Quality Assurance Test Plan

Written By: John McGinnes

A. Overview

1. Software design plan summary

The Endothon Finance website application has an error in the data-gathering logic. In normal operation, the application should retrieve the last five years of financial information for a given business, not including the current fiscal year. Instead, the current iteration is incorrectly gathering data from the first five years after the business's establishment date. If the business does not have enough data to provide for five years in total, the program should retrieve what historical data is available and use a forecasting algorithm to be able to include future years to create a complete five-year profile. This part of the program is currently not functional.

The key actions of the plan include:

1. Correct the logic to obtain the most recent five years of data for any business older than five years. For any business younger than five years, the app should fill in any unavailable historical data with forecasted financial data.
2. Provide Error Handling to ensure the data is forecasted correctly if the business is under five years old.
3. Build the program using a modular software architecture to allow for easy iterative updates of the individual modules including the user interface, authentication, logic, loan profile, and data processing modules.
4. Develop the app using an agile methodology to enable iterative updates and frequent testing to ensure business requirements are met.

2. Functional requirements objective

The objective of the functional requirements that will be tested during the quality assurance phase will ensure that the application can identify and retrieve the correct financial data for a business based on age. The QA process will need to verify that:

1. The app will request the five most recent years of financial data, excluding the current year, for any business that is older than five years.
2. If a business is younger than five years, the application should use the available historical data and generate forecasted financial data to fill in a full five-year span.

Both objectives above will align with the software design plan by confirming that the logic is functioning correctly and providing an accurate loan profile.

2a. Functional requirements objective metrics

1. Data Accuracy: This will be a crucial metric to be sure that the correct fiscal years are selected based on the age of the business. The core functionality of the program relies on identifying and processing the correct data, which will align with the software design's logic for data retrieval based on age.
2. Response Time: As outlined in the software design, the web application should be able to retrieve financial data within 2 seconds. This is a metric directly related to the user experience, ensuring smooth operation and minimal frustration.
3. Error Handling: Measuring the application's ability to forecast the correct data in the case where a business is less than five years old is a key point to meet the business requirements. Effectively handling the errors for this portion of the process will be an achievable metric for the required functionality.

3. Non-functional requirements objective

1. Performance testing will need to take place to verify that the application can retrieve the data within the required timeframe (2 seconds). This will meet the requirement for a seamless user experience.
2. The modular structure of the application will need to be tested to be sure that any future updates or enhancements can be implemented without disruption to the application's functionality.

The non-functional requirements align with the software design plan by ensuring that the application functions correctly (as defined by the functional requirements) and delivers the expected level of performance, in addition to being able to scale for future changes or expanded features.

3a. Non-functional requirements objective metrics

1. Response time (latency) will be the metric associated with the performance. The application will need to retrieve data within 2 seconds, and this would be a convenient and easy metric to measure. Meeting the required response time will ensure a smooth user interface and maintain efficient loan processing as outlined in the design plan.
2. The completeness of the modular structure can be tested by verifying the scalability of the program to confirm that updates and enhancements can be completed without disruption to the overall functionality. This metric supports the sustainability of the application, ensuring it will be easily maintainable in the future.

These metrics will ensure the web application is fast, reliable, and adaptable to future change, which aligns with the objectives listed in the software design plan.

B. Scope

1. In-scope functional requirements

1. Testing to ensure the correct financial data is retrieved for a business that has an established date older than five years. If the application is performing correctly, it should request the most recent five years, excluding the fiscal year in which the test is taking place.
2. Verifying that the correct financial data is forecasted for any business younger than five years to obtain a full five-year compilation of the business's financial data to add to the loan profile.

Both requirements directly align with the functional objectives listed previously and these tests will be necessary to ensure the correct data is being gathered by the application.

2. In-scope non-functional requirements

1. Performance testing for response time (latency) will need to be verified to ensure the data can be retrieved within the 2-second response time window. This will help to provide a seamless user experience.
2. Modular structure testing for scalability would be needed to ensure the application can handle future updates or enhancements without any disruption to the functionality.

These requirements align with the non-functional objectives related to performance and scalability. With these verifications, the app will be confirmed to be running efficiently and remain adaptable to future changes.

3. Out-of-scope functionalities

Two out-of-scope functionalities that will not need to be tested as part of the quality assurance process would be redesigning the user interface and adding new features to the workflow. These features may be considered as part of another request, but do not fit the scope of this issue.

3a. Out-of-scope functionalities explanation

1. Reconfiguring or expanding the UI:
    a. While an enhanced user console could align with the business requirements by allowing easier data entry and retrieval, the primary business requirement is to fix the logic error with the incorrect data gathering, not to enhance the UI.
    b. Redesigning the user interface does not address the software bug causing the data retrieval logic issue. This task is focused on correcting the historical data and forecasted data rather than any part of the UI.
2. Adding new features to the workflow:
    a. This could align with business requirements by adding additional questions or information gathering to the loan profile. However, the key business requirement is to correct the data retrieval based on the business age. This functionality would not correct the logic programming.

b. This is out-of-scope because adding new features or workflow changes is not directly related to historical or forecasted data retrieval. That concern is the main scope of this issue.

C. Test Strategy

1. Testing overview

Test Case Table				
Test Type	**Description of Test**	**Objective**	**Test Owner**	**Environment**
Unit	Test to confirm that the application will perform as expected and return the financial data in the date range required. Sample Input: A company established more than five years ago. Expected Result: The five most recent years of financial data, excluding the current year (if the current year is 2025 it would provide data for 2020-2024).	The objective is to verify that the app retrieves the correct data for businesses older than five years. This logic is the core of this test, though other conditions will be added in future tests.	QA Tester	Local Development Environment with Mock Data
Unit	Test to verify that the application correctly forecasts financial data for a business younger than five years. Sample Input: The name of a business established less than five years ago (a business established in 2022 if the current year is 2025) Expected Result: The application forecasts the data correctly for the missing years (2024-2026 will be forecasted data in the example input above)	The objective of this test is to verify that the app forecasts the correct data for businesses younger than five years. Not yet testing the combination of historical and future data.	QA Tester	Local Testing Environment with Mock Data
Integration	Test to ensure data retrieval and forecasting logic is functional and will combine correctly into the loan profile. Sample Input: A business older than five years and a business younger than five years. Expected Result: The application gathers the five most recent years of historical data for the older business (excluding the current year) and correctly	This test will verify that the app combines the historical data with the forecasted data for businesses younger than five years. This is a combination of both previous tests as one combined unit.	QA Tester	Integrated Testing Environment with Database Access to Mock Data and Staging Server Access

	forecasts financial data for the business younger than five years to fill in gaps in the historical data available.			
System	Test to evaluate that the system can meet the required response time of 2 seconds for data retrieval. Sample Input: A request for financial data from both types of businesses (One older than five years and one younger). Expected result: Both types of business cases should be processed and data retrieved within 2 seconds.	Verifying that the program can retrieve data and operate within the latency requirement (2 seconds)	Performance Engineer	Staging Environment with Load Testing Tools (e.g., LoadRunner)

2. Sequence of testing

1. Unit Testing for Businesses more than 5 years old (Historical data):
    a. Justification: The ability to correctly retrieve the most recent five years of data (excluding the current year) for a business that is older than five years is a core component of the application. This must be verified first as the foundational step, before moving on to more complex components.
2. Unit Testing for Businesses less than 5 years old (Forecasted data):
    a. Justification: Validating the forecasted data is the next logical step in the process. This will ensure that the business-age conditions work correctly, and the forecasting logic is functional independently before moving on to further tests.
3. Integration Test for Data Gathering and Forecasting Capabilities:
    a. Justification: The third logical step in the testing process is to ensure that the app will combine the historical and forecasted data correctly. At this stage, we have verified that both work independently and are ready for combined testing.
4. System Test for to Measure Performance:
    a. The final test will focus on the measurement of latency now all of the components have been combined. This will test that the system meets the required response time of 2 seconds. Performance testing needs to be completed last to be sure the system is performing correctly under the expected use. Testing this earlier would not be an accurate measurement of performance as all the components would not be tested at once.

D. Sources

No external references were used.