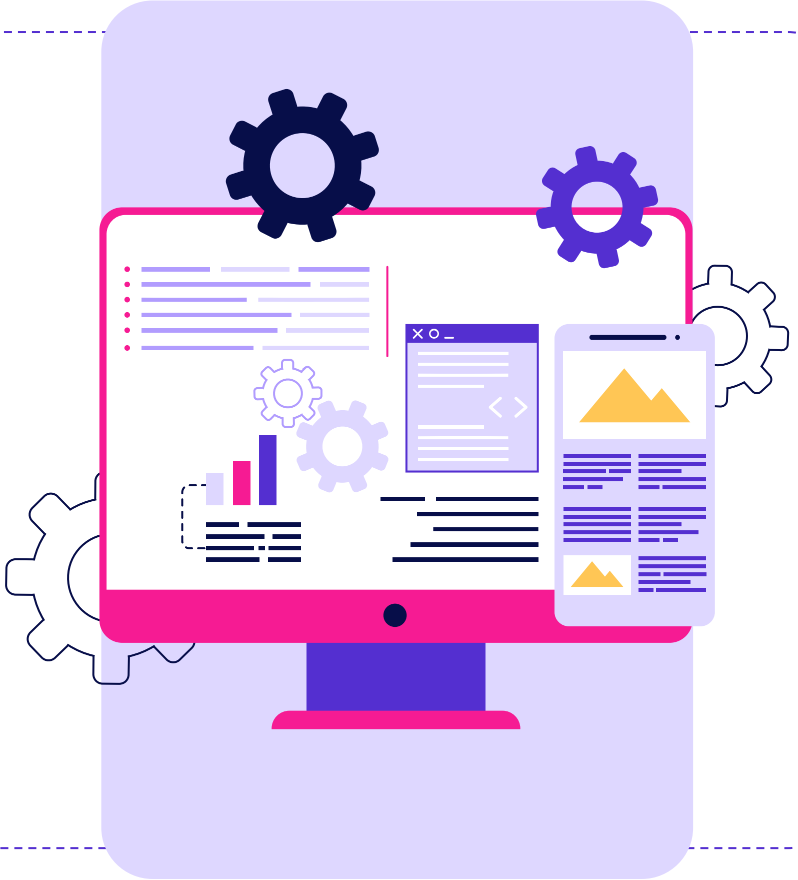
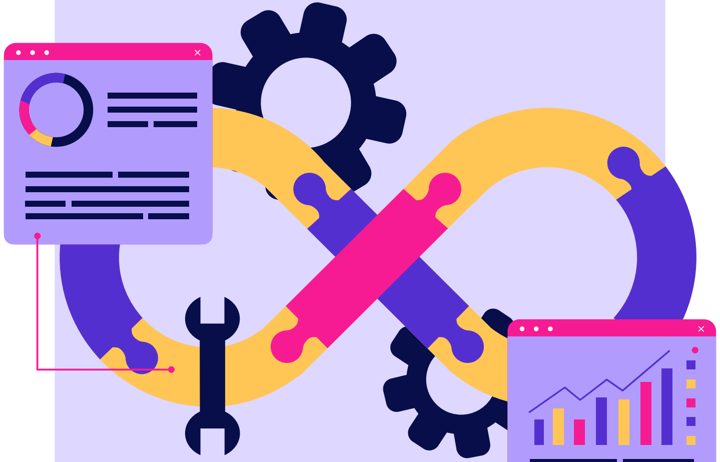
Project 1

Team 23





Team

Members

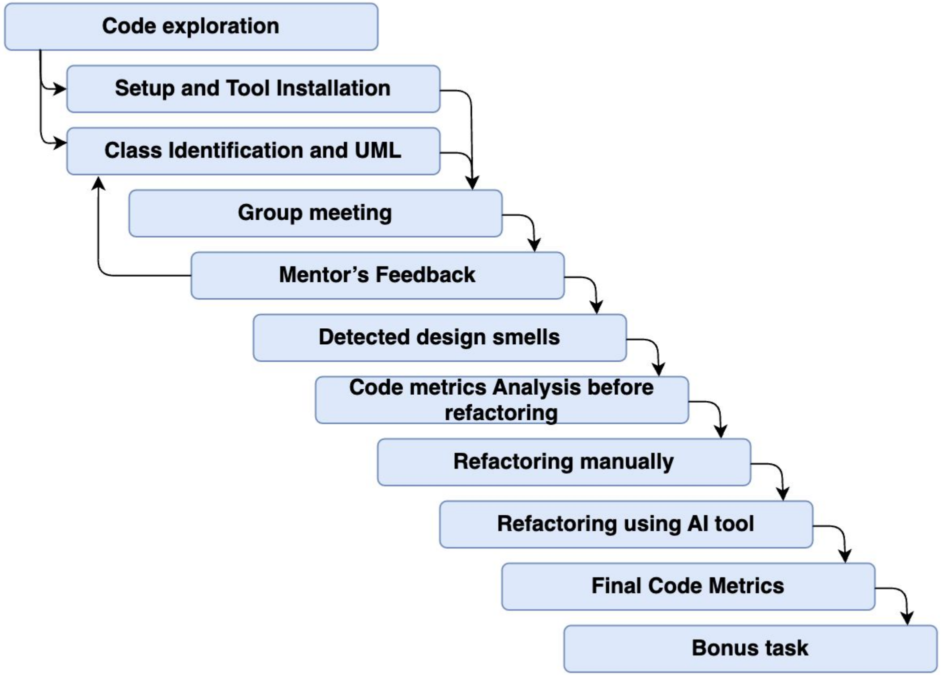
- Karan Bhatt
- Vedashree Ranade
- Madhusree Bera
- Yash Maheshwari
- Piyush Rana

Mentor

- Shubham Kulkarni

Team Dynamics and Process flow

Name	Contribution
Karan Bhatt	UML, Design Smell, Code Refactor, Report
Madhusree Bera	UML, Design Smell, Code Refactor, Report
Vedashree Ranade	UML, Design Smell, Code Refactor, Report
Yash Maheshwari	Code Metrics, Design Smell, Code Refactor, Report
Piyush Rana	Code Metrics, Design Smell, Code Refactor, Report



Code exploration

Setup and Tool Installation

Class Identification and UML

Group meeting

Mentor’s Feedback

Detected design smells

Code metrics Analysis before refactoring

Refactoring manually

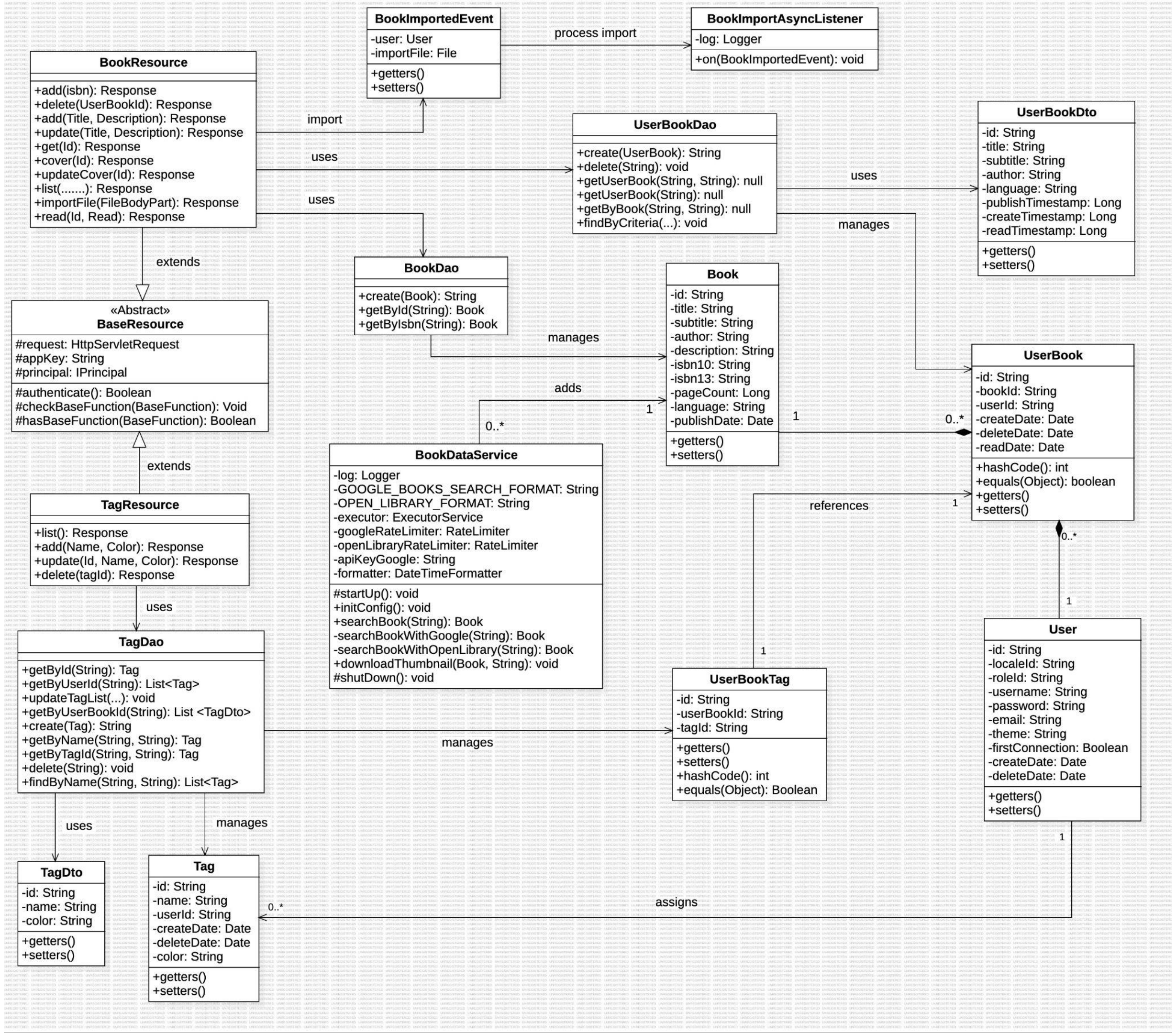
Refactoring using AI tool

Final Code Metrics

Bonus task

Relevant Class Identification
and
UML Diagram

Book Addition & Display Subsystem



BookImportedEvent
-user: User
-importFile: File
+getters()
+setters()

BookImportAsyncListener
-log: Logger
+on(BookImportedEvent): void

BookResource
+add(isbn): Response
+delete(UserBookId): Response
+add(Title, Description): Response
+update(Title, Description): Response
+get(Id): Response
+cover(Id): Response
+updateCover(Id): Response
+list(.......): Response
+importFile(FileBodyPart): Response
+read(Id, Read): Response

UserBookDao
+create(UserBook): String
+delete(String): void
+getUserBook(String, String): null
+getUserBook(String): null
+getByBook(String, String): null
+findByCriteria(...): void

UserBookDto
-id: String
-title: String
-subtitle: String
-author: String
-language: String
-publishTimestamp: Long
-createTimestamp: Long
-readTimestamp: Long
+getters()
+setters()

BookDao
+create(Book): String
+getById(String): Book
+getByIsbn(String): Book

Book
-id: String
-title: String
-subtitle: String
-author: String
-description: String
-isbn10: String
-isbn13: String
-pageCount: Long
-language: String
-publishDate: Date
+getters()
+setters()

«Abstract»
BaseResource
#request: HttpServletRequest
#appKey: String
#principal: IPrincipal
#authenticate(): Boolean
#checkBaseFunction(BaseFunction): Void
#hasBaseFunction(BaseFunction): Boolean

UserBook
-id: String
-bookId: String
-userId: String
-createDate: Date
-deleteDate: Date
-readDate: Date
+hashCode(): int
+equals(Object): boolean
+getters()
+setters()

BookDataService
-log: Logger
-GOOGLE_BOOKS_SEARCH_FORMAT: String
-OPEN_LIBRARY_FORMAT: String
-executor: ExecutorService
-googleRateLimiter: RateLimiter
-openLibraryRateLimiter: RateLimiter
-apiKeyGoogle: String
-formatter: DateTimeFormatter
#startUp(): void
+initConfig(): void
+searchBook(String): Book
-searchBookWithGoogle(String): Book
-searchBookWithOpenLibrary(String): Book
+downloadThumbnail(Book, String): void
#shutDown(): void

TagResource
+list(): Response
+add(Name, Color): Response
+update(Id, Name, Color): Response
+delete(tagId): Response

TagDao
+getById(String): Tag
+getByUserId(String): List<Tag>
+updateTagList(...): void
+getByUserBookId(String): List <TagDto>
+create(Tag): String
+getByName(String, String): Tag
+getByTagId(String, String): Tag
+delete(String): void
+findByName(String, String): List<Tag>

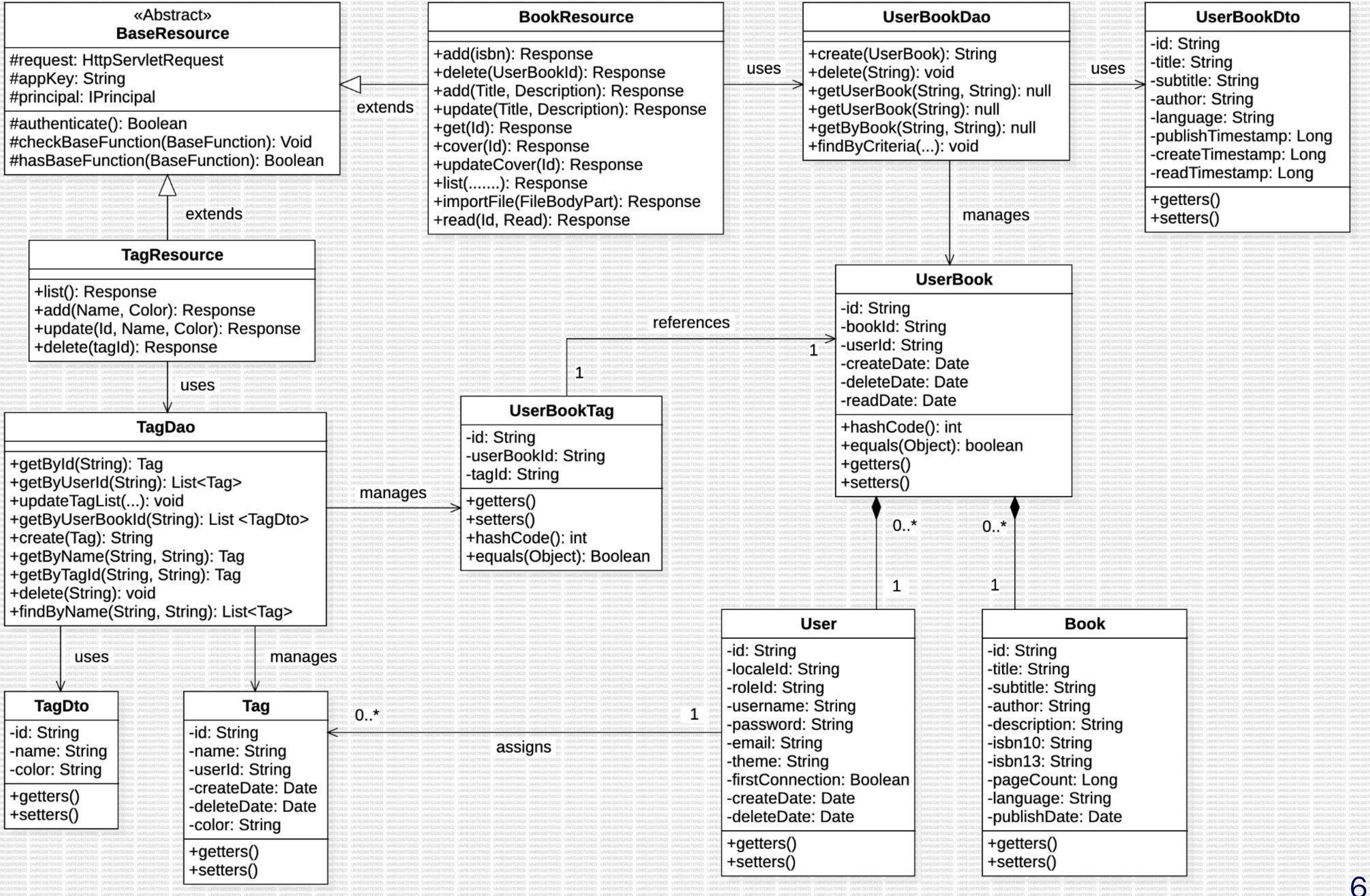
UserBookTag
-id: String
-userBookId: String
-tagId: String
+getters()
+setters()
+hashCode(): int
+equals(Object): Boolean

User
-id: String
-localeId: String
-roleId: String
-username: String
-password: String
-email: String
-theme: String
-firstConnection: Boolean
-createDate: Date
-deleteDate: Date
+getters()
+setters()

TagDto
-id: String
-name: String
-color: String
+getters()
+setters()

Tag
-id: String
-name: String
-userId: String
-createDate: Date
-deleteDate: Date
-color: String
+getters()
+setters()

Bookshelf Management Subsystem



«Abstract»
BaseResource

#request: HttpServletRequest
#appKey: String
#principal: IPrincipal

#authenticate(): Boolean
#checkBaseFunction(BaseFunction): Void
#hasBaseFunction(BaseFunction): Boolean

BookResource

+add(isbn): Response
+delete(UserBookId): Response
+add(Title, Description): Response
+update(Title, Description): Response
+get(Id): Response
+cover(Id): Response
+updateCover(Id): Response
+list(.......): Response
+importFile(FileBodyPart): Response
+read(Id, Read): Response

UserBookDao

+create(UserBook): String
+delete(String): void
+getUserBook(String, String): null
+getUserBook(String): null
+getByBook(String, String): null
+findByCriteria(...): void

UserBookDto

-id: String
-title: String
-subtitle: String
-author: String
-language: String
-publishTimestamp: Long
-createTimestamp: Long
-readTimestamp: Long

+getters()
+setters()

TagResource

+list(): Response
+add(Name, Color): Response
+update(Id, Name, Color): Response
+delete(tagId): Response

UserBook

-id: String
-bookId: String
-userId: String
-createDate: Date
-deleteDate: Date
-readDate: Date

+hashCode(): int
+equals(Object): boolean
+getters()
+setters()

UserBookTag

-id: String
-userBookId: String
-tagId: String

+getters()
+setters()
+hashCode(): int
+equals(Object): Boolean

TagDao

+getById(String): Tag
+getByUserId(String): List<Tag>
+updateTagList(...): void
+getByUserBookId(String): List <TagDto>
+create(Tag): String
+getByName(String, String): Tag
+getByTagId(String, String): Tag
+delete(String): void
+findByName(String, String): List<Tag>

TagDto

-id: String
-name: String
-color: String

+getters()
+setters()

Tag

-id: String
-name: String
-userId: String
-createDate: Date
-deleteDate: Date
-color: String

+getters()
+setters()

User

-id: String
-localeId: String
-roleId: String
-username: String
-password: String
-email: String
-theme: String
-firstConnection: Boolean
-createDate: Date
-deleteDate: Date

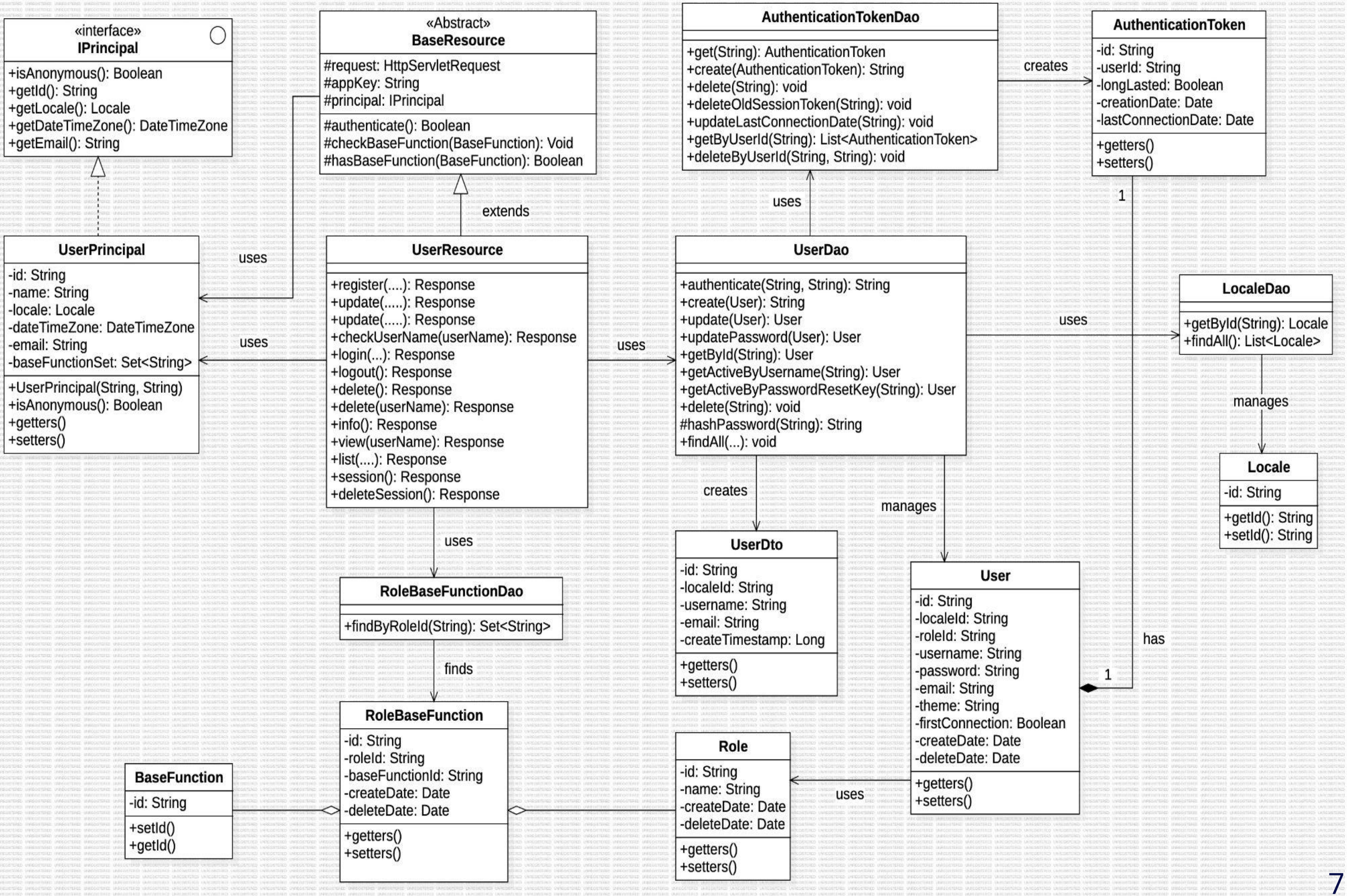
+getters()
+setters()

Book

-id: String
-title: String
-subtitle: String
-author: String
-description: String
-isbn10: String
-isbn13: String
-pageCount: Long
-language: String
-publishDate: Date

+getters()
+setters()

User Management Subsystem



«interface»
IPrincipal

+isAnonymous(): Boolean
+getId(): String
+getLocale(): Locale
+getDateTimeZone(): DateTimeZone
+getEmail(): String

«Abstract»
BaseResource

#request: HttpServletRequest
#appKey: String
#principal: IPrincipal

#authenticate(): Boolean
#checkBaseFunction(BaseFunction): Void
#hasBaseFunction(BaseFunction): Boolean

AuthenticationTokenDao

+get(String): AuthenticationToken
+create(AuthenticationToken): String
+delete(String): void
+deleteOldSessionToken(String): void
+updateLastConnectionDate(String): void
+getByUserId(String): List<AuthenticationToken>
+deleteByUserId(String, String): void

AuthenticationToken

-id: String
-userId: String
-longLasted: Boolean
-creationDate: Date
-lastConnectionDate: Date

+getters()
+setters()

UserPrincipal

-id: String
-name: String
-locale: Locale
-dateTimeZone: DateTimeZone
-email: String
-baseFunctionSet: Set<String>

+UserPrincipal(String, String)
+isAnonymous(): Boolean
+getters()
+setters()

UserResource

+register(....): Response
+update(.....): Response
+update(.....): Response
+checkUserName(userName): Response
+login(...): Response
+logout(): Response
+delete(): Response
+delete(userName): Response
+info(): Response
+view(userName): Response
+list(....): Response
+session(): Response
+deleteSession(): Response

UserDao

+authenticate(String, String): String
+create(User): String
+update(User): User
+updatePassword(User): User
+getById(String): User
+getActiveByUsername(String): User
+getActiveByPasswordResetKey(String): User
+delete(String): void
#hashPassword(String): String
+findAll(...): void

LocaleDao

+getById(String): Locale
+findAll(): List<Locale>

Locale

-id: String

+getId(): String
+setId(): String

RoleBaseFunctionDao

+findByRoleId(String): Set<String>

UserDto

-id: String
-localeId: String
-username: String
-email: String
-createTimestamp: Long

+getters()
+setters()

User

-id: String
-localeId: String
-roleId: String
-username: String
-password: String
-email: String
-theme: String
-firstConnection: Boolean
-createDate: Date
-deleteDate: Date

+getters()
+setters()

RoleBaseFunction

-id: String
-roleId: String
-baseFunctionId: String
-createDate: Date
-deleteDate: Date

+getters()
+setters()

BaseFunction

-id: String

+setId()
+getId()

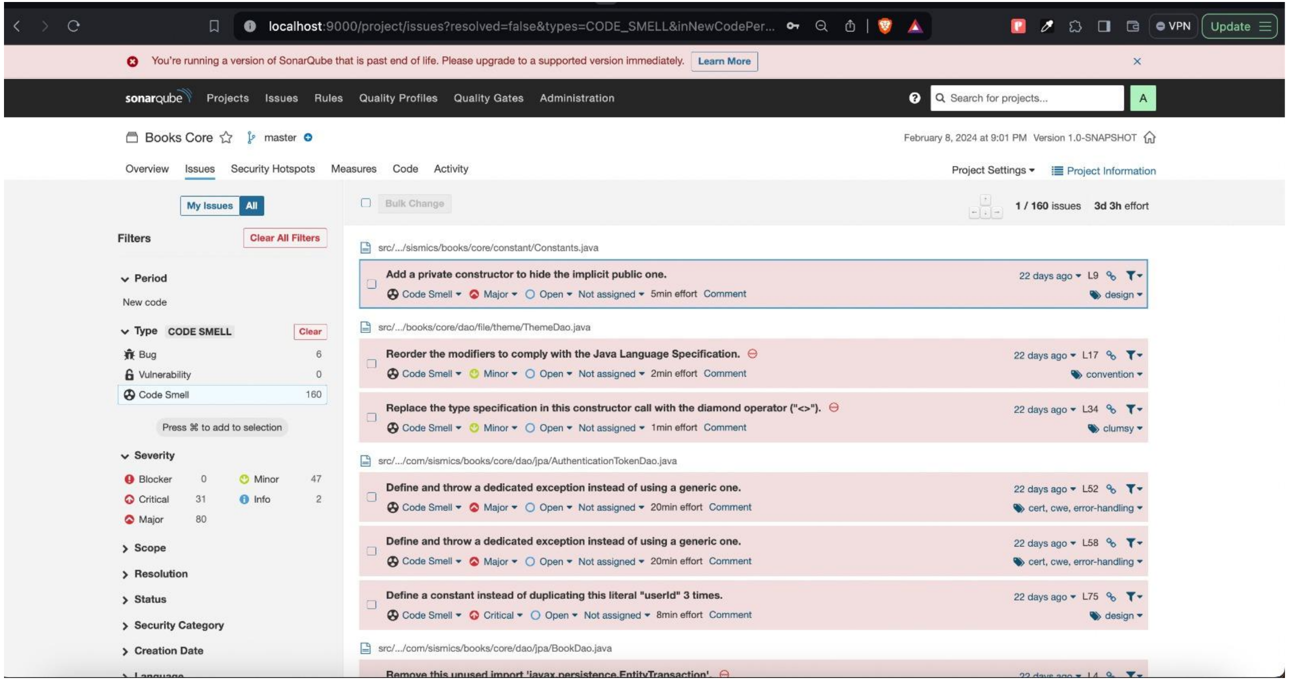
Role

-id: String
-name: String
-createDate: Date
-deleteDate: Date

+getters()
+setters()

Design smell detection
and
Refactoring

SonarQube: Code Smell detection



You're running a version of SonarQube that is past end of life. Please upgrade to a supported version immediately.

Books Core · master

February 8, 2024 at 9:01 PM Version 1.0-SNAPSHOT

1 / 160 issues 3d 3h effort

Filters

Period: New code

Type: CODE SMELL

Type	Count
Bug	6
Vulnerability	0
Code Smell	160

Press ⌘ to add to selection

Severity

Severity	Count	Severity	Count
Blocker	0	Minor	47
Critical	31	Info	2
Major	80		

src/.../sismics/books/core/constant/Constants.java

- Add a private constructor to hide the implicit public one. — Code Smell · Major · Open · Not assigned · 5min effort · Comment — 22 days ago · L9 · design

src/.../books/core/dao/file/theme/ThemeDao.java

- Reorder the modifiers to comply with the Java Language Specification. — Code Smell · Minor · Open · Not assigned · 2min effort · Comment — 22 days ago · L17 · convention
- Replace the type specification in this constructor call with the diamond operator ("<>"). — Code Smell · Minor · Open · Not assigned · 1min effort · Comment — 22 days ago · L34 · clumsy

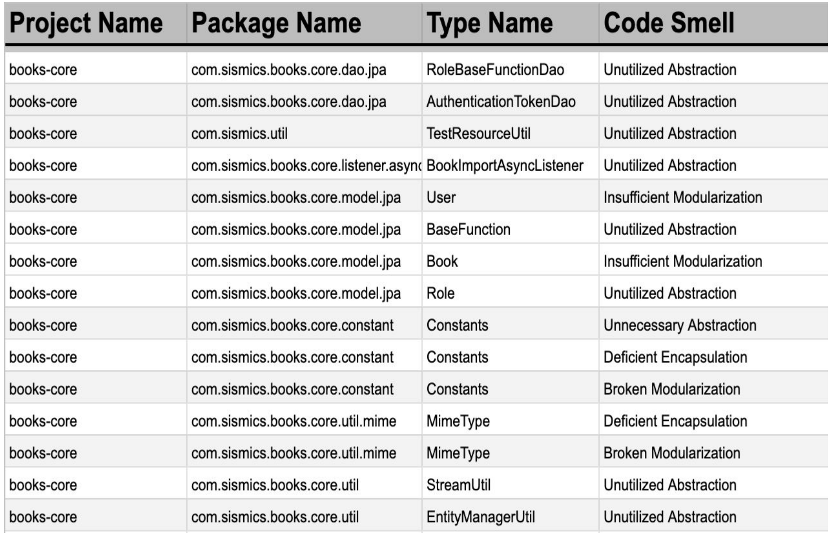
src/.../com/sismics/books/core/dao/jpa/AuthenticationTokenDao.java

- Define and throw a dedicated exception instead of using a generic one. — Code Smell · Major · Open · Not assigned · 20min effort · Comment — 22 days ago · L52 · cert, cwe, error-handling
- Define and throw a dedicated exception instead of using a generic one. — Code Smell · Major · Open · Not assigned · 20min effort · Comment — 22 days ago · L58 · cert, cwe, error-handling
- Define a constant instead of duplicating this literal "userId" 3 times. — Code Smell · Critical · Open · Not assigned · 8min effort · Comment — 22 days ago · L75 · design

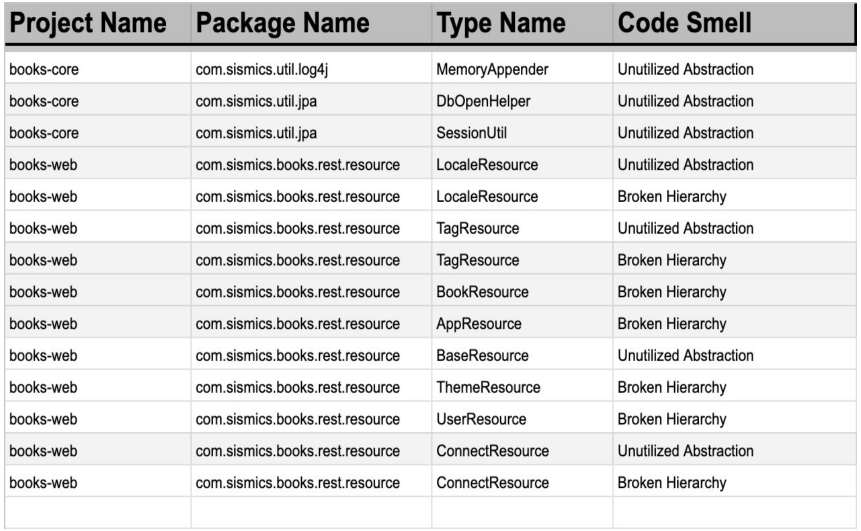
src/.../com/sismics/books/core/dao/jpa/BookDao.java

- Remove this unused import 'javax.persistence.EntityTransaction'. — 22 days ago · L4

Designite Java: Design Smell detection

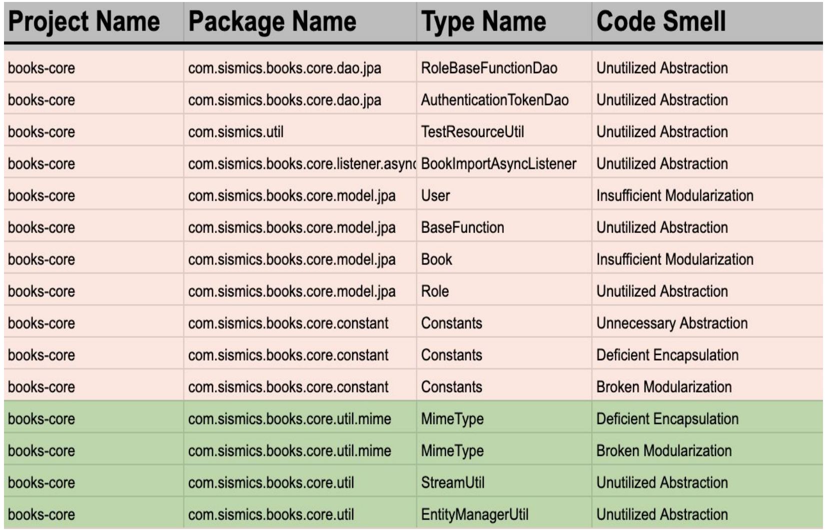


Project Name	Package Name	Type Name	Code Smell
books-core	com.sismics.books.core.dao.jpa	RoleBaseFunctionDao	Unutilized Abstraction
books-core	com.sismics.books.core.dao.jpa	AuthenticationTokenDao	Unutilized Abstraction
books-core	com.sismics.util	TestResourceUtil	Unutilized Abstraction
books-core	com.sismics.books.core.listener.async	BookImportAsyncListener	Unutilized Abstraction
books-core	com.sismics.books.core.model.jpa	User	Insufficient Modularization
books-core	com.sismics.books.core.model.jpa	BaseFunction	Unutilized Abstraction
books-core	com.sismics.books.core.model.jpa	Book	Insufficient Modularization
books-core	com.sismics.books.core.model.jpa	Role	Unutilized Abstraction
books-core	com.sismics.books.core.constant	Constants	Unnecessary Abstraction
books-core	com.sismics.books.core.constant	Constants	Deficient Encapsulation
books-core	com.sismics.books.core.constant	Constants	Broken Modularization
books-core	com.sismics.books.core.util.mime	MimeType	Deficient Encapsulation
books-core	com.sismics.books.core.util.mime	MimeType	Broken Modularization
books-core	com.sismics.books.core.util	StreamUtil	Unutilized Abstraction
books-core	com.sismics.books.core.util	EntityManagerUtil	Unutilized Abstraction

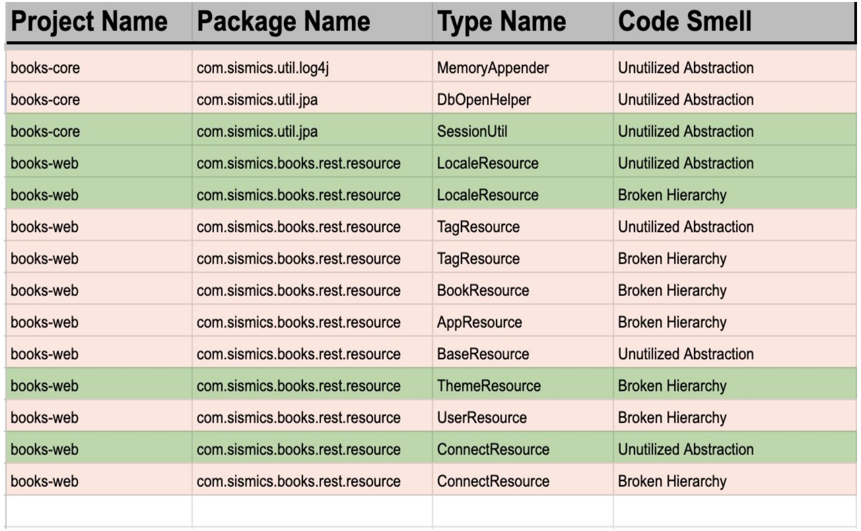


Project Name	Package Name	Type Name	Code Smell
books-core	com.sismics.util.log4j	MemoryAppender	Unutilized Abstraction
books-core	com.sismics.util.jpa	DbOpenHelper	Unutilized Abstraction
books-core	com.sismics.util.jpa	SessionUtil	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	LocaleResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	LocaleResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	TagResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	TagResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	BookResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	AppResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	BaseResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	ThemeResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	UserResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	ConnectResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	ConnectResource	Broken Hierarchy

Designite Java: Design Smell detection



Project Name	Package Name	Type Name	Code Smell
books-core	com.sismics.books.core.dao.jpa	RoleBaseFunctionDao	Unutilized Abstraction
books-core	com.sismics.books.core.dao.jpa	AuthenticationTokenDao	Unutilized Abstraction
books-core	com.sismics.util	TestResourceUtil	Unutilized Abstraction
books-core	com.sismics.books.core.listener.asyn	BookImportAsyncListener	Unutilized Abstraction
books-core	com.sismics.books.core.model.jpa	User	Insufficient Modularization
books-core	com.sismics.books.core.model.jpa	BaseFunction	Unutilized Abstraction
books-core	com.sismics.books.core.model.jpa	Book	Insufficient Modularization
books-core	com.sismics.books.core.model.jpa	Role	Unutilized Abstraction
books-core	com.sismics.books.core.constant	Constants	Unnecessary Abstraction
books-core	com.sismics.books.core.constant	Constants	Deficient Encapsulation
books-core	com.sismics.books.core.constant	Constants	Broken Modularization
books-core	com.sismics.books.core.util.mime	MimeType	Deficient Encapsulation
books-core	com.sismics.books.core.util.mime	MimeType	Broken Modularization
books-core	com.sismics.books.core.util	StreamUtil	Unutilized Abstraction
books-core	com.sismics.books.core.util	EntityManagerUtil	Unutilized Abstraction

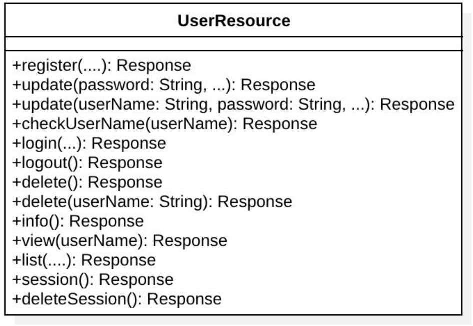


Project Name	Package Name	Type Name	Code Smell
books-core	com.sismics.util.log4j	MemoryAppender	Unutilized Abstraction
books-core	com.sismics.util.jpa	DbOpenHelper	Unutilized Abstraction
books-core	com.sismics.util.jpa	SessionUtil	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	LocaleResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	LocaleResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	TagResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	TagResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	BookResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	AppResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	BaseResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	ThemeResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	UserResource	Broken Hierarchy
books-web	com.sismics.books.rest.resource	ConnectResource	Unutilized Abstraction
books-web	com.sismics.books.rest.resource	ConnectResource	Broken Hierarchy

Green: Correct design smells after manual inspection
Orange: Irrelevant design smell after manual inspection

Design Smell 1: Missing Hierarchy

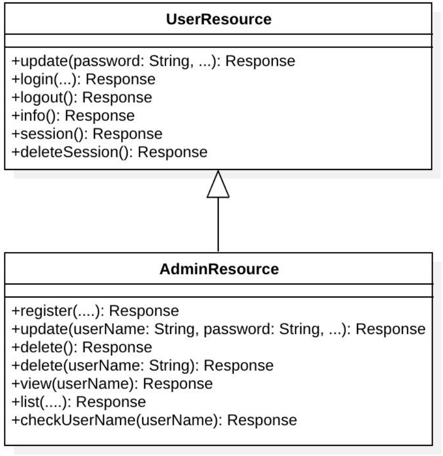
- UserResource has functions of both User and Admin type all in one class.
- **If-conditions** are used to check if the current user has Admin permissions
- Design smell because when more user types will be added, it will **increase the number of methods** and **number of if-conditions** for each role
- **Solution** is to create a **separate class for AdminResource**. Since Admin "IS-A" UserResource, so AdminResource class should extend UserResource class



UserResource

+register(....): Response
+update(password: String, ...): Response
+update(userName: String, password: String, ...): Response
+checkUserName(userName): Response
+login(...): Response
+logout(): Response
+delete(): Response
+delete(userName: String): Response
+info(): Response
+view(userName): Response
+list(....): Response
+session(): Response
+deleteSession(): Response

Design smell



UserResource

+update(password: String, ...): Response
+login(...): Response
+logout(): Response
+info(): Response
+session(): Response
+deleteSession(): Response

AdminResource

+register(....): Response
+update(userName: String, password: String, ...): Response
+delete(): Response
+delete(userName: String): Response
+view(userName): Response
+list(....): Response
+checkUserName(userName): Response

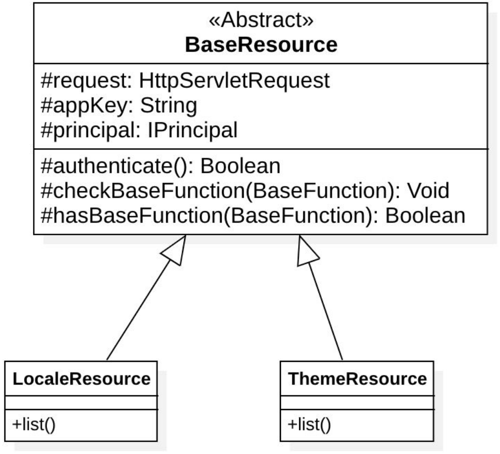
Proposed solution

Design Smell 2: Broken Hierarchy

- This smell arises when a supertype and its subtype conceptually **do not share an "IS-A"** relationship resulting in broken substitutability.

- LocaleResource and ThemeResource extend BaseResource but **do not utilize any function of BaseResource**.

- **Refactored** the code to remove the inheritance from BaseResource



«Abstract»
BaseResource

#request: HttpServletRequest
#appKey: String
#principal: IPrincipal

#authenticate(): Boolean
#checkBaseFunction(BaseFunction): Void
#hasBaseFunction(BaseFunction): Boolean

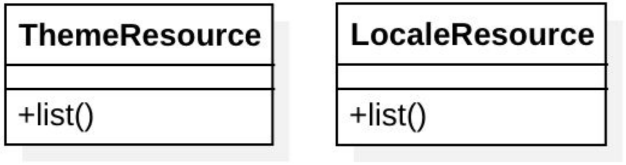
LocaleResource

+list()

ThemeResource

+list()

before refactoring



ThemeResource

+list()

LocaleResource

+list()

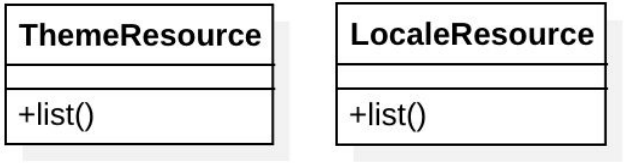
after refactoring

Design Smell 3: Missing Interface hierarchy

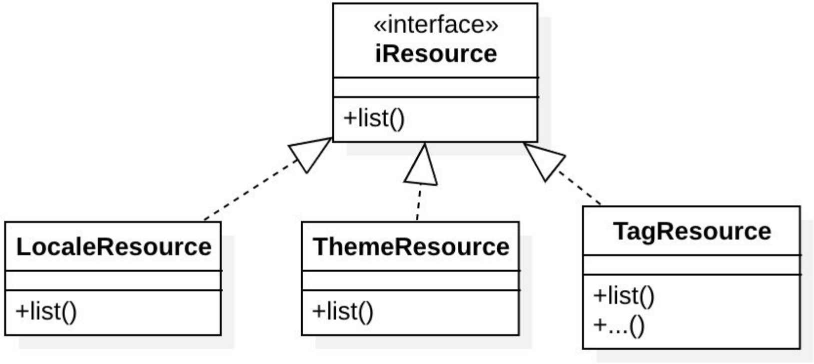
- The classes LocaleResource and ThemeResource both have only one method **list()**. This shows that they both can implement a **common interface**.

- In future, such common functions of both ThemeResource and LocaleResource can be added to the same interface.

- **Refactored** the code to add an **interface named IResource** which is implemented by LocaleResource, ThemeResource and TagResource.



before refactoring

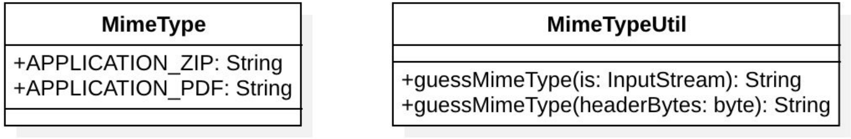


after refactoring

Design Smell 4: Broken Modularization

- This smell arises when data and/or methods that ideally **should have been localized into a single abstraction are separated** and spread across multiple abstractions.

- **Refactored** the code to extract and **move the functions** from MimeTypeUtil to MimeType class



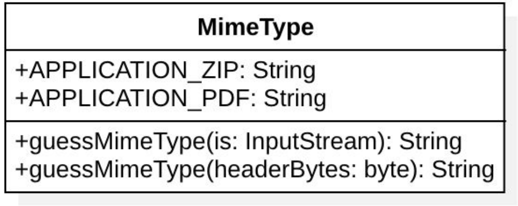
MimeType

+APPLICATION_ZIP: String
+APPLICATION_PDF: String

MimeTypeUtil

+guessMimeType(is: InputStream): String
+guessMimeType(headerBytes: byte): String

before refactoring



MimeType

+APPLICATION_ZIP: String
+APPLICATION_PDF: String

+guessMimeType(is: InputStream): String
+guessMimeType(headerBytes: byte): String

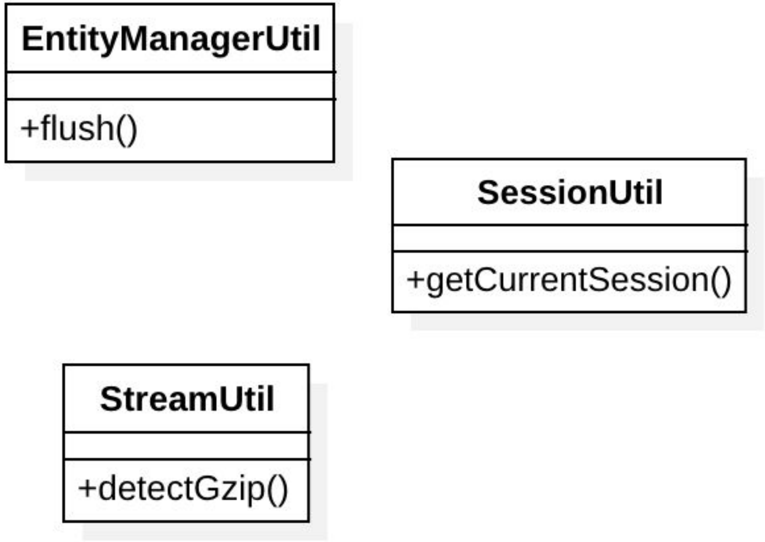
after refactoring

Design Smell 5: Imperative Abstraction

- Imperative Abstraction is indicated when **an operation is turned into a class that has only one method** defined in it.

- These functions are **not utilised anywhere** as well

- **Refactored** the code to **remove** the unused classes and functions or **move the function to a suitable class** following the **Information Expert** principle



before refactoring

Code Metrics Analysis

Code Metrics Analysis

01 Tools Used

02 Before Refactoring

03 After Refactoring

04 Implication Discussion

Tools Used

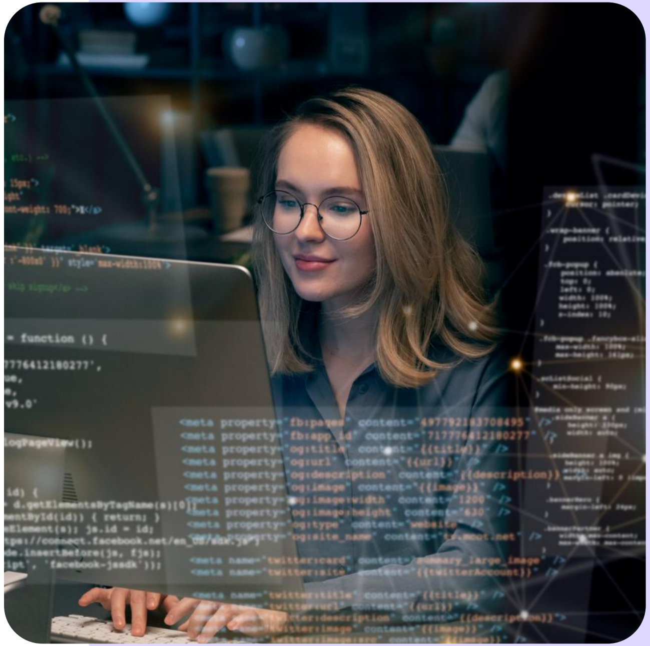
We majorly used these tools for Code Metric Analysis:

- **SonarQube:** It is used for Code quality and Code Analysis. It gives mainly three Code Metrics, viz., Line of Code, Cyclomatic Complexity and Code Duplication.

- **CodeMR:** It allows users to visualize their software architecture using different views such as TreeMap, Dependency views, etc.

  It not only evaluates our code quality but also offers in-depth visualizations and analysis capabilities

- **Designite:** Designite is a used for assessing the quality of Java code, providing valuable insights through metrics analysis. It helps us to identify and address issues to enhance the overall performance and maintainability of the codebase.



Code Metric Impact on Software Quality

We know what these Code Metrics are. We will mainly focus on their Impact on Software Quality and Potential Performance Issues

- **Line of Code:** A higher LOC can indicate a potentially more complex project, which may be harder to maintain. However, LOC alone isn't a reliable measure of quality. It's possible to have a small codebase that's poorly written or a large one that's well-organised and maintainable. While LOC doesn't directly impact performance, a larger codebase might lead to longer compile times and potentially more bugs, which can indirectly affect performance.
- **Cyclomatic Complexity:** Cyclomatic complexity can be interpreted as the number of linearly independent paths through the source code. In simpler terms, it represents the number of different ways a program can be executed.

$$M=E-N+2P$$

High cyclomatic complexity indicates a more complex and potentially less maintainable codebase. Complex code is harder to understand, test, and modify, increasing the risk of bugs. While not directly related to performance, complex code can lead to inefficient algorithms that degrade performance. Also, complex code might be more challenging to optimise.

Code Metric Impact on Software Quality

Coupling Between Object Classes: CBO ia a measure to evaluate how much one class in a program depends on (or connected to) other classes via direct references or Method calls.

High coupling makes a system more brittle and harder to change because modifications in one class can require changes in all coupled classes. Lower coupling enhances modularity, making the system easier to understand, modify, and maintain. High coupling can lead to performance issues since changes in one part of the system may have widespread effects, requiring more extensive testing and potentially leading to less efficient code execution.

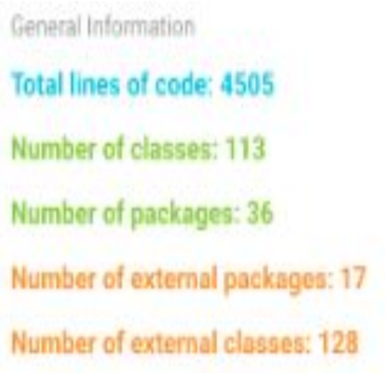
Code Metric Analysis after Refactoring

Line of Code:

Before

While start working on this Code Metric, we observed that we have **4505** Lines of Code in the given Repo.

We Ignore some redundant files like files located in any 'test' directories within the 'src' directory.



General Information

Total lines of code: 4505

Number of classes: 113

Number of packages: 36

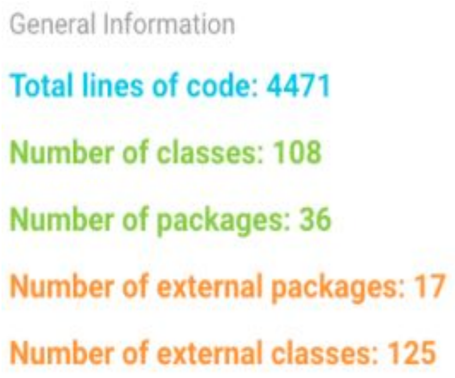
Number of external packages: 17

Number of external classes: 128

After

After Refactoring, we reduced our lines of Code by **4471**.

While going through the Codebase, we found some duplicate Codes and some dead codes with no use at all. So we refactored them, and thus, our Lines of Code decreases.



General Information

Total lines of code: 4471

Number of classes: 108

Number of packages: 36

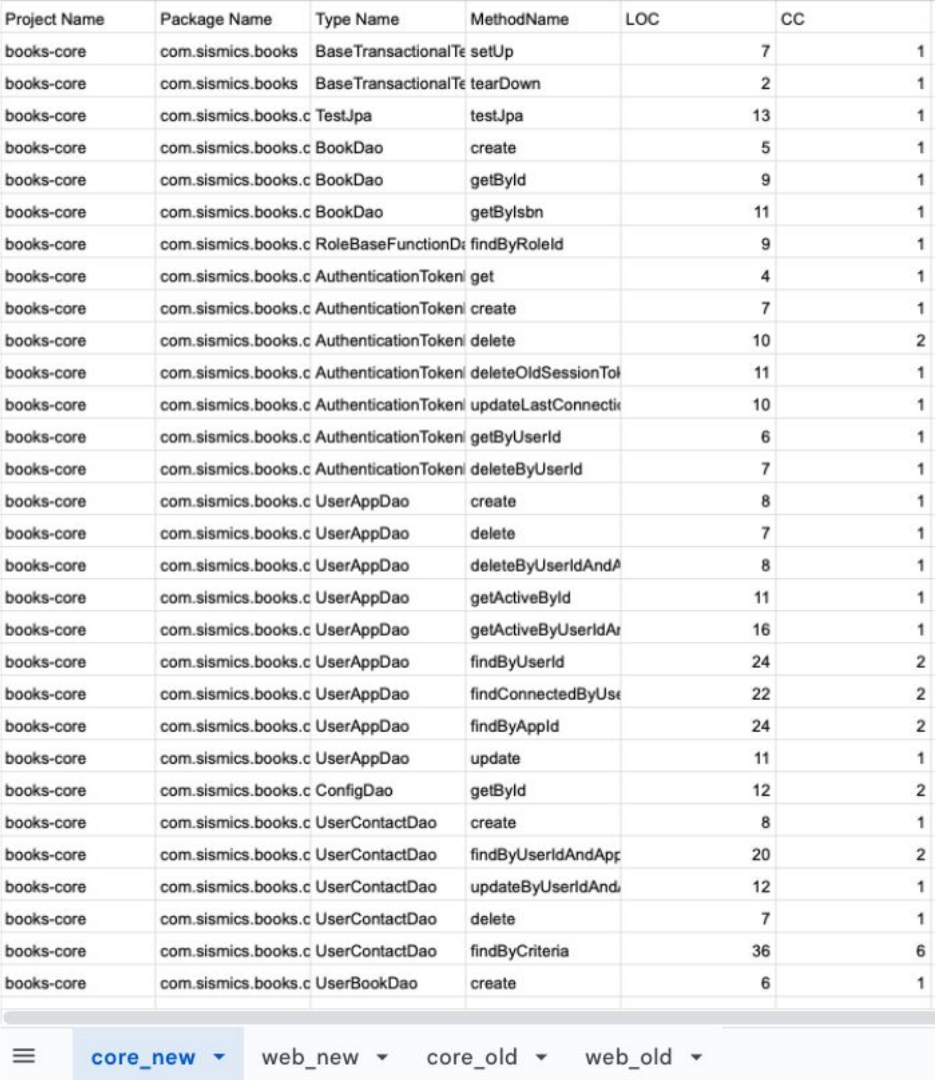
Number of external packages: 17

Number of external classes: 125

Code Metric Analysis after Refactoring

Cyclomatic Complexity:

Using Designite Java, we found the cyclomatic complexity before and after the refactoring on each modules of our Repo, and we didn’t find any significant Change in this Code Metric.

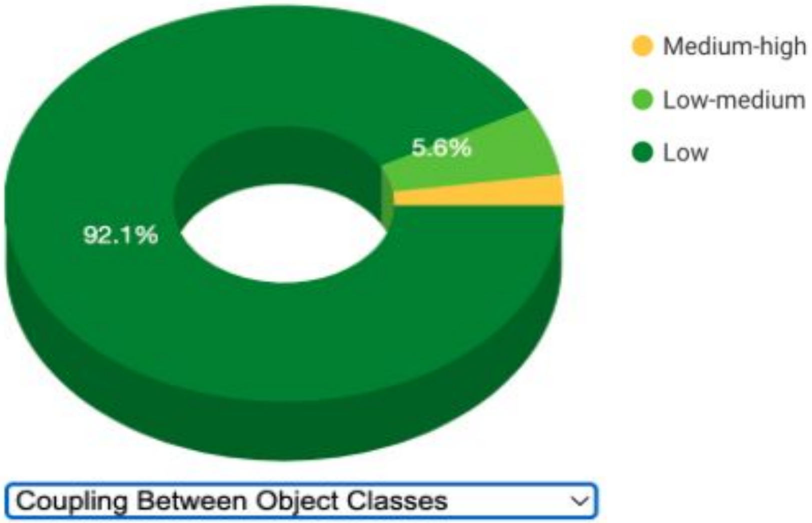


Project Name	Package Name	Type Name	MethodName	LOC	CC
books-core	com.sismics.books	BaseTransactionalTe	setUp	7	1
books-core	com.sismics.books	BaseTransactionalTe	tearDown	2	1
books-core	com.sismics.books.c	TestJpa	testJpa	13	1
books-core	com.sismics.books.c	BookDao	create	5	1
books-core	com.sismics.books.c	BookDao	getById	9	1
books-core	com.sismics.books.c	BookDao	getByIsbn	11	1
books-core	com.sismics.books.c	RoleBaseFunctionDa	findByRoleId	9	1
books-core	com.sismics.books.c	AuthenticationToken	get	4	1
books-core	com.sismics.books.c	AuthenticationToken	create	7	1
books-core	com.sismics.books.c	AuthenticationToken	delete	10	2
books-core	com.sismics.books.c	AuthenticationToken	deleteOldSessionTok	11	1
books-core	com.sismics.books.c	AuthenticationToken	updateLastConnecti	10	1
books-core	com.sismics.books.c	AuthenticationToken	getByUserId	6	1
books-core	com.sismics.books.c	AuthenticationToken	deleteByUserId	7	1
books-core	com.sismics.books.c	UserAppDao	create	8	1
books-core	com.sismics.books.c	UserAppDao	delete	7	1
books-core	com.sismics.books.c	UserAppDao	deleteByUserIdAndA	8	1
books-core	com.sismics.books.c	UserAppDao	getActiveById	11	1
books-core	com.sismics.books.c	UserAppDao	getActiveByUserIdAr	16	1
books-core	com.sismics.books.c	UserAppDao	findByUserId	24	2
books-core	com.sismics.books.c	UserAppDao	findConnectedByUse	22	2
books-core	com.sismics.books.c	UserAppDao	findByAppId	24	2
books-core	com.sismics.books.c	UserAppDao	update	11	1
books-core	com.sismics.books.c	ConfigDao	getById	12	2
books-core	com.sismics.books.c	UserContactDao	create	8	1
books-core	com.sismics.books.c	UserContactDao	findByUserIdAndApp	20	2
books-core	com.sismics.books.c	UserContactDao	updateByUserIdAndA	12	1
books-core	com.sismics.books.c	UserContactDao	delete	7	1
books-core	com.sismics.books.c	UserContactDao	findByCriteria	36	6
books-core	com.sismics.books.c	UserBookDao	create	6	1

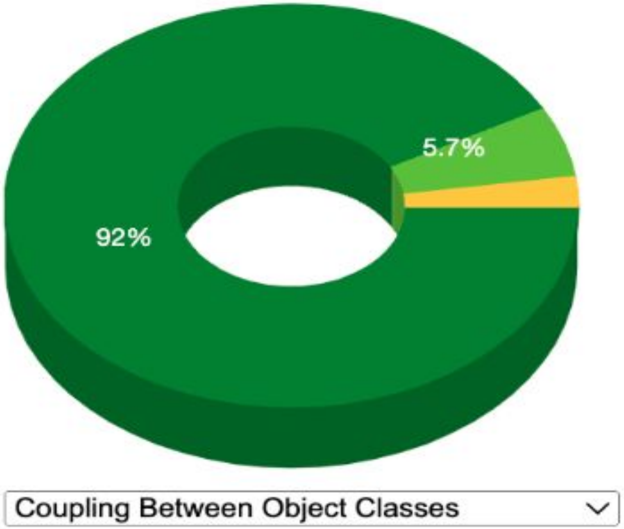
Code Metric Analysis after Refactoring

Coupling between Object Class:

Before



After



Code Metric Impact on Software Quality

Percentage of Code Duplication: or PC is a code metric that measures the extent to which identical or highly similar code segments exist within a codebase.

When the same or similar functionality is duplicated across multiple places in the code, it becomes challenging to make changes or updates consistently and take more efforts in maintaining copies of the same logic. it can leads to inconsistency as we have to make change in every copy of duplicated code.

PC = (Size of Duplicated Code / Total Size of Code) × 100

Lack of Cohesion of Methods: LCOM is a code metric used to evaluate the degree of cohesion within a class by assessing the relationships between its methods(intra-relationship). Cohesion refers to how closely the functionalities of methods within a class are related to each other. LCOM specifically focuses on the lack of such relationships, indicating a potential design issue in the class.

LCOM = Number of Method Pairs without Shared Variables − Number of Method Pairs with Shared Variables

Classes with low cohesion (high LCOM) can be harder to maintain and understand. Such classes might need to be refactored into smaller, more focused classes, improving modularity and maintainability. Indirectly, low cohesion can lead to unnecessary dependencies and overhead, as objects of the class may carry more data and behaviour than required for particular operations, potentially affecting performance.

Code Metric Analysis after Refactoring

Percentage of Code Duplication:

Before

The Sum of Total PC was 105 in Books-web and in Book-Core the PC was 273

After

The Sum of Total PC was 95 in Books-web and in Book-Core the PC was 272

The reduction in code duplication after the refactor can be attributed primarily to addressing the issue of Broken Modularization.Before refactoring, the code suffered from a lack of proper modularization, resulting in duplicated and scattered implementations across various location. During refactoring process, we made clear and well-defined modules , promoting a more cohesive and organized structure. As a result, the codebase now shows significantly reduced duplication, providing a more maintainable Consistent and easy to work code.

Code Metric Analysis after Refactoring

Lack of Cohesion of Methods

Before (TC=113)

Class with low LCOM: 64.1% => 71
Low-medium LCOM: 3.8% => 4
Medium-high LCOM: 4.9% => 5
High LCOM: 21.1% => 25
Very high LCOM: 6.2% => 8

After (TC=108)

Class with low LCOM: 63.8% => 69
Low-medium LCOM: 3.8% => 4
Medium-high LCOM: 4.9% => 5
High LCOM: 21.2% => 23
Very high LCOM: 5.9 % =>6

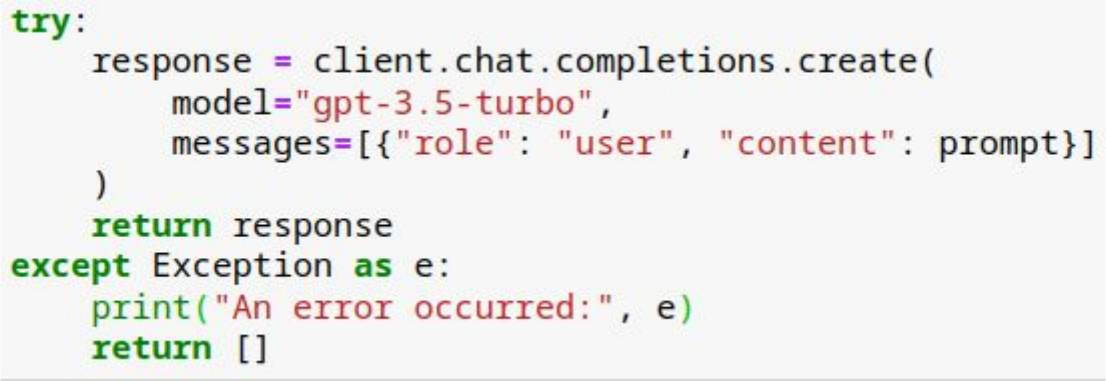
The result are as expected as we refactored the imperative abstraction classes and these are the classes which has only one method in it so having very high lcom. So we have reduce the classes with high or very high lcom as we remove the unused classes or move the function to suitable class using the information expert principal.

In the refactoring of broken Modularization also we have methods or data scattered having high lcom while ideally they should be localized into a single abstraction .

Bonus Task

Bonus Task

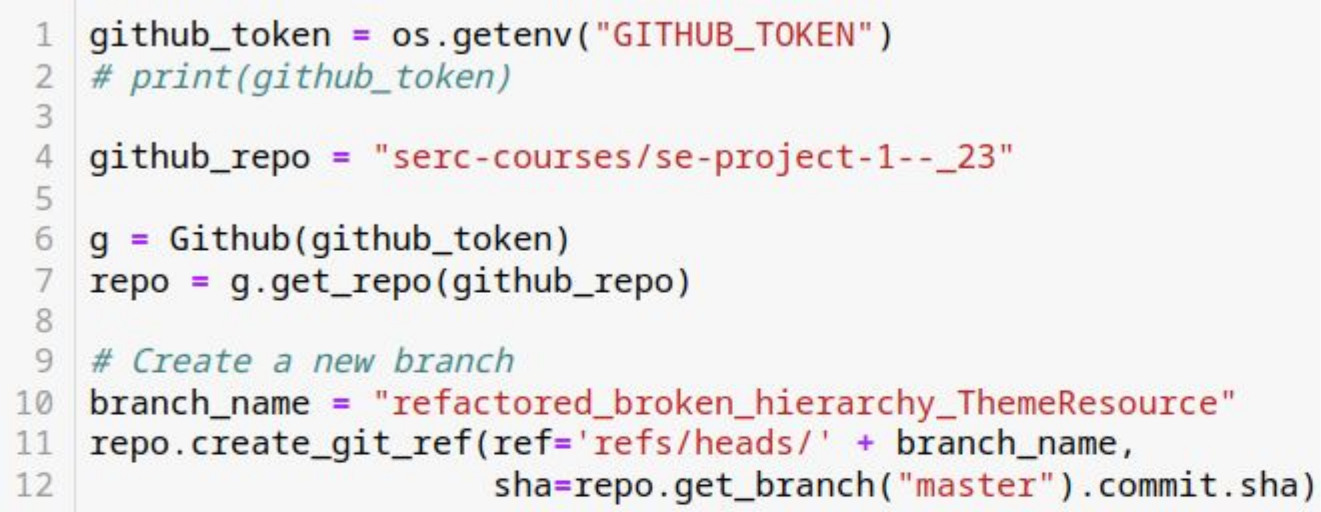
- Language and Tools used: Python 3, PyGithub library, openai library
- Steps:
  - Read the file and prompt ChatGPT (model: gpt-3.5-turbo) using ***openai*** library to detect the design smell in the appended code and refactor it.



```
try:
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )
    return response
except Exception as e:
    print("An error occurred:", e)
    return []
```

Bonus Task

- Create and switch to a new branch using ***PyGithub*** library.

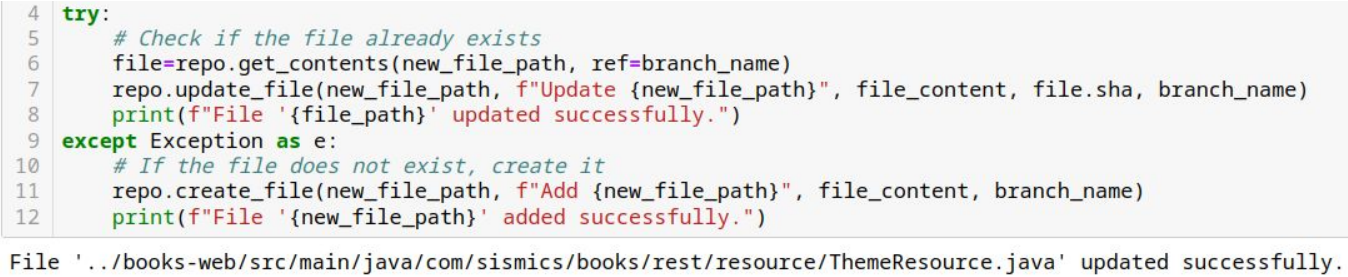


```
github_token = os.getenv("GITHUB_TOKEN")
# print(github_token)

github_repo = "serc-courses/se-project-1--_23"

g = Github(github_token)
repo = g.get_repo(github_repo)

# Create a new branch
branch_name = "refactored_broken_hierarchy_ThemeResource"
repo.create_git_ref(ref='refs/heads/' + branch_name,
                    sha=repo.get_branch("master").commit.sha)
```

Bonus Task

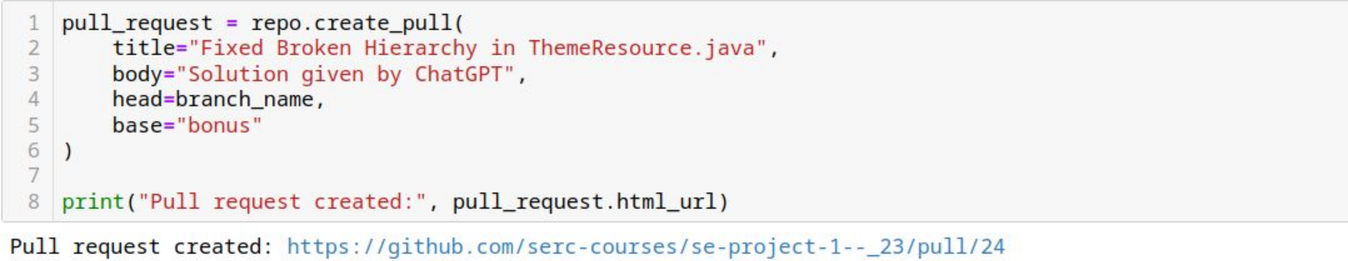
- Extract the code from the response given by ChatGPT and overwrite the file with the new code.

- Commit the file and generate Pull Request.



```python
try:
    # Check if the file already exists
    file=repo.get_contents(new_file_path, ref=branch_name)
    repo.update_file(new_file_path, f"Update {new_file_path}", file_content, file.sha, branch_name)
    print(f"File '{file_path}' updated successfully.")
except Exception as e:
    # If the file does not exist, create it
    repo.create_file(new_file_path, f"Add {new_file_path}", file_content, branch_name)
    print(f"File '{new_file_path}' added successfully.")
```

```
File '../books-web/src/main/java/com/sismics/books/rest/resource/ThemeResource.java' updated successfully.
```



```python
pull_request = repo.create_pull(
    title="Fixed Broken Hierarchy in ThemeResource.java",
    body="Solution given by ChatGPT",
    head=branch_name,
    base="bonus"
)

print("Pull request created:", pull_request.html_url)
```

```
Pull request created: https://github.com/serc-courses/se-project-1--_23/pull/24
```

Challenges faced

01 Understanding the codebase

02 Detecting correct design smells from output of tools used

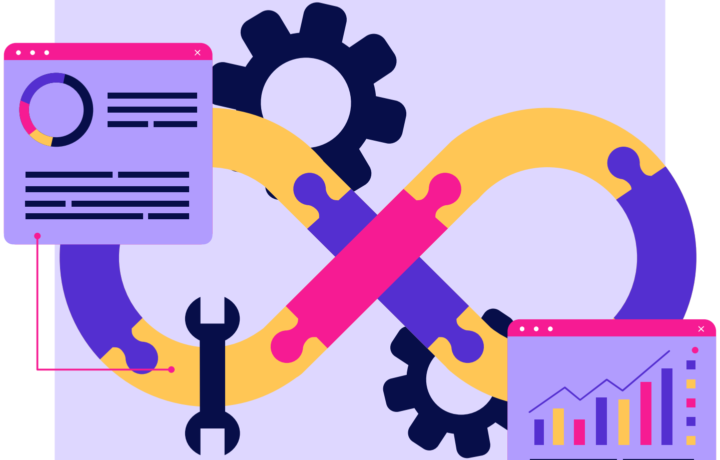
03 Prompting LLM with Design smell context

04 Refactoring codebase to add admin class

Report Link

Report: Project1 Report

Bonus Task Report: project1_bonus_23



THANKS!

Questions are welcome!