61FIT3SQA

Software Quality Assurance

Spring 2021

Banking Management System

Code Inspection Report

1. Introduction

1.1 Purpose

This report, is aimed to provide a detailed description about the software design, especially in the Banking management system

1.2 References

This report is built on the "Code Inspection Report" of Software Design and Development from the Department of Computer Science and Electrical Engineering of UMBC.

1.3 Coding Conventions

Given the high demand for face-to-face currency exchanges, we developed a scaled-down version of a system that was considered essential when customers exchanged money to each other. Many of the forms used in the framework are true to fact. However, due to financial and logistical constraints, this research was only carried out in a theoretical university environment.

1.4 Defect Checklist

In this project, we create a standard defect checklist for the system as well as the advanced for evaluating the system

- ❏ Code does not contain inline JavaScript event listeners
- ❏ Code does not contain inline style attributes
- ❏ Code does not contain deprecated elements & attributes
- ❏ Page begins with a valid DTD (HTML5 doctype)
- ❏ Code is indented using hard tabs

- ❑ Tags and attributes are lowercase
- ❑ Tags are closed and nested properly
- ❑ Markup is semantic (e.g. class names do not denote presentation, Items in list form are housed in a UL, OL, or DL)
- ❑ Tables are only used to display tabular data
- ❑ Element IDs are unique
- ❑ Code validates against the W3C validator
- ❑ DOM nesting depth does not exceed 12 levels
- ❑ Total page weight does not exceed client requirements (e.g. 1000kb)
- ❑ TItle case is used for headers/titles and forced to all caps using the CSS declaration text-transform: uppercase;
- ❑ Where text is included via images, CSS image replacement is used.

2. Code Inspection Process

2.1 Description

In this section, we briefly describe the code inspection process. First, the Banking Management System ideally starts with the “Informal” process, “Walkthrough” process, “Technical” process and then “Inspection process”. But, in a practical environment, we focus on the “Informal” process since this is just a university-based project.

2.2 Impressions of Process

The process we are using in our project is just effective enough for small-space development. Furthermore, the Banking management system project we provide some basic features and for that, in self-adjust, we find out the project’s quality is medium. However, this system can also be optimized for some advanced features in further development.

2.3 Inspection Meetings

The project is only applicable in a university environment so the meetings is quite basic like: offline meetings. Furthermore, we create a repository on Github for system version control and Google Meetings for some initial discussions.

The offline meetings: Hanoi University Library

- Start time: 4.30 PM
- End time: 5.30 PM
- Attendance: All members
- Purpose: Coding front-end and create database

The online meetings: Google Meetings

- Start time: 9.00 PM
- End time: 10.30 PM
- Attendance: All members
- Purpose: Coding back-end and optimize front-end

3. Modules Inspected

Our project is based on MVC design patterns, it is basic design patterns in software development. Looking at the details, not all the whole project is about MVC design. Some of our packages in project is a grouping of functions: accountController.js, transationController.js, authentication.js

In discussing the quality and quantity of comments and adherence of “coding convention”, we focus on the quantity the most.However, all the coding convention comments are provided but we will publicize it in the future for the project security.

4. Defects

4.1 The login defect

In the package “Controller” especially in “authentication.js” we have not optimized the login response time. After calling the method “/login”, it is still delayed for at least 1000ms and then we can sign in the system.

- Correctness: The method is stable when running in practice.
- Completeness: The method is fulfilled with exactly what the functions are supposed to be.
- User friendliness: The method is not well-displayed organized, when calling it, the error handling still needs to be printed.

When trying to fix this defect, we still not flawlessly debug the function. However, we fix almost everything we can to make it optimized.

4.2 The transaction defect

In the package “Controller” especially in “transactionController.js” we do not have a good way of transferring data from the database. We just make the basic transferring as plus balance the account received money and minus balance the account transferred.

- Correctness: The method is very stable when running in practice
- Completeness: The method is filled with the initial requirements that we declare.
- User friendliness: The method is stable while the customer makes transactions. However the way to edit and update databases is not optimized enough.

The issue of transaction is a hot potato problem, we have not encountered any good solutions for this problem. However, we are trying to make our existing solution optimized as good as possible.