within the following 'if' statement and the absence of consideration for the condition 'data == HYPER_THRESHOLD.'

Original code

Final optimization: refactor code from complex 'if-else' chains into modularized functions, enhancing readability and maintainability.

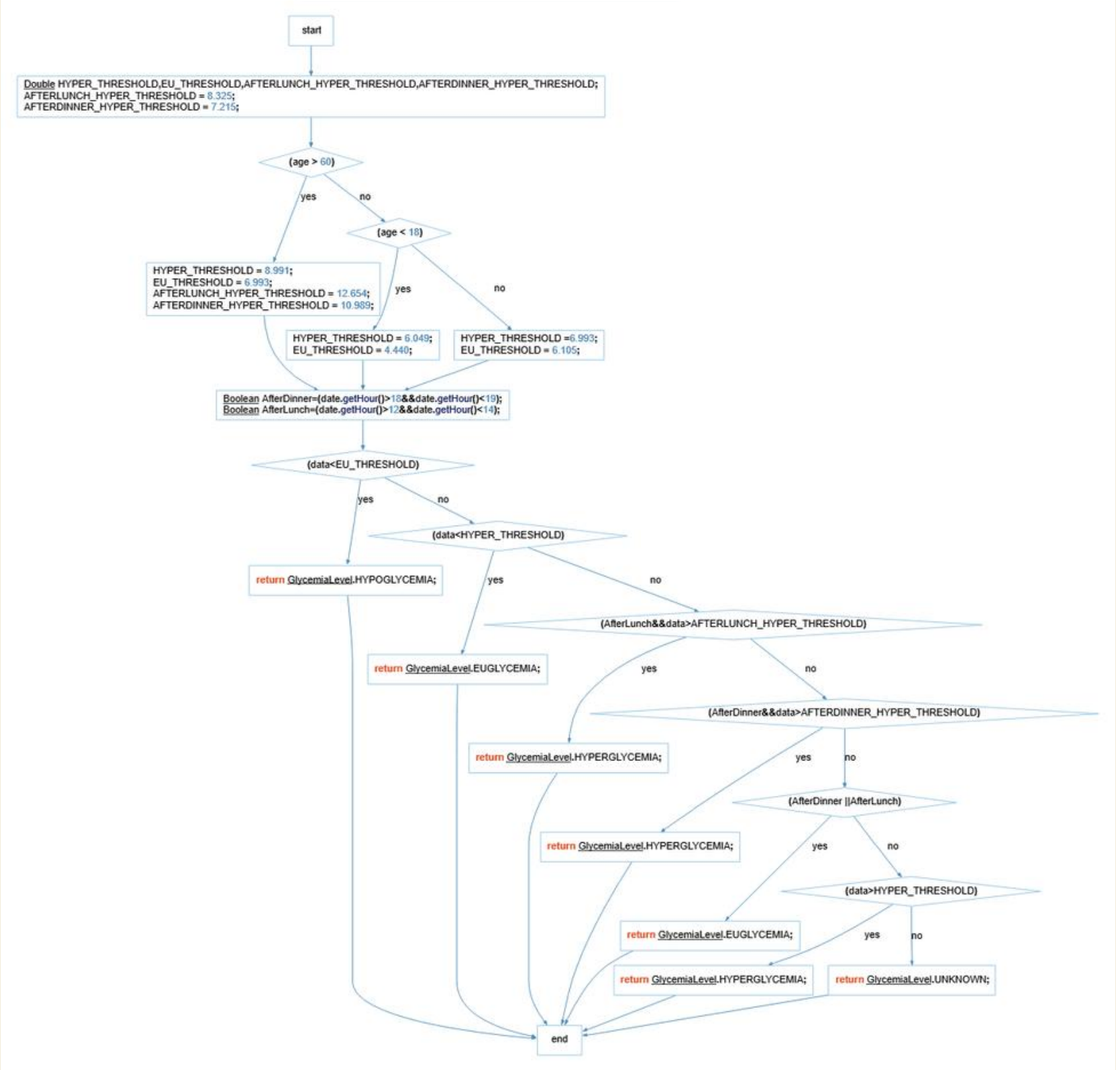
```
if(data<EU_THRESHOLD)//RGBA for Red
    return GlycemiaLevel.HYPOGLYCEMIA://低血糖
else if(data<HYPER_THRESHOLD){
    return GlycemiaLevel.EUGLYCEMIA;//正常
}else if(AfterLunch&&data>AFTERLUNCH_HYPER_THRESHOLD){//如果在餐后，可能高血糖或正常血糖
    return GlycemiaLevel.HYPERGLYCEMIA;
}else if(AfterDinner&&data>AFTERDINNER_HYPER_THRESHOLD)
    return GlycemiaLevel.HYPERGLYCEMIA;
else if(AfterDinner ||AfterLunch){
    return GlycemiaLevel.EUGLYCEMIA;
}else{
    if(data>HYPER_THRESHOLD)
        return GlycemiaLevel.HYPERGLYCEMIA;
}
return GlycemiaLevel.UNKNOWN;
```

unreachable

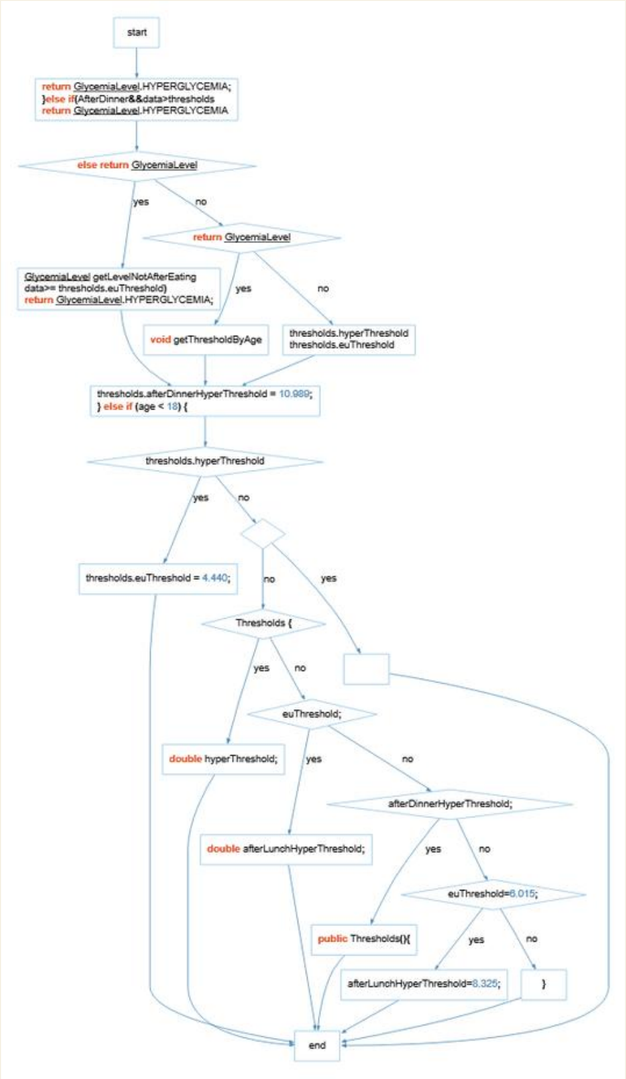
duplicate judge

07Control Flow Graph

Original code



Optimized code



Comparing:

22 vertices

29 edges

9 cycles

07Control Flow Graph

Original code

Although we made modifications to address the warnings and enhanced code readability and decoupling through encapsulation, the fundamental logic of the decision-making process remained unaltered, thus resulting in an unchanged cyclomatic complexity😩.

Despite this outcome, we deem it acceptable.

Optimized code

```
public GlycemiaLevel GetGlycemiaLevel(Double age,LocalDateTime date,Double data){
    Thresholds thresholds=new Thresholds();
    getThresholdByAge(thresholds,age);
    boolean AfterDinner=(date.getHour()>=18&&date.getHour()<=19);//根据时间判断是否在餐后
    boolean AfterLunch=(date.getHour()>=12&&date.getHour()<14);
    if(data<thresholds.euThreshold)//RGBA for Red
        return GlycemiaLevel.HYPOGLYCEMIA;//低血糖
    else if(AfterDinner ||AfterLunch){//餐后
        return getLevelAfterEating(AfterDinner,AfterLunch,data,thresholds);
    }else{//不在餐后
        return getLevelNotAfterEating(data,thresholds);
    }
}
```

newly added

Thanks For Watching !

Advised by Prof.Qin Liu

Group 16:
2151409 Yuntao Hu
2152085 Yifei Sun
2054099 Jieying Ye
2152193 Yixin Li