UNIVERSITÉ Concordia UNIVERSITY

SOEN 6431 - SOFTWARE COMPREHENSION AND MAINTENANCE

Deliverable 1

DÉJÀ VU

Supervisor
Prof. Pankaj Kamthan

Team D
Authors
Saeed Jamalifashi- 40093292
Darshakkumar Kachchhi - 40206619
Siddhartha Kattoju - 29209905
Navdeep Kaur - 40237921

GitHub : `https://github.com/darshak-k/SOEN6431-D-SCM`

Contents

1 Introduction

The Software Comprehension and Maintenance Project aims to address the challenges of reengineering and improving an existing software system. The project involves a team of individuals who will explore and select a candidate system, for reengineering. The specific domain of it can vary, ranging from games to data science programs, graphics programs, information systems, machine learning programs, parsers, scientific computing programs, or system utilities. The critical aspect is that the problem and solution domains of the system should be understandable and communicable to all team members.

The primary objective of this project is to comprehensively understand the chosen system and identify areas where improvements can be made. This involves analyzing the codebase, understanding the system's architecture, documenting its functionality, and comprehending its underlying algorithms and data structures. The team will collaborate to gather knowledge about the system and establish a shared understanding of its inner workings.

2 Project Brainstorming

2.1 Criteria

Complex Architecture :
A less complex application setup is generally preferred as it reduces the time spent on project setup, allowing more time to be dedicated to analyzing the project and meeting the metrics of the candidate system.

Technical debt :
During the evaluation of the current codebase, it is crucial to identify sections that exhibit high levels of technical debt. Technical debt refers to code that is poorly designed and requires maintenance or refactoring. It is recommended to prioritize projects that address technical debt, as doing so will lead to improvements in code quality, maintainability, and performance.

Resource availability :
After evaluating the availability of skills, resources, and expertise necessary for the project, we have considered the presence of developers, testers, and other essential team members.

Cost-benefit analysis:
In our assessment of the project, we thoroughly examined the costs associated with resources and time. While we did not specifically consider the budget for this project, it can be a factor to take into account during the cost-benefit analysis phase. Additionally, we carefully evaluated the potential benefits of each project, including enhancements in efficiency, user satisfaction, and competitive advantage.

3 Candidate System

Library Management System Saeed JamaliFashi - 40093292

3.1 Purpose

The purpose of a library management system in Java is to provide an efficient and organized platform for managing and automating various library operations and processes. It enables librarians to handle tasks such as cataloging, circulation, patron management, inventory management, and reporting in a streamlined manner. Additionally, a library management system in Java facilitates tasks like searching and retrieving books, tracking borrowing and return transactions, generating reports, and ensuring smooth communication between library staff and users. Ultimately, the goal is to enhance the overall efficiency, accuracy, and accessibility of library services.

3.2 Project Description

This project implements the library management system, including book and author management, user, etc. The system uses the Spring Security module for user authentication and authorization provided with a login system. The back end of this project uses Spring Boot and Spring Data JPA to provide a web application with functionality for managing books and their categories, authors, publishers, and users.

3.3 System Source Code

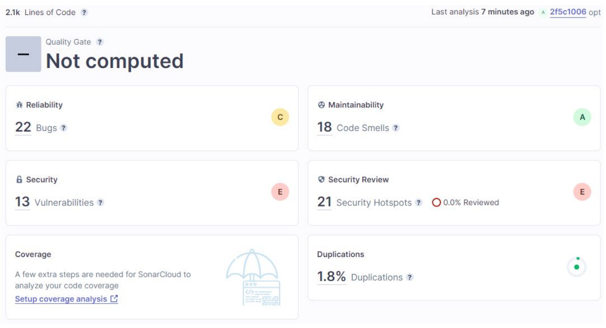
https://github.com/knowledgefactory4u/librarymanagementsystem

3.4 System Stack :

Java, HTML, Spring Boot framework, Spring security.

3.5 Analysis

After conducting a thorough analysis of the code base, we have identified **18 significant code smells** that are negatively impacting the system's performance, maintainability, and scalability. We will employ appropriate refactoring techniques to address the identified code smells. This includes breaking down long methods into smaller, more manageable units, eliminating code duplication by extracting reusable functions or classes, simplifying complex conditionals, introducing design patterns to improve system architecture, and improving naming conventions for better code readability. This proposal outlines our plan to address these code smells and security issues to improve the overall quality of the system.



2.1k Lines of Code ? Last analysis 7 minutes ago 2f5c1006 opt

Quality Gate ?
Not computed

Reliability
22 Bugs ? C

Maintainability
18 Code Smells ? A

Security
13 Vulnerabilities ? E

Security Review
21 Security Hotspots ? 0.0% Reviewed E

Coverage
A few extra steps are needed for SonarCloud to analyze your code coverage
Setup coverage analysis

Duplications
1.8% Duplications ?

Figure 3.1: Preliminary Analysis of Library Management System code base

Analysis Link: Sonar Cloud Dashboard - Library Management System

3.6 Rationale for selection :

3.6.1 Technical Reason

Programming Proficiency:
After considering the skill set of each team member, it was concluded that Java would be the most preferred programming language for the project. This determination was based on the fact that all team members have significant experience using Java as their primary development language.

Technical problems and solutions:
The technical problems and corresponding solutions of this candidate system such as test issue, is understandable to all the group members.

Modularity:
The modularity of source code can be improved and dependencies can be reduced across functionalities. In that sense, our team will be involved in the process of separating functionalities of the candidate system and creating more independent and interchangeable pieces of modules.

Programming Style:
The developer's writing method determines the ease of updating and understanding software.

Documentation:
Inadequate documentation leads to confusion and inefficiency in project implementation, hindering progress and causing delays. Poorly documented processes and procedures increase the likelihood of errors and mistakes, impacting the quality and reliability of the final deliverable.

3.6.2 NonTechnical Reason

Learning purpose: This candidate system covers full-stack development technologies from the front-end and back-end, which is a good candidate for the purpose of understanding maintenance.

Software familiarity:
The candidate system possesses the necessary software, IDE, and system components, enabling us to effectively conduct code re-engineering. Additionally, the entire team is well-informed about the system requirements and is familiar with the required software, ensuring a smooth reengineering process.

Incorporation of emerging technologies:
Implementing emerging technologies and features can improve the functionality, user experience, efficiency, and robustness of the system. Machine learning and AI can be utilized to implement recommendation systems that suggest books based on user preferences. Integrating cloud services into the project can help improving scalability, flexibility, and enhanced functionality

Dependency of External environment/Software:
The independence of our library management system from the external environment gives us the flexibility to make internal modifications to the code. This advantageous feature of the system greatly supports our decision in selecting it, as it enables us to easily customize and adapt the software to meet our specific needs without relying on external factors.

4 Other System Descriptions

4.1 Stationary Shop Management

Darshakkumar Kachchhi - 40206619

Project Description:
Stationary Shop Management aims to streamline the record-keeping process for a stationery shop, which involves tracking the buying and selling of products. The shopkeeper regularly purchases stock from specific vendors and sells them to customers. It becomes challenging to keep track of the current number of products in the shop, anticipate which products may run out of stock in the next one or two days, monitor the quantity of damaged or broken items, calculate the total sales revenue for the day, and determine the amount of money paid to each vendor for stock purchases. Using this software, the shopkeeper can eliminate the inconvenience of relying on pen and paper, while also saving time. The software includes features such as viewing reports, checking stock levels, listing products, purchasing products from vendors, tracking customer sales, and monitoring damaged items.

System Source Code :
https://github.com/theanasuddin/Stationary-Shop-Management

System Stack :
Java [JDK 8+], Java Swing

Rationale for rejection :

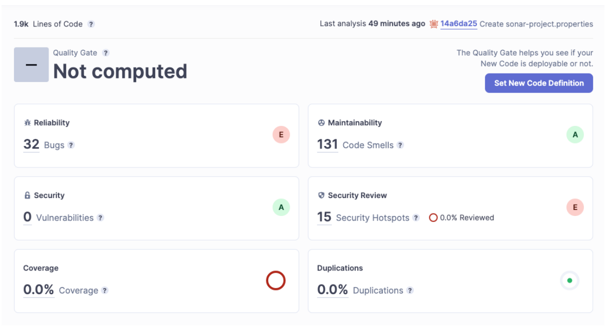
- Many features were not implemented properly.
- Performance issue could have been resolved with a more efficient approach.
- Interface lacked user-friendliness and the ability to perform multiple conversions simultaneously.
- Issues were primarily related to software design rather than the code itself.

4.2 EGOS-2000: A minimal RISC-V Operating System

Siddhartha Kattoju - 29029905

Project Description:
EGOS 2000 is a minimal operating system that implements components for managing resources on a computer. It runs on RISC-V hardware as well as the QEMU software emulator. The system was developed with education in mind. With only 2000 lines of code, egos-2000 implements every component of a functional operating system. The project's vision is to help every college student read all the code of an operating system. It is currently used as a teaching OS for CS5411/4411 at Cornell University.

Preliminary Analysis:
Preliminary analysis using SonarQube[3] showed that the code base is fairly well maintained with less that 5 percent overall technical debt. Despite this, the code base has significant re-engineering opportunities. SonarQube's sonar scanner tool identified 32 potential bugs, 131 potential code smells and 15 potential security hot spots when run on a fedora VM with the project recommended compiler tool chain. An interesting point to note was that there were no unit tests in the code base. This may be have been intentional as the code base was meant to be limited to 2000 lines of code with the goal of being used as a learning tool rather than in a production setting.



1.9k Lines of Code ?   Last analysis 49 minutes ago 14a6da25 Create sonar-project.properties

Quality Gate ?
Not computed

The Quality Gate helps you see if your New Code is deployable or not.

Reliability — 32 Bugs ? — E
Maintainability — 131 Code Smells ? — A
Security — 0 Vulnerabilities ? — A
Security Review — 15 Security Hotspots ? 0.0% Reviewed — E
Coverage — 0.0% Coverage ?
Duplications — 0.0% Duplications ?

Figure 4.1: Preliminary Analysis of EGOS-2000 code base

Analysis Link: Sonar Cloud Dashboard - A minimal RISC-V Operating System

System Source Code:
https://github.com/yhzhang0128/egos-2000

System Stack:
C/C++, Qemu / RISC-V system

Rationale for rejection:

- Team members being unfamiliar with C/C++
- Lack of unit tests - meaning that changes would have to be tested manually
- Requirement to run on Qemu or RISC-V hardware which the team was not familiar with

4.3 Online School Management System

Navdeep Kaur - 40237921

Project Description :
The school management system implements the backend service for an online school management service. This project caters to multiple levels of users, including administrators, teachers, and students. It offers a login system with registration functionality. Students can access features like the notice board, exam results, and attendance records. Teachers can submit exam results, daily attendance, and perform other related tasks. Administrators have privileges to add new students and teachers, create new academic sessions and classes, and perform administrative functions.

System Source Code :
https://github.com/galibBd/OnlineSchoolManagementSystem

System Stack :
Spring MVC, Spring Data REST.

Analysis :

- Code Duplication Reduction: We have noticed the presence of similar methods in the controller, each handling different endpoints but with repetitive logic. To enhance code maintainability and readability, it is recommended to extract common functionality into reusable methods. Alternatively, utilizing inheritance or composition techniques can also help eliminate code duplication.
- Business Logic Extraction: Currently, the business logic resides within the controller. To achieve a better separation of concerns and enhance code modularity, it is advisable to extract the business logic from the controller and move it to separate service classes. This approach will promote better organization and maintainability of the codebase.
- Exception Handling Implementation: The existing code lacks proper exception handling mechanisms. To ensure the application gracefully handles errors, logs exceptions, and provides meaningful error messages to users, it is crucial to implement robust exception handling. This improvement will enhance the reliability and user experience of the application.

By addressing these areas during the reengineering process, we can significantly enhance the code quality, maintainability, and scalability of the project.

5 References

1. ISO/IEC 25010:2011," 03 2011. .

2. ISO/IEC 9126-1:2001," 06 2001..

3. Technical Documentation of Solar Cloud.

4. Design Pattern by Refactoring Guru

5. What is Software Maintenance and Why is it Important for an Organization? By Maitray Gadhavi