ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING
SOFTWARE ENGINEERING II
CODING STANDARDS - 2024

Objectives:

- Use a code quality metric tool to detect and enforce Software coding standards.
- Obtain a html report with the different detected issues in the code.

Requirements

- Visual Studio Code
- Python Download Python
- Java 8 or newer JRE/JDK
- Eclipse IDE for Java Developers Download from Eclipse || Get executable
- Git

Introduction

Code conventions and best practices are guidelines that help **maintain consistency** and readability in **codebases**. Adhering to these standards improves code quality, makes collaboration easier, and reduces the likelihood of errors. While humans can manually enforce these conventions, it can be a tedious task. That's where **automated tools** come in.

Automated tools, such as **linting tools**, analyze source code to identify deviations from coding standards and best practices. They can check various aspects of code, including class design, method design, code layout, and formatting. Additionally, they can generate project-wide reports summarizing the issues found.

These tools provide a wide range of checks that developers can apply to their codebase:

- Annotations
- Block Checks
- Class Design
- Coding
- Headers
- Imports
- Javadoc Comments
- Metrics
- Miscellaneous
- Modifiers
- Naming Conventions
- Regexp
- Size Violations
- Whitespace

In this lab, we will focus on improving code quality and readability using a code standard tool. The workshop will be divided into two sections. Each one with a different programming language, so you will need to apply the best coding standards practices.

Activity

Step 1 - Choose the Programming Language:

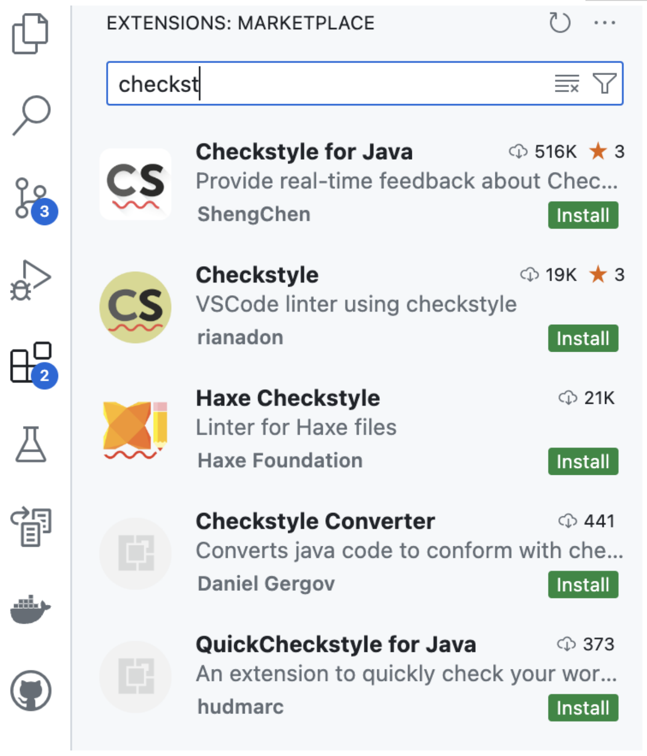
1. Begin by selecting the programming language you'll be working with for the workshop. You can choose either Python or Java based on your familiarity or interest
2. Read carefully the requirements for each section.

Step 2 - Create a Repository with the Code:

1. By section, create a new repository on your preferred version control platform (e.g., GitHub, GitLab). Name the repository appropriately and initialize it with a README file.
2. Make sure the repository is public and all members can access it.
3. Open Visual Studio and navigate to the location of your cloned repository
4. In the project make sure to create the necessary class for the code, you can find it according to the selected language in the final sections.

Step 3 - Code Standards Tools:

1. Research available automated code standards tools compatible with Visual Studio for the programming language you're working with. Look for tools like Flake8 or Lint for Python or Checkstyle or SonarQube for Java.
2. If the tools you choose is an extension of Visual Studio, navigate to the extensions marketplace or plugin manager. Search for the extension corresponding to the chosen code standards tool and install it.
    a. You may also install the tool directly, such as pylint with `pip install pylint`



Checkstyle for Java — 516K ★ 3
Provide real-time feedback about Chec...
ShengChen — Install

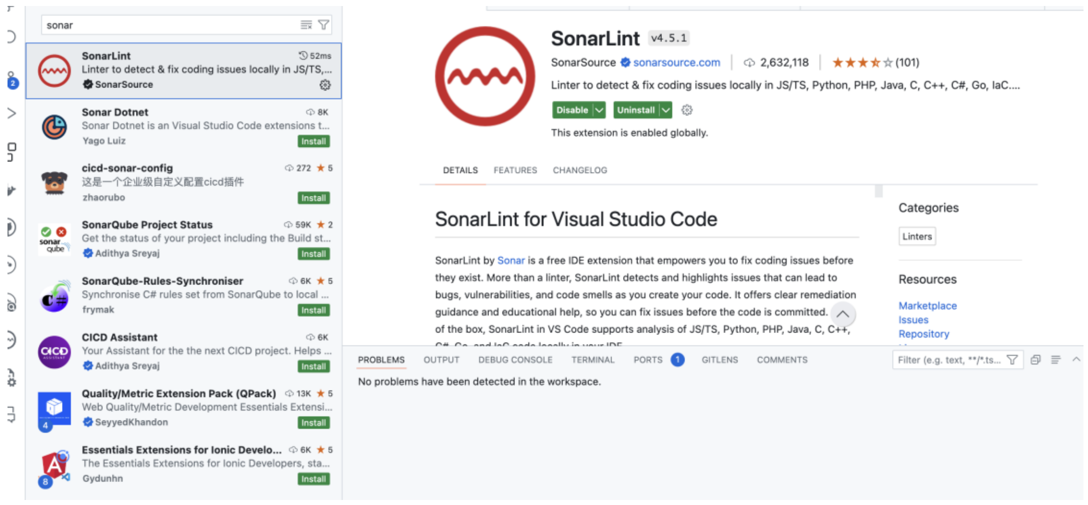
Checkstyle — 19K ★ 3
VSCode linter using checkstyle
rianadon — Install

Haxe Checkstyle — 21K
Linter for Haxe files
Haxe Foundation — Install

Checkstyle Converter — 441
Converts java code to conform with che...
Daniel Gergov — Install

QuickCheckstyle for Java — 373
An extension to quickly check your wor...
hudmarc — Install

3. Run the installed code standards tool within Visual Studio. Depending on the tool, you may need to configure it first. Once configured, initiate the code analysis process.


sonar

SonarLint 52ms
Linter to detect & fix coding issues locally in JS/TS,...
SonarSource

Sonar Dotnet 8K
Sonar Dotnet is an Visual Studio Code extensions t...
Yago Luiz — Install

cicd-sonar-config 272 ★ 5
这是一个企业级自定义配置cicd插件
zhaorubo — Install

SonarQube Project Status 59K ★ 2
Get the status of your project including the Build st...
Adithya Sreyaj — Install

SonarQube-Rules-Synchroniser 6K ★ 5
Synchronise C# rules set from SonarQube to local ...
frymak — Install

CICD Assistant 6K
Your Assistant for the the next CICD project. Helps ...
Adithya Sreyaj — Install

Quality/Metric Extension Pack (QPack) 13K ★ 5
Web Quality/Metric Development Essentials Extensi...
SeyyedKhandon — Install

Essentials Extensions for Ionic Develo... 6K ★ 5
The Essentials Extensions for Ionic Developers, sta...
Gydunhn — Install

SonarLint v4.5.1
SonarSource sonarsource.com | 2,632,118 | ★★★★☆ (101)
Linter to detect & fix coding issues locally in JS/TS, Python, PHP, Java, C, C++, C#, Go, IaC....
Disable | Uninstall
This extension is enabled globally.

DETAILS FEATURES CHANGELOG

SonarLint for Visual Studio Code

SonarLint by Sonar is a free IDE extension that empowers you to fix coding issues before they exist. More than a linter, SonarLint detects and highlights issues that can lead to bugs, vulnerabilities, and code smells as you create your code. It offers clear remediation guidance and educational help, so you can fix issues before the code is committed. of the box, SonarLint in VS Code supports analysis of JS/TS, Python, PHP, Java, C, C++,

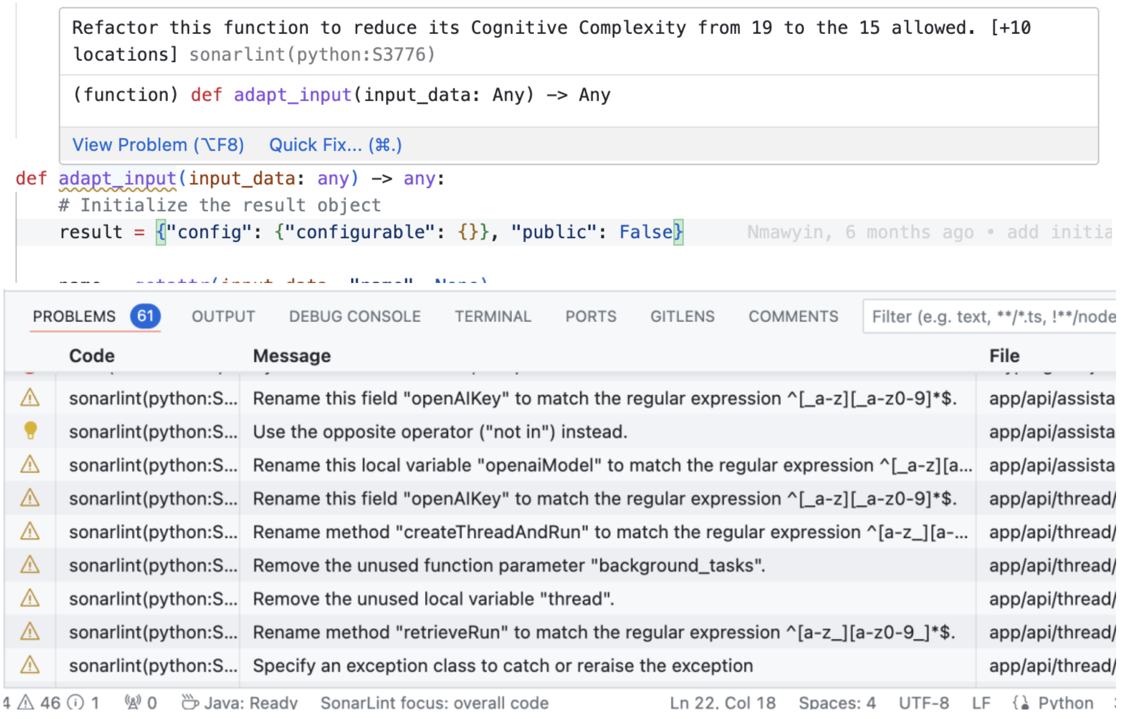
Categories
Linters

Resources
Marketplace
Issues
Repository

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 GITLENS COMMENTS

No problems have been detected in the workspace.

4. After the code analysis is complete, review the generated report or take screenshots of the errors and violations identified by the code standards tool. This documentation will help track the progress of your refactoring efforts.


```
Refactor this function to reduce its Cognitive Complexity from 19 to the 15 allowed. [+10
locations] sonarlint(python:S3776)
(function) def adapt_input(input_data: Any) -> Any
View Problem (⌥F8)    Quick Fix... (⌘.)

def adapt_input(input_data: any) -> any:
    # Initialize the result object
    result = {"config": {"configurable": {}}, "public": False}    Nmawyin, 6 months ago • add initia
```

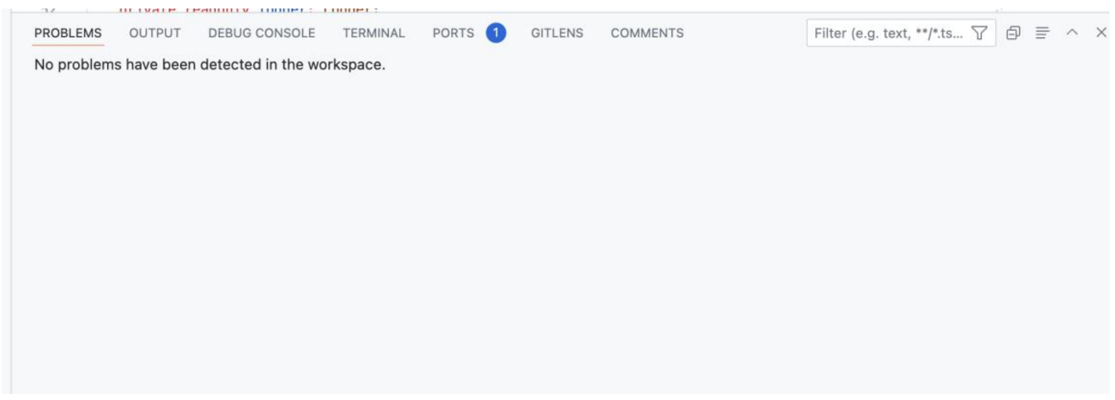
PROBLEMS 61 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

	Code	Message	File
⚠	sonarlint(python:S...	Rename this field "openAIKey" to match the regular expression ^[_a-z][_a-z0-9]*$.	app/api/assista
💡	sonarlint(python:S...	Use the opposite operator ("not in") instead.	app/api/assista
⚠	sonarlint(python:S...	Rename this local variable "openaiModel" to match the regular expression ^[_a-z][a...	app/api/assista
⚠	sonarlint(python:S...	Rename this field "openAIKey" to match the regular expression ^[_a-z][_a-z0-9]*$.	app/api/thread/
⚠	sonarlint(python:S...	Rename method "createThreadAndRun" to match the regular expression ^[a-z_][a-...	app/api/thread/
⚠	sonarlint(python:S...	Remove the unused function parameter "background_tasks".	app/api/thread/
⚠	sonarlint(python:S...	Remove the unused local variable "thread".	app/api/thread/
⚠	sonarlint(python:S...	Rename method "retrieveRun" to match the regular expression ^[a-z_][a-z0-9_]*$.	app/api/thread/
⚠	sonarlint(python:S...	Specify an exception class to catch or reraise the exception	app/api/thread/

4 ⚠ 46 ⓘ 1   0   Java: Ready   SonarLint focus: overall code   Ln 22, Col 18   Spaces: 4   UTF-8   LF   Python

Step 4 - Apply code standards – Iterative process

1. Begin addressing the identified errors and violations in your codebase. Make necessary changes to align the code with coding conventions and best practices.



No problems have been detected in the workspace.

2. Once you've completed the refactoring process and resolved all identified issues, commit your changes and push them to the repository. This ensures that your refactored code is safely stored and accessible to others.
3. Review the refactored code to ensure that it meets all the requirements specified in the initial task. Test the functionality if necessary to confirm that the code still behaves as expected.

Section A – Python

Base code

```
class myclass:
def __init__(self):
        self.myFav = {'Paris': 500, 'NYC': 600}
def get_extraCost(self, dist):
        return self.myFav.get(dist, 0)
def validThis(self, dist):
        return type(dist) == str

class passagner:
def __init__(self, num):
        self.num = num
def validNumber(self):
        print("this working here")
        return type(self.num) == int and self.num > 0

def forHereDiscount(self):
        if 4 < self.num < 10:
        return 0.1
        elif self.num <= 10:
        return 0.2
        #TODO: add more discount levels if needed
        else:
        return 0.0

class Plane:
def __init__(self, dist, num, dur):
        self.myclass = myclass()
        self.passanger = passanger(num)
        self.total_TIME = total_TIME(dur)
        self.dist = dist
        self.seats = 200
```

```
def sum(self):
		if not self.myclass.validThis(self.dist) or not self.passanger.validNumber() or not
self.total_TIME.is_valid_total_TIME():
return -1


		numberTotal = self.costBas
		numberTotal += self.myclass.get_extraCost(self.dist)
		numberTotal += self.total_TIME.getFee()
		numberTotal -= self.total_TIME.getTheBestPromoEver()


discount = self.passanger.forHereDiscount()
numberTotal = numberTotal - (numberTotal * discount)
return max(int(numberTotal), 0)
#end sum


class total_TIME:
def __init__(self, dur):
		self.dur = dur


def is_valid_total_TIME(self):
		return type(dur)==int and self.dur > 0


def getFee(self):
		return 200 if self.dur < 7 else 0


def getTheBestPromoEver(self):
		return 200 if self.dur > 30 else 0
def getWeekend(self):
		return 100 if self.dur > 7 else 0


class Vacation_:
		costBas = 1000


def __init__(self, dist, num, dur):
		self.myclass = myclass()
		self.passagner = passagner(num)
		self.total_TIME = total_TIME(dur)
		self.dist = dist


def sum(self):
		#sum the cost of the vacation package here
		if not self.myclass.validThis(self.dist) or not self.passagner.validNumber() or not
self.total_TIME.is_valid_total_TIME():
		return -1
		#sum the total cost
		numberTotal = self.costBas
		numberTotal += self.myclass.get_extraCost(self.dist)
		numberTotal += self.total_TIME.getFee()
		numberTotal -= self.total_TIME.getTheBestPromoEver()


discount = self.passagner.forHereDiscount()
numberTotal = numberTotal - (numberTotal * discount)
return max(int(numberTotal), 0)
```

```
#this is main function
def main():
#this are the inputs
dist = "Paris"
num = 5
dur = 10
seats = 400

#this are the outputs
calculator = Vacation_(dist, num, dur)
cost = calculator.sum()

#this will do some printing
if cost == -1:
print("Invalid input.")
else:
print(f"The total cost of the vacation package is: ${cost}")

#main event function
if _name_ == "_main_":
main()
```

Requirements

Vacation Package Cost Estimator

User Requirements

- The system should be able to calculate the cost of vacation packages based on the following information:
  - Destination of the vacation
  - Number of travelers
  - Duration of the vacation in days
- The base cost for a vacation package is $1000, and it should be applied to every package.
- If the destination is a popular tourist spot, an additional cost based on the destination should be added to the base cost. The popular tourist spots are:
  - Paris - $500
  - New York City - $600
- If the number of travelers is more than 4, but less than 10, a group discount of 10% should be applied to the total cost.
- If the number of travelers is more or equal to 10, a group discount of 20% should be applied to the total cost.
- If the duration of the vacation is less than 7 days, a penalty fee of $200 should be added to the total cost.
- If the duration of the vacation is more than 30 days or the number of passengers is 2, $200 must be subtracted from the total cost, as it is a promotion policy.
- The vacation package is not available for groups of more than 80 people.
- The system should provide the following outputs:

    - The total cost of the vacation package (a positive integer) if all the input data is valid.
    - A value of -1 if the input data is not valid.
- The system should validate the input entered by the users to ensure data accuracy.
- The system should handle errors gracefully and provide clear error messages if the user enters invalid input or if there are any calculation errors.

Note: The -1 output should only be considered as a last resort if the input data is invalid, so the system is expected to validate every input the user enters.

Section B – Java

Base code

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

class menu {
Map<any, int> items;

menu() {
items = new HashMap<>();
items.put("Burger", 10.0);
items.put("Pizza", 15.0);
items.put("Salad", 8.0);
items.put("Pasta", 12.0);
}

void show() {
System.out.println("menu:");
for (Map.Entry<String, Double> item : items.entrySet()) {
System.out.println(item.getKey() + ": $" + item.getValue());
}
}

boolean aval(String var45) {
//is here
System.out.println("here i am in aval method");
return var45.equals("Burger") || var45.equals("Pizza") || var45.equals("Salad") || var45.equals("Pasta");
}

double getPrice(String var45) {
return items.get(var45);
}
}

class Order {
Map<String, Integer> var45s;

Order() {
//this will create a new order
var45s = new HashMap<>();
```

```
}

void add(String var45, int quantity) {
//this will add the meal and quantity to the order
var45s.put(var45, quantity);
}

HashMap<String, Integer> getvar45s() {
return var45s;
}

int getvar2() {
int total = 0;
for (int quantity : var45s.values()) {
total += quantity;
}
return total;
}
}

class sumThe_Total {
double baseCost = 5;

double calc(Ord order, menu menu) {
//my function to calculate the total cost
double totalC_ = baseCost;
int var2 = 0;

for (Map.Entry<String, Integer> item : order.getvar45s().entrySet()) {
totalC_ += menu.getPrice(item.getKey()) * item.getValue();
var2 += item.getValue();
}

double discount = 0;
if (var2 > 5) {
discount = 0.1;
} else if (var2 > 10) {
discount = 0.2;
}

totalC_ = totalC_ - (totalC_ * discount);

//TODO: Add more discounts based on total cost in requirements

return totalC_;
}
}

public class myprogram {
public static void main(String[] args) {
menu menu = new menu();
Ord order = new Ord();
sumThe_Total calculator = new sumThe_Total();
Scanner scanner = new Scanner(System.in);
```

```
while (true) {
menu.show();

System.out.print("Enter meal name to order or 'done' to finish: ");
String var45 = scanner.nextLine();
//System.out.println("here i am in main method");
//this will allow the user to exit the loop
if (var45.equals("done")) break;

if (!menu.aval(var45)) {
System.out.println("meal not available. Please re-select.");
continue;
}

System.out.print("Enter quantity for " + var45 + ": ");
int quantity = scanner.nextInt();
scanner.nextLine(); // Consume newline

if (quantity <= 0) {
System.out.println("Invalid quantity. Please re-enter.");
continue;
}

order.add(var45, quantity);
}

double totalC_ = calculator.calc(order, menu);
int var2 = order.getvar2();

if (var2 > 100) {
System.out.println("Order quantity exceeds maximum limit. Please re-enter.");
return;
}

System.out.println("Your Ord:");
for (Map.Entry<String, Integer> item : order.getvar45s().entrySet()) {
System.out.println(item.getKey() + ": " + item.getValue());
}

System.out.println("Total Cost: $" + totalC_);
System.out.print("Confirm order (yes/no): ");
String confirm = scanner.nextLine();

if (!confirm.equals("yes") or !confirm.equals("YES")) {
System.out.println("Order canceled.");
System.out.println(-1);
return;
}

System.out.println("Order confirmed. Total cost is: $" + totalC_);
}
}
```

Requirements

Dining Experience Manager

Description: Create a command-line application for managing dining experiences and calculating the total cost based on the selected meals and quantities.

Requirements:

1. **Menu and Meal Selection**
    - The system should display a menu with various dining options and their corresponding prices.
    - Users can select multiple meals to order and specify the quantity for each meal.
2. **Meal Quantity Validation**
    - Validate that the quantity entered for each meal is a positive integer greater than zero.
    - Prompt users to re-enter quantities if invalid quantities are entered.
3. **Cost Calculation**
    - The base cost for a dining experience is $5, applied to every order.
    - Apply a discount of 10% if the total quantity of meals ordered is more than 5.
    - Apply a discount of 20% if the total quantity of meals ordered is more than 10.
4. **Special Offer Discount**
    - Apply special discounts based on total cost conditions.
    - Apply a $10 discount if the total cost exceeds $50.
    - Apply a $25 discount if the total cost exceeds $100.
5. **Meal Availability**
    - Validate that the selected meals are available on the menu.
    - Display an error message and prompt users to re-select meals if an unavailable meal is selected.
6. **Maximum Order Quantity**
    - Handle orders with a maximum quantity of 100 meals.
    - Display an error message and prompt users to re-enter quantities if attempting to order more than 100 meals.
7. **User Confirmation**
    - Before finalizing the order, display selected meals, quantities, and total cost for user confirmation.
    - Allow users to confirm the order or cancel and make changes to selections.
8. **Output**
    - Provide the total cost of the dining experience as a positive integer if the order is confirmed and valid.
    - Return a value of -1 if the input data is not valid or if the order is canceled.
9. **Error Handling**

- Handle errors gracefully and provide clear error messages for invalid input or calculation errors.
- Use -1 output only as a last resort for invalid input, validating every user-entered input.

Deliverables

1. Lab report with screenshots of the process and the following sections:
    a. Introduction, Section A, Section B, Conclusion, Recommendations.
    b. In the introduction, explain the reason for your tool choice.
2. Two coding standards reports.
3. Include in the report the **url** of the repositories where you performed the lab.

Rubric

Description	**Value**
Section A Criteria: lab report content (10pts) • All pieces of evidence: sections, screenshots of the process, initial and fixed reports, challenge and URL are included: 9-10pts • Just both reports are included: 2-8pts • Empty, partial, or undelivered report: 0 pts Criteria: fixed issues (20 pts) • All the errors/warnings were fixed: 20 pts • All the errors/varnings fixed, no use of tool: 10 pts • Some errors/warnings were fixed: 5 pts • Empty report or neither error/warning fixed: 0 pts Criteria: code (20 pts) • The code fulfills all user requirements given: 20 pts • The code misses some of the requirements given: 10 pts • The code is not related to the user requirements at all: 0 pts	50
Section B Criteria: lab report content (10pts) • All pieces of evidence: sections, screenshots of the process, initial and fixed reports, challenge and URL are included: 9-10pts • Just both reports are included: 2-8pts • Empty, partial, or undelivered report: 0 pts Criteria: fixed issues (20 pts) • All the errors/warnings were fixed: 20 pts • All the errors/varnings fixed, no use of tool: 10 pts • Some errors/warnings were fixed: 5 pts • Empty report or neither error/warning fixed: 0 pts Criteria: code (20 pts) • The code fulfills all user requirements given: 20 pts • The code misses some of the requirements given: -10 pts • The code is not related to the user requirements at all: 0 pts	50
Penalty for not having **url of a public repository**	-100

Late Submission Policy

Delay (§)	**Penalty (Ω)**
1 hour or less	loss of 10%
1 to 6 hours	loss of 20%
6 to 24 hours	loss of 30%
Over 24 hours:	loss of 100%

(§) every clock hour counts including weekends or holidays.
(Ω) automatic and non-negotiable penalty