SOEN 6431
Software Comprehension and Maintanence

Varun Pandey
Preet Angad Singh Nanda
Mir Prasad
Mahavir Patel

Deliverable 1
Reengineering Oppurtunity

GitHub Link

Contents

1 Abstract

In today's dynamic market, software systems must undergo constant evolution to remain competitive and successful. This process, known as software maintenance, involves continuous development and updates to align the system with evolving market demands. Successful maintenance requires careful analysis of source code and documentation, along with a comprehensive understanding of the system. This abstract explores the importance of software maintenance and highlights the need for positive changes in software systems to keep them up-to-date with current trends and demands. By embracing evolutionary software maintenance, businesses can ensure their software remains relevant, robust, and adaptable in an ever-changing technological landscape.

In the face of rapidly advancing technology and evolving customer expectations, software maintenance plays a vital role in the long-term success of software systems. It encompasses various activities such as bug fixing, performance optimization, feature enhancements, and security updates. Through continuous analysis of the source code and documentation, software maintenance teams identify areas for improvement and implement positive changes to address emerging challenges. This iterative process enables the software system to adapt to market demands, exploit new opportunities, and deliver enhanced user experiences. By embracing a proactive approach to software maintenance, organizations can not only stay competitive but also foster customer satisfaction, loyalty, and trust in their products or services.

2 Introduction

Software Maintenance refers to the set of activities involved in managing, modifying, and improving a software system after its initial development and deployment. It encompasses various tasks such as bug fixing, performance optimization, feature enhancements, and ensuring compatibility with new hardware or software platforms. The goal of software maintenance is to ensure the continued functionality, reliability, and efficiency of the software system throughout its life cycle.To carry out effective maintenance, it is crucial to thoroughly analyze the source code and documentation of the software system. This analysis helps identify areas that require improvement or modification. By understanding the system's inner workings, dependencies, and design principles, maintainers can make informed decisions about implementing positive changes. Additionally, having a comprehensive understanding of the software system allows maintainers to anticipate potential issues and proactively address them during the maintenance process.

In summary, software maintenance plays a vital role in the long-term success of a software system. It involves continuously refining the software to adapt to changing market demands, improve its performance, and ensure its overall quality. This requires careful analysis of the source code and documentation, as well as a deep understanding of the system's intricacies. By effectively carrying out software maintenance, organizations can keep their software systems competitive, reliable, and aligned with the needs of their users.

3 Responsibility Matrix

Name/ Problem	Research	Ideation	Code Quality Checks	Discussion	Documentation
Preet Angad Singh Nanda	✓	✓	✓	✓	✓
Varun Pandey	✓	✓	✓	✓	✓
Mir Pankaj Pasad	✓	✓	✓	✓	✓
Mahavir Patel	✓	✓	✓	✓	✓

4 Candidate Selection Criteria

While finalizing our candidate system we decided to focus on the fulfilment of certain criteria so that the process of refactoring could be improved. Those criteria are

4.1 Code Quality

Low code quality means that it would take more time to understand and refactor the system, so selecting a candidate with good code was vital. Since code quality is a broad term, we specifically observed the following.

1. Proper Nomenclature
2. Readability and indentation
3. Modularity

4.2 File Structure

Having is a proper file structure is important, its doesn't effect the compilation of the code but it drastically improves code visualization and modular decoupling. It promotes the programmer to write independent units of code which is helpful in debugging and making changes in the future.

4.3 Documentation

Having a proper documentation if probably the most important criteria in this list, it enhances the clarity on the workings of a system and aids developers in understanding and navigating the source code. Its also helps in collaboration, code maintenance, making updates and bug fixes.

4.4 Language Comfort

After considering the expertise and comfort level of each individual teammates we narrowed down our selected language to Java.

4.5 Dependencies

Software dependencies are external frameworks, libraries, and other parts that a piece of software needs to perform properly. These dependencies can either be installed separately or directly within the software. Software dependencies include things like user interface frameworks, database management tools, and libraries for processing data. It is crucial to keep dependencies up to date for enhancing the safety, functionality, and compatibility of software.

4.6 Compatibility

The ability of software and hardware from various sources to function together without requiring modification is referred to as software compatibility. The program must function flawlessly across all OS architectures in today's wide range of platforms, such as Windows (x86 or x64 architecture), Mac OS, and Linux.

4.7 Accessibility

Making ensuring that application software is accessible to and used by the broadest audience is the aim of accessibility. All users must be able to see, comprehend, and utilize the controls for this to be possible. To do this, the design must take into consideration the reality that each user's capacity to see, hear, input data, read text, and process information differs over time and in different contexts. To do these tasks, some users need specialized assistive input and output devices or certain display features. When creating the software application, this must be considered.

5 System Table

Candidate System	Source Code	Proposed by	Status
Employee Payroll System	Click Here	Preet Angad Singh Nanda	Selected
Online Banking System	Click Here	Varun Pandey	Not Selected
ExamShow - Online Exam System	Click Here	Mir Pasad	Not Selected
Library Management System	Click Here	Mahavir Patel	Not Selected

6 System Description

6.1 Employee Payroll System

The Employee Payroll System for Multiple Departments is a comprehensive software application designed to manage and automate the payroll process for organizations with multiple departments. It provides a centralized platform to handle the payroll-related activities of various departments, ensuring accurate compensation calculations, efficient payroll processing, and streamlined reporting for each department.

The naming convention employed for functions and variables was intuitive, making it easier to discern the purpose of each object. Although comments were absent in the code snippets, the precise naming convention contributed to a clear understanding of the code flow, surpassing the clarity offered by other projects mentioned. Notably, the project consisted of an optimal 1.6k lines of code (LOC), aligning with the deliverable's instruction of 1-2k LOC.

Tech Stack : Java, Java Swing, AWT (Abstract Window Toolkit), SQL (Structured Query Language)

6.2 Online Banking System

This project is for maintaining records of the employee and their salary. The Banking system is divided into two components: the User-Front and the Admin-Portal. The User-Front is designed for users and includes modules such as User Signup/Login, Account Management, Transfer, Appointment Scheduling, Transaction History, and User Profile. On the other hand, the Admin-Portal is primarily used by administrators and encompasses modules for managing User Accounts and Appointments.

The reason we didn't select this project is because while going through the git repository I couldn't find any reasonable documentation, so to understand the project I had to go through the entire code, which was not properly refactored, the indentation was not properly done, the naming conventions of the variable and functions was not properly done. I tried running the application but since the documentation was missing, I was not able to run it. Another major issue is that a couple of libraries used for the project are deprecated. The author of the program have added many redundant comments to the places where comments are not necessary and at some

places the comments are completely not there, so this inconsistency of the comments on the source code is adding another layer of difficulty in understanding the source code.

So if we add up all these issue I came up with the conclusion that refactoring this project is going to take a long time because the code quality is not up to par.

Tech Stack : Spring Boot, Spring Security, Thymeleaf, Spring Data JPA, Spring Data REST, JavaScript, JQuery

6.3 Online Examination System

This project is about the online exam system. The online exam system is used by educational institutions to provide exams and online quizzes for students in a variety of subject areas, including sports, programming, science, and other topics. It is also used for quizzes during college Tech-Fest events. It has various features for both students and faculty. Student can View Subjects, Practice MCQs, student can register and appear for Quiz, View recent Events, student can View Results, Notifications. Faculty can make/delete papers, faculty can view/update papers, faculty can view results,append questions,send Notifications.

There are multiple reasons for not selecting this project, the first one being the lack of documentation about the features explanation. Due to missing documentation I was able to understand all the features of student and faculty. The second is that these system is basically designed to be used by educational institution because nowadays these type of system are also required by corporate sectors for job interview exams. Other reason is that it has a very basic UI just simple which i didn't find attractive to use. Also the biggest problem is that it does not have proctoring feature. Nowadays anyone can access internet and get answers to anything so it should have a full screen lock mode feature and also continuous camera proctoring. So all these issues conclude me that this project needs a lot of refactoring and it would take a lot of time to do that.

Tech Stack :

- BackEnd : J2SE,J2EE,JSTL,JUNIT
- FrontEnd : Twitter Bootstrap,Ajax,jQuery,JSON,CSS,HTML
- Database : Mysql

6.4 Library Management System

This project illustrates the Java OOP idea for a small library management system. It is a Java Swing-based GUI application that stores its data in a SQL database using MsSQL. The three different user categories are librarian, clerk, and borrower. They can carry out certain functionalities or processes, as determined by the user. For instance, librarians are capable of overseeing the whole library system. The clerk performs a variety of tasks to ensure that the system runs well. One of these tasks is overseeing the library's collection of books, from which patrons can borrow or rent them.

This proposal was not chosen because it lacked several critical non-technical and technical components. The project doesn't have a good file structure. The programmer must examine each file to make any future changes to the project. It is challenging to trace back the system's execution cycle due to a lack of adequate documentation and code comments. This project is dependent on a few things, namely the sqljdc driver that connects the system's database. It is challenging to make the project compatible with other developing environments since it is IDE-centric on NetBeans. There are generally not enough user-centric error messages in the user interface and all of the error messages and warnings can only be comprehended by technical people. The absence of documentation and a decent file format is the primary deterrent to choosing this project.

Tech Stack : JAVA, MsSQL, Java Swing

7 Candidate System Description

For a business, a reliable payroll system portal is essential for a number of reasons. First off, it makes sure that compensation computations and payments are correct and made on time, reducing mistakes and delays that can cause employee unhappiness. Second, by automating payroll procedures, it reduces human labour and boosts productivity, streamlining administrative responsibilities. Thirdly, it guarantees compliance with labour and tax rules, lowering legal risks and fines. A trustworthy payroll system site also offers protected access to employee data, guaranteeing data protection and confidentiality. Last but not least, it improves overall happiness and employee involvement by offering self-service alternatives for obtaining pay stubs, tax forms, and other pertinent information.

A strong reporting capability provided by a decent payroll system site also enables management to learn more about labour expenses, spending, and staff productivity. These reports assist the business make well-informed decisions, manage resources wisely, and pinpoint opportunities for development. A productive payroll system portal also interfaces with other HR and accounting systems, providing smooth data sharing and obviating the need for manual data entry or reconciliation.

8 Candidate System Rational

The reason for choosing the Employee Payroll System is that after comparing the pros and cons of each system, we decided that this system is the suitable for making further updates and refactoring. This system satisfies more requirements than any other system that we have discussed above. This source code of the system was greater than 1000 lines of code but wasn't too big that it would create an issue. Using sonar, we were able to detect 167 issues with the code, we will go over those issue more in details in the next deliverable. Also, this project was entirely programmed in Java which is an added bonus since all of our teammates are comfortable in that language.

9 References

1. PANKAJ KAMTHAN (2023) "Understanding Maintainability"
2. PANKAJ KAMTHAN (2023) "The 'Laws' Of Software Evolution"
3. Why Maintaining Software Dependencies is Important and How to Do it Effectively
   https://www.codiga.io/blog/maintain-software-dependencies/
4. What is compatibility in IT?
   https://www.techtarget.com/whatis/definition/compatibility
5. About Application Software Accessibility
   https://universaldesign.ie/technology-ict/.../application-software/about-application-software-accessibility/