Group6 Code Review Report

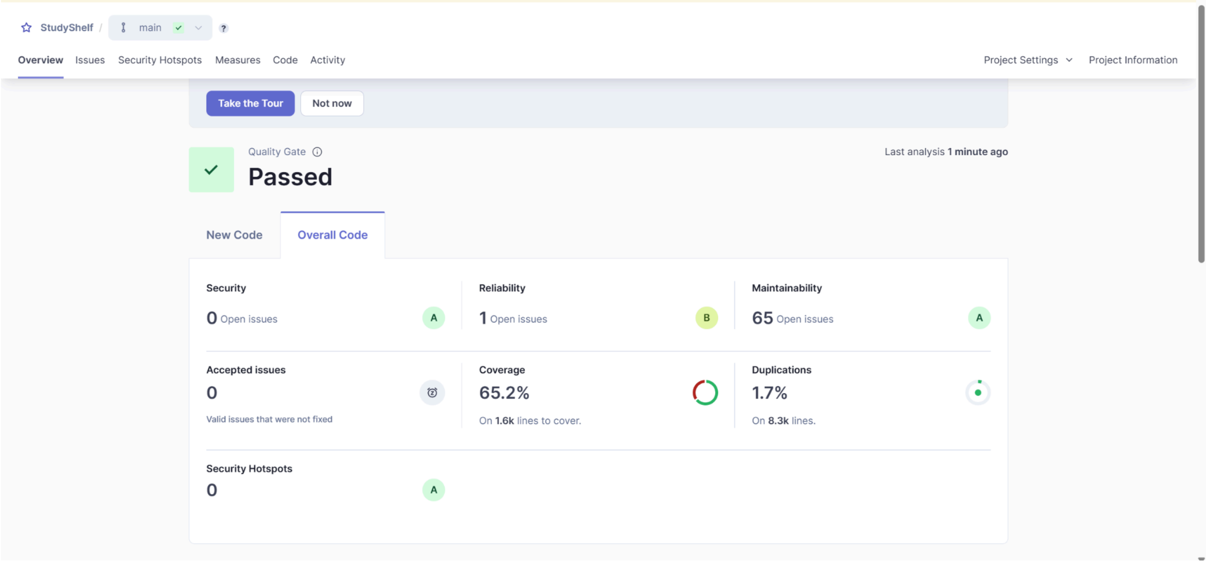
1. Tools Used

- SonarQube

2. Key Code Metrics

Metric	Grade	Value	Description
Lines of Code	N/A	5.6k	Total Java lines analyzed
Lines of code per method	N/A	13.1	The average number of lines of code per method is approximately 13.1, as indicated by SonarQube (6,203 LOC / 473 methods)
Quality Gate	Passed	Passed	Project meets the minimum quality standards
Security	A	0 Open Issues	No security vulnerabilities detected
Reliability	B	1 Open Issue	One potential bug to investigate
Maintainability	A	65 Open Issues	Minor issues found, but maintainability is rated A, indicating strong structure
Accepted issues	N/A	0	No issues accepted as “won’t fix” or excluded
Coverage	N/A	65.2%	Test coverage is below 80% but still represents a solid effort
Duplicated	N/A	1.7%	Low level of code duplication
Security Hotspots	A	0	No security hotspots detected
Cognitive Complexity	N/A	509	Total cognitive complexity from all methods
Cyclomatic Complexity	N/A	960	Indicates decision-making complexity in code

The results from Static Code Analysis Tools



Quality Gate
Passed

Last analysis 1 minute ago

Security
0 Open issues A

Reliability
1 Open issues B

Maintainability
65 Open issues A

Accepted issues
0
Valid issues that were not fixed

Coverage
65.2%
On 1.6k lines to cover.

Duplications
1.7%
On 8.3k lines.

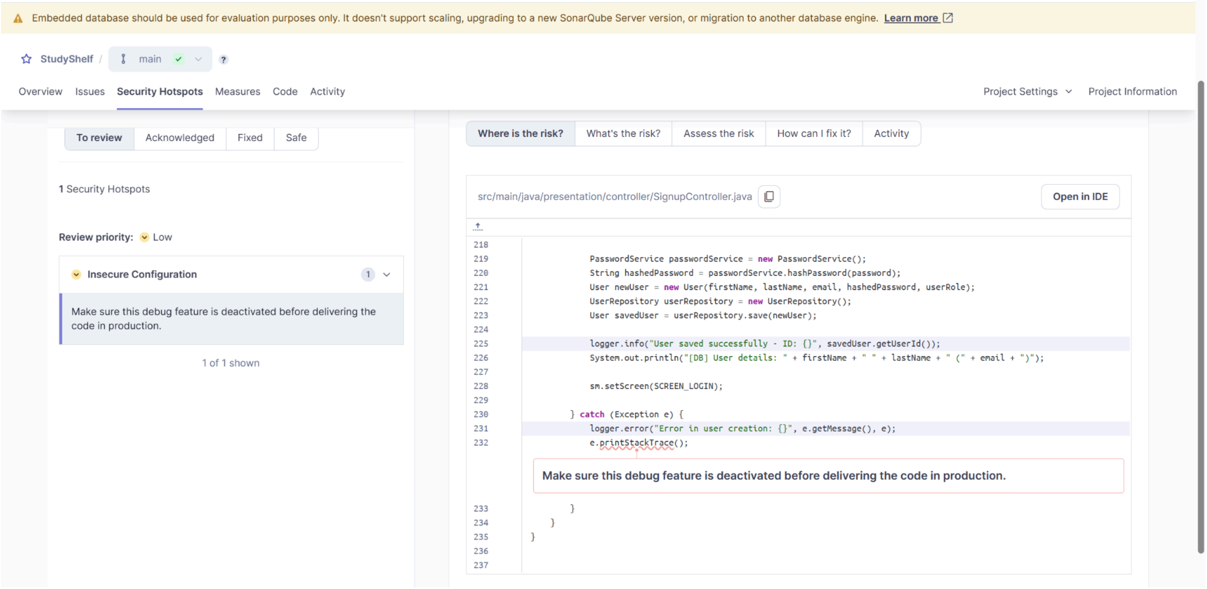
Security Hotspots
0 A

Figure 1: SonarQube Dashboard Overview

3. Issues Identified

Issue 1 Debug Output in Production Code

- **File:** SignupController.java: line 226, 231
- **Issue Type:** Security Hotspot (significant finding)
- **Explanation:** The code contains debug output statements, including *System.out.println()* and *e.printStackTrace()*. These debug output statements can expose sensitive information or clutter application logs in a production environment. Although this issue is marked as low-priority by SonarQube, it is considered significant due to its relevance to production-readiness and secure coding practices.
- **Screenshots:**


Embedded database should be used for evaluation purposes only. It doesn't support scaling, upgrading to a new SonarQube Server version, or migration to another database engine. Learn more

1 Security Hotspots

Review priority: Low

Insecure Configuration 1

Make sure this debug feature is deactivated before delivering the code in production.

1 of 1 shown

src/main/java/presentation/controller/SignupController.java

```
218
219                PasswordService passwordService = new PasswordService();
220                String hashedPassword = passwordService.hashPassword(password);
221                User newUser = new User(firstName, lastName, email, hashedPassword, userRole);
222                UserRepository userRepository = new UserRepository();
223                User savedUser = userRepository.save(newUser);
224
225                logger.info("User saved successfully - ID: {}", savedUser.getUserId());
226                System.out.println("[DB] User details: " + firstName + " " + lastName + " (" + email + ")");
227
228                sm.setScreen(SCREEN_LOGIN);
229
230            } catch (Exception e) {
231                logger.error("Error in user creation: {}", e.getMessage(), e);
232                e.printStackTrace();
```

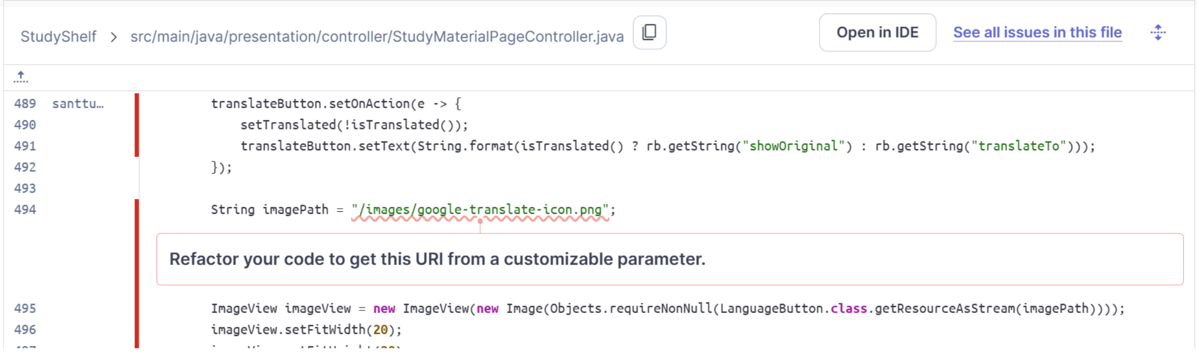
Make sure this debug feature is deactivated before delivering the code in production.

```
233            }
234        }
235    }
236
237
```

Figure 2: SonarQube Issue Screenshot - Debug Output in Production Code

Issue 2 Hard Coded Image URI Path

- **File:** StudyMaterialPageController.java: line 494
- **Issue Type:** Maintainability (significant finding)
- **Explanation:** The code contains a hardcoded URI string ("*/images/google-translate-icon.png*") for accessing image resources, which violates best practices for configurability and maintainability. The issue is considered significant due to its long-term impact on code clarity, reuse, and adherence to object-oriented design principles.
- **Screenshots:**

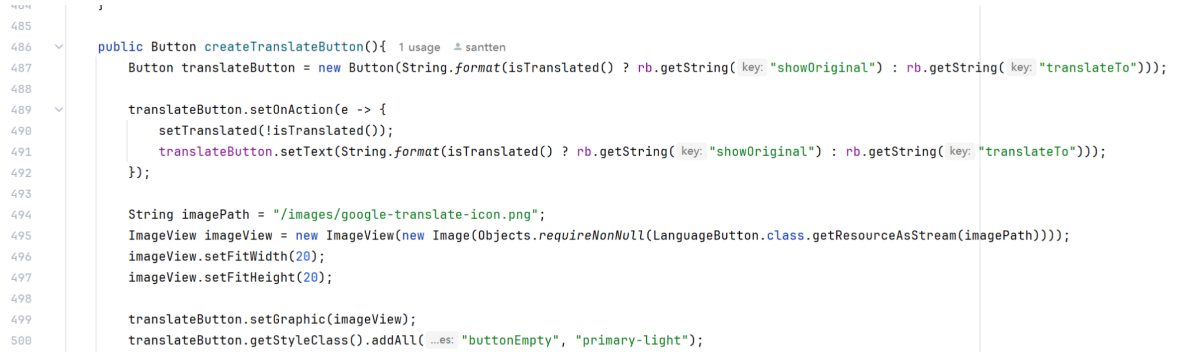

StudyShelf > src/main/java/presentation/controller/StudyMaterialPageController.java
Open in IDE
See all issues in this file

```
489  santtu…      translateButton.setOnAction(e -> {
490                   setTranslated(!isTranslated());
491                   translateButton.setText(String.format(isTranslated() ? rb.getString("showOriginal") : rb.getString("translateTo")));
492               });
493
494               String imagePath = "/images/google-translate-icon.png";
```

Refactor your code to get this URI from a customizable parameter.

```
495               ImageView imageView = new ImageView(new Image(Objects.requireNonNull(LanguageButton.class.getResourceAsStream(imagePath))));
496               imageView.setFitWidth(20);
```

Figure 3: SonarQube Issue Screenshot - Hard Coded Image URI Path



```
485
486      public Button createTranslateButton(){
487          Button translateButton = new Button(String.format(isTranslated() ? rb.getString( key: "showOriginal") : rb.getString( key: "translateTo")));
488
489          translateButton.setOnAction(e -> {
490              setTranslated(!isTranslated());
491              translateButton.setText(String.format(isTranslated() ? rb.getString( key: "showOriginal") : rb.getString( key: "translateTo")));
492          });
493
494          String imagePath = "/images/google-translate-icon.png";
495          ImageView imageView = new ImageView(new Image(Objects.requireNonNull(LanguageButton.class.getResourceAsStream(imagePath))));
496          imageView.setFitWidth(20);
497          imageView.setFitHeight(20);
498
499          translateButton.setGraphic(imageView);
500          translateButton.getStyleClass().addAll( ...es: "buttonEmpty", "primary-light");
```

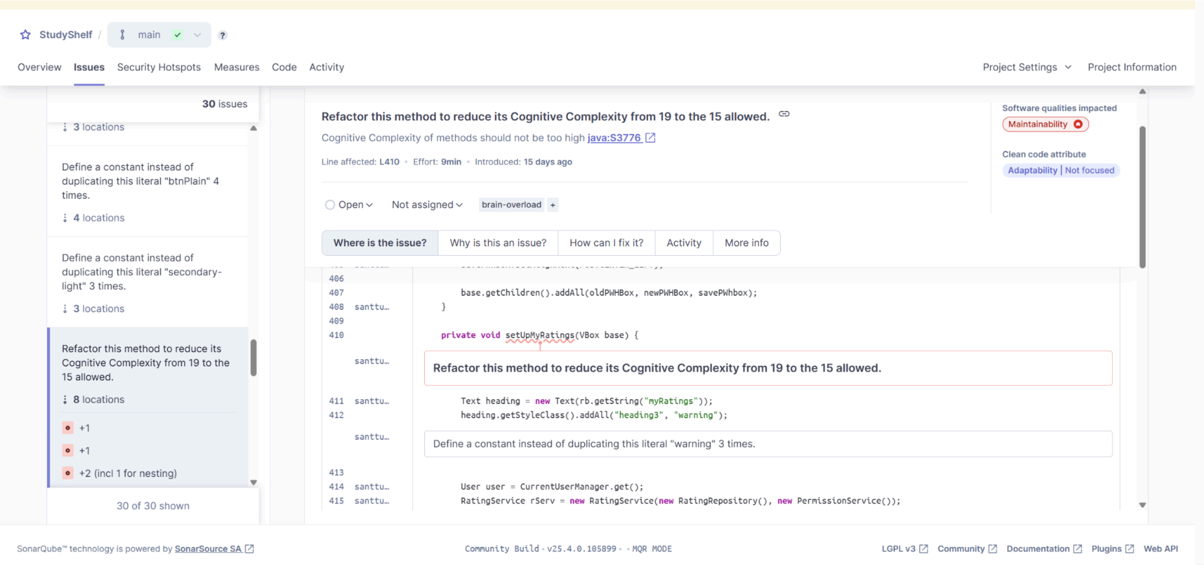
Figure 4: Code Screenshot - Hard Coded Image URI Path

Issue 3 High Cognitive Complexity

- **Files:**
    - MyProfileController.java: line 410 (Complexity: 19)
    - DatabaseInitializer.java: line 30 (Complexity: 18)
    - StudyMaterialService.java: line 36 (Complexity: 16)
- **Issue Type:** Maintainability (significant finding)
- **Explanation:**SonarQube flagged three methods that exceed the recommended cognitive complexity threshold of 15. These methods are too deeply nested or contain too much logic, making them difficult to read, understand, test, and maintain. Left unaddressed, such methods can increase the likelihood of bugs and reduce overall code quality. Although these issues are not runtime errors, they are

considered significant due to their strong impact on long-term maintainability and development efficiency.

- **Screenshots:**



⋮ 3 locations

Define a constant instead of duplicating this literal "btnPlain" 4 times.

⋮ 4 locations

Define a constant instead of duplicating this literal "secondary-light" 3 times.

⋮ 3 locations

Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.

⋮ 8 locations

- +1
- +1
- +2 (incl 1 for nesting)

Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.

Cognitive Complexity of methods should not be too high java:S3776

Line affected: L410 • Effort: 9min • Introduced: 15 days ago

Software qualities impacted: Maintainability

Clean code attribute: Adaptability | Not focused

```
        base.getChildren().addAll(oldPWHBox, newPWHBox, savePWhbox);
    }

    private void setUpMyRatings(VBox base) {
```

Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.

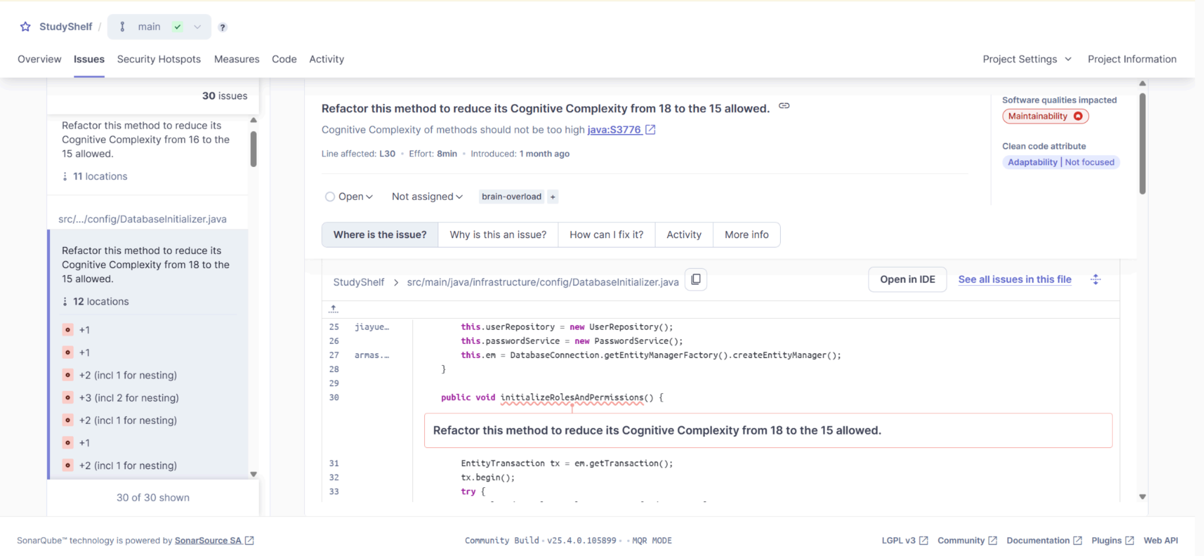
```
        Text heading = new Text(rb.getString("myRatings"));
        heading.getStyleClass().addAll("heading3", "warning");
```

Define a constant instead of duplicating this literal "warning" 3 times.

```

        User user = CurrentUserManager.get();
        RatingService rServ = new RatingService(new RatingRepository(), new PermissionService());
```

SonarQube™ technology is powered by SonarSource SA   Community Build • v25.4.0.105899 • • MQR MODE   LGPL v3 Community Documentation Plugins Web API

Figure 5: SonarQube Issue Screenshot - High Cognitive Complexity



Refactor this method to reduce its Cognitive Complexity from 16 to the 15 allowed.

⋮ 11 locations

src/.../config/DatabaseInitializer.java

Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

⋮ 12 locations

- +1
- +1
- +2 (incl 1 for nesting)
- +3 (incl 2 for nesting)
- +2 (incl 1 for nesting)
- +1
- +2 (incl 1 for nesting)

Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

Cognitive Complexity of methods should not be too high java:S3776

Line affected: L30 • Effort: 8min • Introduced: 1 month ago

Software qualities impacted: Maintainability

Clean code attribute: Adaptability | Not focused

StudyShelf > src/main/java/infrastructure/config/DatabaseInitializer.java

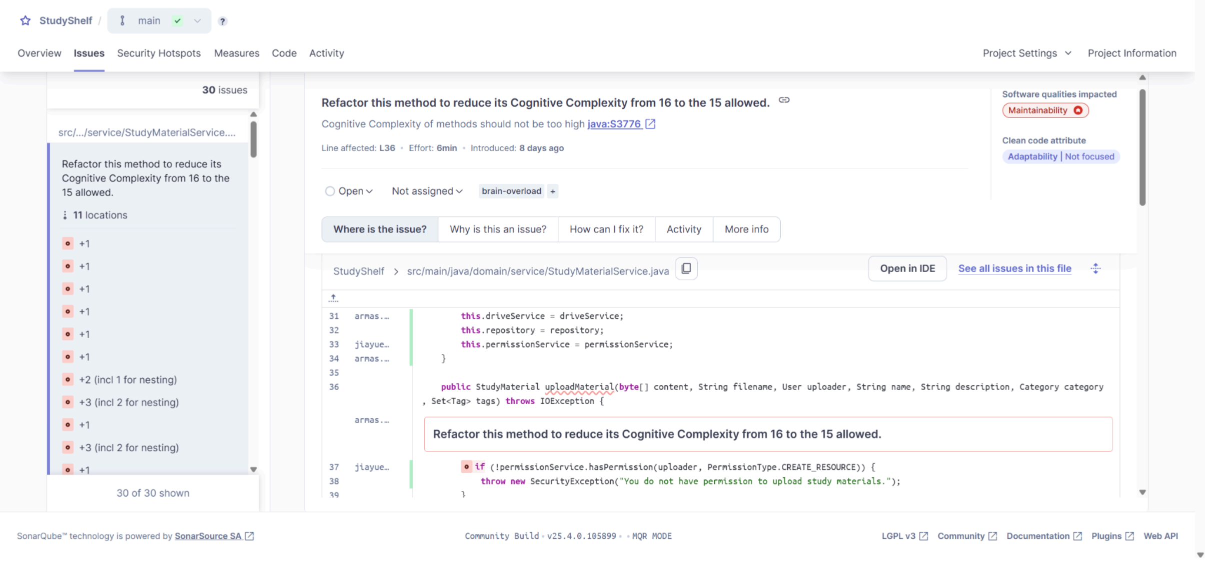
```
        this.userRepository = new UserRepository();
        this.passwordService = new PasswordService();
        this.em = DatabaseConnection.getEntityManagerFactory().createEntityManager();
    }

    public void initializeRolesAndPermissions() {
```

Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed.

```
        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {
```

SonarQube™ technology is powered by SonarSource SA   Community Build • v25.4.0.105899 • • MQR MODE   LGPL v3 Community Documentation Plugins Web API

Figure 6: SonarQube Issue Screenshot - High Cognitive Complexity



Refactor this method to reduce its Cognitive Complexity from 16 to the 15 allowed.

Cognitive Complexity of methods should not be too high java:S3776

Line affected: L36 · Effort: 6min · Introduced: 8 days ago

```
        this.driveService = driveService;
        this.repository = repository;
        this.permissionService = permissionService;
    }

    public StudyMaterial uploadMaterial(byte[] content, String filename, User uploader, String name, String description, Category category
, Set<Tag> tags) throws IOException {
```

Refactor this method to reduce its Cognitive Complexity from 16 to the 15 allowed.

```
        if (!permissionService.hasPermission(uploader, PermissionType.CREATE_RESOURCE)) {
            throw new SecurityException("You do not have permission to upload study materials.");
        }
```

Figure 7: SonarQube Issue Screenshot - High Cognitive Complexity

4. Recommended Changes

Issue 1 Debug Output in Production Code

- **Recommendation:** Remove or replace with structured logging (e.g., *logger.debug()* and *logger.error()*).

Issue 2 Hard Coded Image URI Path

- **Recommendation:** Extract the image URI from the hardcoded string and define it in a static configuration class (e.g., AppConfig) to improve code flexibility and reduce long-term maintenance effort.

Issue 3 High Cognitive Complexity

- **Recommendation:** Refactored three methods with cognitive complexity greater than 15 by breaking them into smaller, logically grouped helper methods. This reduces complexity, improves readability, testability, long-term maintainability, and aligns with clean code principles.

Code Clean-Up Actions Summary

Action Area	**Description**	**Status**
Refactored variables according to SonarQube recommendations	Changed some class variables to be local to the methods according to recommendations	Done
Variable and method names follow Java conventions	Changed variable and method names to follow the Java conventions (camel case)	Done
Class StyleClasses instead of hardcoded strings	Replaced all *getStyleClass().add()* calls using hardcoded strings with a StyleClasses-class that holds them	Done
Refactored SVGContents utility class	Instead of one line String methods, this utility class holds the needed strings as public class variables	Done
Refactored getInstance() of a few Singleton classes to be multi-thread safe	SceneManager and LanguageManager use holders now to ensure thread safety	Done
Removed debug output statements	Removed *System.out.println()* and *e.printStackTrace()* from SignupController.java to avoid exposing debug information in production	Done
Replaced with structured logging	Replaced debug outputs with *logger.debug()* and *logger.error()* using proper log formatting	Done
Extracted hard coded image URI	Moved "/images/google-translate-icon.png" into *AppConfig* static class to improve configurability and reduce maintenance cost	Done
Refactored high-complexity methods	Split methods with cognitive complexity > 15 into smaller helper functions	Done
Improved maintainability	Reduced nesting and logic per method for clarity and testability	Done

5. Conclusion

The static code analysis performed using SonarQube revealed that the codebase for the StudyShelf project is generally well-structured, with good adherence to security and maintainability standards. Overall, the code is clean and maintainable, and most issues identified have been addressed effectively.