Software Design Plan

A. Business Case

1. Problem Statement

Endothon Finance is facing an error on their customer facing web app. Their small company is focused on providing business loans and they have recently upgraded their systems to a new solution. In this new web app, the potential customer enters their information to be collected, sorted, and sent via **Endothon Finance** to themselves and other loan lenders that they work in parallel to. This application process asks for the applicant companies' last 5 years of financial records, and for companies that are less than 5 years old, all of their previous financial data and then also projects covering a 5 year span. This part of the application process on the web app is having an error that asks for the incorrect years records when prompting for the applicants financial information.

2. Business Requirements

The web app in use must provide the proper years for those applicants who have 5 years of financial information to back up their loan application. The current system is not prompting the applicants for the proper years that are needed to go forward with the loan process.

The web app is also supposed to provide prompting for the applicants to prove the financial information they have for their companies over less than 5 years if their companies haven't been established for the full five years, and then also prompt them for predictions of their financials to fill in the missing amount of years with these projections. The current system, with its year prompting bug that is currently affecting the system, does not perform this task properly.

3. In-Scope Action Items

This design plan only focuses on the software sections that are responsible for the prompting of the yearly financials for both 5+ year old companies and less than 5 year old ones. The front-end sections of code that reveal the years will also be checked to assess if it is a problem to do with the front-end display or just the back-end functions.

4. Out-of-Scope Action Items

Any other aspect of the code that is not directly related to the years for financial information being prompted is not included in the scope of this design plan. No other front-end functions will be checked other than the aforementioned parts that show the prompting of years for loan applicants.

B. Requirements

1. Functional Requirements

The application needs to update the year prompting system for companies that have 5 years of financial history available. The prompt options for companies that are less than 5 years of age must appropriately suggest asking for the most current years of operation that the company does have, as well as a forecast of future financials for the remainder of the unfinished years in the 5 year period.

2. Non-Functional Requirements

The web application has to have a proper uptime and be able to be readily used. Also, the companies back-end processes have to also be up and running and working in tandem to the web application front-end.

C. Software Design

1. Software Behavior

The web based application needs to have entered into it the applicant companies year of establishment which it will then use to prompt for the financial history that the system needs for financial information depending on the company's age. Information that is non-accurate or nonsensical should not be allowed in this input area: such as years before 1700 or past the current year.

The web based application must then, in this next step of retrieving from the customer company information, have inputted into it the financial information for the last 5 years of the company's existence, or what financial information the company has if under 5 years old and then future forecasts. The years of information that the software asks for need to be accurate, and the data needs to be properly stored into company systems to be propagated to other parts of the loan process. These fields that are prompted should only allow for the financial information to be set into them, with documents also provided that support these numbers provided in the application being submitted by potential customer companies.

2. Software Structure

The design approach being proposed will directly approach the issues being presented in the current program by focusing in on the “LoanPrompting” class and its functions for prompting the year and retrieving the data therefrom the user interface. The current functions in the program will be assessed to see what error is causing the current bug, and if the functions are found to be inoperable in their current forms, then a new function will take their place and use computer time tracking components to calculate the year, and then prompt the web app user the proper years based on this new calculation.

The front-end systems will also be assessed if no error is found at the immediate location of yearTime functions within the “LoanPrompting” class to identify which area of code this error is existing in. If, in fact, it is in the front-end part of the current software, then the html will be updated to reflect properly the back-end prompting for the right years of financial information from the user.

D. Development Approach

1. Planned Deliverables

The software will receive a bug-fix that patches the area of code that has been causing the wrong prompting of years to the customers when requesting financial data from their companies. This new/fixed code will be able to be updated into the system to have the software perform as intended.

To accomplish this deliverable, the bug will be assessed through looking into the area of back-end code that is responsible for prompting the correct year to the potential customer using the web app. Next, either a rewriting of this code will need to be done or the front-end may need to be checked for the error of years and the html code fixed in that section of code. Then, the web app form will be tested to ensure accuracy of the fixes done to the system. Finally, the software will be fully updated and this solution will be complete.

As a deliverable as well, there will be documentation citing the specific area of code being altered to fix this bug and the difference between new and old functions will be expressed in this document for future reference if something needs to be realigned or revisited. This will be done through writing to this documentation report during the entire bug-fixing process to make sure each detail is given to this document. Then, at the end of the completion of this design plan, there will be a synapsis of the specific changes done.

2. Sequence of Deliverables

The deliverables will be conducted at the same time, though the bug-fix itself will come shortly before the documentation describing what was done within the system as the final summation will need to be stated on the document.

3. Development Environment

We will enter into the code that is currently running from the company's database in the *AWS* cloud. Here, we will be able to access the running copy of the current software for inspection. We will use *IntelliJ IDEA* for the editing of the specific files and functions necessary for the bug-fix, namely accessing the LoanPrompting.java file and then also the loanAppPromts.html if necessary. We will look at these files to find the root of the current bug, and then program a patch using either java or html for the problem.

4. Development Methodology

We will begin using the Waterfall methodology of development for this bug-fix. This is because the scope has been determined to be static and small, which lets a direct approach to the problems being presented in the current code be suitable for this deployment. This type of methodology helped to inform the development process by outlining the clear problem being faced by users of the program currently, and laid out a direct approach to a viable solution because of the seemingly simplistic nature of the current dilemma.

The RAD development methodology approach was considered but not implemented, as the small scope and direct correlations between easy to navigate documents and functions makes the rushed and iterative nature of RAD unnecessary.