**Software Design Plan**

| **WGU Student ID** | 010771133 |
| --- | --- |

# A. Business Case

## 1. Problem Statement

The problem is that the applicant-facing web app is returning the wrong fiscal years for the applicants. Instead of using the data from the most recent 5 years or forecasted fiscal years, it is returning the first 5 fiscal years from the business. Ticket #D480-AEN1 illustrates the issue in more detail, where the website pulled fiscal data from 2000, 2001, 2002, 2003, and 2004 instead of the most recent 5 years of fiscal data.

## 2. Business requirements

The first additional business requirement is that the web application needs to be able to generate a loan profile for a business that has been active over 5 years with a positive fiscal history using the data on the most recent 5 fiscal years excluding the current year. The current application draws data from the first 5 fiscal years instead of the latest 5 fiscal years of a company, therefore it could deny businesses a loan profile if the business had poor fiscal years during its first 5 years.

The second additional business requirement that is not met is that the application must automatically build an accurate loan profile for businesses. Because the application is taking data from the incorrect fiscal years, the loan profile is also inaccurate, hence the web application fails this requirement of automatically generating an accurate loan profile for its applicants.

The third business requirement is that the web application must be able to generate a loan profile for a business that is less than 5 years old. It should request the available fiscal years from a business and then supplement the data with forecasted fiscal years data if needed to generate a loan profile. The existing web application fails to meet this requirement because it will always use the first 5 fiscal years to generate a loan profile and may be ignoring the forecasted fiscal data. It fails the requirement because the web application fails to automatically identify the specific years of financial data being requested based on the conditions specified in Ticket#D480-AEN1.

## 3. In-scope action items

An in-scope action item is examining and modifying the underlying logic and code for selecting the fiscal years used to generate a loan profile on businesses that have over 5 fiscal years of history. This will allow the application to successfully generate a loan profile for businesses with over 5 fiscal years of history using the correct fiscal years.

Another in-scope action item is to examine or create validation functions to check for business age. This will make sure the application knows how old a business is and how many fiscal years of history they have and will let the application show which fields to show. For example, having a business age validator function will help prevent the application from asking for forecasted fiscal years for businesses that have 5 or more fiscal years of history. Vice versa, it will help prevent the application from not showing the option to insert forecasted fiscal years for businesses that are younger than 5 years.

Another in-scope action item is examining and modifying the underlying logic and code for selecting the fiscal years used to generate a loan profile on businesses that have less than 5 fiscal years of history. This will allow the application to successfully generate a loan profile for businesses with less than 5 fiscal years of history using the correct fiscal years.

## 4. Out-of-scope action items

One out of scope action item would be to optimize the retrieval process for the loan profile creation to be less of a strain on our resources as well as the applicants resources. This is out of scope because it is not needed to complete this ticket but can provide tangible assistance for it. This action item is of low priority and can be completed later down the line.

Another out of scope action item would be to provide a way to automatically calculate the forecasted fiscal years for businesses that do not have a 5 fiscal year history. Applicants can supply data from previous fiscal years and internal calculations and logic can be done using the data to automatically calculate the forecasted fiscal years instead of having the applicants supply the forecast themselves. This would allow us to validate the forecasted fiscal years ourselves and also streamline the process to be less of a burden on applicants. This action is out of scope because it is not necessary for the resolvement of this ticket even if it aligns with the goal of the ticket which is to provide accurate loan profiles to applicants using the correct fiscal years and/or forecasted fiscal years.

# B. Requirements

## 1. Functional requirements

* Rework or fix the data retrieval logic so it correctly returns the last 5 years of financial data for applicants with 5 or more fiscal years of data. As of right now, the application is drawing data from the first 5 fiscal years of a loan applicant's history instead of the most recent 5 fiscal years which are more relevant.
* Make sure and verify that the application gathers both the forecasted fiscal years and the previous fiscal years for a total of 5 years worth of fiscal information for applicants with 5 or less years of fiscal history. Must make sure that any changes to the logic doesn’t impact the application still needing to request and use forecasted fiscal years to generate a loan profile for applicants with less than 5 fiscal years.

## 2. Non-functional requirements

* We need to make sure to guarantee availability and access for relevant parties of all this financial data, aiming for 99.9% uptime with minimum accepted uptime of 99%
* The financial data and all private information must be secure, with various security measures installed and set in place. There should be a way to authorize and authenticate who has/should have access to this information and a way to log any activity involving these private financial data

# C. Software Design

## 1. Software behavior

* A category of input should be a date selector for the applicant to enter the date that their business was created so that the application knows how many years of fiscal history the applicant has and can be used to verify business age as well as select the appropriate fiscal years to use for loan profiling. The data selector should be done via popup calendar for easy and intuitive user experience. The input constraint is that this inputted date should not be in the future of the current date at the time of applying to ensure that the inputted date does not break the logic used to select the appropriate fiscal years for loan profiling.
* Another category of input should be 5 fields used for fiscal history and/or projected fiscal years. These field names should change dynamically depending on the date entered for the creation of the applicant’s business. For example, for businesses that have entered a creation date indicating they are over 5 years old, there should be 5 input fields that show up, 1 for each of their most recent 5 fiscal years. Alternatively, if a business is only 3 years old, there should be 3 fields for 3 most recent fiscal years and then another 2 fields for their next 2 forecasted fiscal years. The constraints for these inputs is that there should only be at any given time, 5 input fields available for the user, no more and no less.

## 2. Software structure

* One necessary function would be something akin to getBusinessDataInput function which would be used in the User Interface section. This function would retrieve the data the user enters for their business (such as creation year) and return values such as integer variable fiscalYears and fiscalYearsForecasted which would then be used to display the appropriate number of fields for input for fiscal years and forecasted fiscal years.
* Another function would be something like caculateLoanProfile(fiscalData[]), which will take all the fiscal data for a company in the form of a list, and select the relevant years of data and calculate/create a loan profile based on that data. It would then be displayed to the user using a function such as displayLoanProfile(company), which print all the company’s loan profile and data and relevant information to the screen for the user

# D. Development Approach

## 1. Planned deliverables

* The first deliverable is the getBusinessDataInput function, which handles gathering the input for business data that the user puts in, parses it, and calculates the age and the necessary forecasted fiscal data fields and previous fiscal data fields. The function needs to accept all the business details as arguments such as applicant name (string), date created (date), business name (string), etc. The function should then correctly assign these to a custom made data structure containing variables for each of these data inputs. It will then take the current date and subtract 5 from the year value and compare it with the inputted business creation date to calculate how old the business is. Using a switch function, it will compare it to the cases and display the relevant fields for fiscal data or forecasted fiscal data. Example is if the case is 4 business year age, it will display 4 fiscal history fields and 1 forecasted fiscal field. If the case is 2 then it will display 2 fiscal history fields and 3 forecasted fiscal fields. If the age is not 1,2,3,4 then it will default to displaying 5 fiscal history fields.
* Another deliverable is the calculateLoanProfile function. It should take data in the form of a list of fiscal data and then use it to form a loan profile. It must first verify that the data is successfully obtained by checking the list using a lengthof(list) function to see if the list is of the length 5, 1 for each fiscal data year. Then it should loop through the list to make sure the fields are all populated, in case an error happens where the list length is correct but some data is missing from the list. After successfully verifying that the correct fiscal data is there in the list, it should calculate a loan profile and then populate a custom data type loanProfile with it, before returning the loanProfile.

## 2. Sequence of deliverables

* First deliverable should be the getBusinessDataInput function. The calculateLoanProfile is not useful on its own and hard to test without the user input data first, hence the getBusinessDataInput function must be completed first. The getBusinessDataInput function is what populated the list of fiscal data that is then used as an argument for the calculateLoanProfile. It also is needed to create the loanProfile data type and populate some of the fields in it, including business name, age, applicant name, etc.
* The second deliverable would be the calculateLoanProfile function. Because it depends on the data gathered from the getBusinessDataInput function, it should be delivered after the getBusinessDataInput function is first delivered. We must first be able to gather and parse the business information and fiscal data before we can create a loan profile for the business.

## 3. Development environment

* Coding Languages: Python will be used on the backend logic and for building the classes and functions we need. Python is a powerful, versatile, and relatively user friendly coding language. Because it has an extensive libraries list and active community, it will be reliably future proof. The frontend will be made using Javascript and HTML.
* Environment: Visual Studio will be the environment we use for this project. It contains built in code editors, plugins, and an all in one environment for deployment and testing as well. Due to its various plugins it is highly versatile and should have no problem working with our chosen coding language and project. The plugins will let us have an easy setup for testing deployment for our application to the web.
* CI/CD Tools and Platform: GitLab will be used as the CI/CD platform, due to all of its features and packages. It can be used to maintain git repositories, has issue ticketing features, CI/CD, and many more. Using GitLab for CI/CD not only fullfills our CI/CD needs but also lets us do many more things without having to use extra software and platforms that will complicate the development process.

## 4. Development Methodology

* The chosen methodology for this ticket is Agile. Agile’s flexibility and concurrent development style will let us have working parts of the fix ready for deployment and testing quickly. This allows us to obtain quick feedback and iterative improvements. Additionally, the scope of this ticket is very small, so the possibility of large scope creep is minor.
* Agile was chosen over Waterfall because Waterfall requires sequential development and a full requirement-gathering phase, which means that there's no room for feedback until after deployment and that bugs will only be discovered late into the development cycle. Agile ensures that testing can happen immediately after a part is developed rather than having to wait for a full system deployment.