**Quality Assurance Test Plan**

# A. Overview

## 1. Software design plan summary

The design plan was to fix the bug in Endothon Finance’s loan web app, where the wrong financial years were showing. The fix would update the app logic to show the most recent five fiscal years, not the first 5 years. If a business is newer, it would show the available years plus future forecast years.

To make this happen, the design plan said:

* **Development tools**: Visual Studio Code and Chrome DevTools for testing.
* **Programming language**: JavaScript (for frontend logic) and Python (backend APIs if needed).
* **Methodology**: Agile, using short sprints and code reviews with the Web App Team.
* **Deliverables**: Updated code, unit tests for the year logic, and a demo of the working loan profile submission.
* **Version control**: Git and GitHub were used to manage updates safely.

The idea was to fix the logic, clean it, test it properly, and ensure no other parts of the app break when the fix is pushed live.

## 2. Functional requirements objective

Make sure the app chooses the correct years based on when the business started:

* If the business is over 5 years old, show the five most recent full years (not the current year).
* If the business is under 5 years old, show the years they have + future years until it’s five.

### 2a. Functional requirements objective metrics

* **Year selection should be correct every time.** No weird years from the past.
* **The logic should work differently for old vs new businesses.** AC1 and AC2 both have to work.

These matter because they’re the rules from the ticket. Lenders won’t get the correct info if it picks the wrong years.

## 3. Non-functional requirements objective

* The right fields should show up quickly.
* It should work the same in all browsers.

### 3a. Non-functional requirements: objective metrics

* **Load time:** The year fields should pop up in under 2 seconds.
* **Browser test:** Gotta work the same in Chrome, Safari, Firefox.
* **No new bugs:** The fix shouldn’t mess up other parts of the app.

These things matter so users don’t get annoyed or confused while applying for loans.

# B. Scope

## 1. In-scope functional requirements

* Ensure the app asks for the **last five completed years** of financial data if the business is older than 5 years. (AC1)
* Ensure the app asks for **available years + future forecast** if the business is under 5 years old. (AC2)

These are the main things the bug messed up, so they must be tested.

## 2. In-scope non-functional requirements

* Check that the **right year fields show up fast** (under 2 seconds).
* Ensure the **logic works similarly in all major browsers** like Chrome, Firefox, and Safari.

These are important so the user experience is smooth, no matter where they are applied.

## 3. Out-of-scope functionalities

* The part where the app sends the loan profile to vendors.
* How do the vendors handle or rank the loan profile after they get it?

### 3a. Out-of-scope functionalities explanation

* The **vendor sharing system** is part of the loan processing, not the logic we’re fixing. So even though it’s connected to the business goal (getting the loan out fast), it’s not the problem we’re solving here.
* We cannot control or test vendor response behavior in this bug fix. That’s more about vendor systems, not our web app logic.

# C. Test Strategy

## 1. Testing overview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Table** | | | | |
| **Test Type** | **Description of Test** | **Objective** | **Test Owner** | **Environment** |
| System Test | Business older than 5 years. Start date: 2000. Expect: 2018–2022 fields only. | Make sure old businesses show recent years | QA Analyst | Chrome, Firefox, Safari |
| System Test | Business started in 2020. Expect: 2020–2022 + 2023–2024 forecast fields | Make sure new businesses get forecast years | QA Analyst | Chrome, Firefox, Safari |
| Performance | Measure how fast year fields load after entering start date | Show fields in <2 seconds | QA Analyst | Chrome DevTools |
| Compatibility | Run same test cases in 3 different browsers to compare behavior | Consistent logic across all browsers | QA Analyst | Chrome, Firefox, Safari |

## 2. Sequence of testing

1. **Start with functional testing first:** Test both types of business logic (AC1 and AC2) to ensure the logic is fixed and the right years show.
2. **Test performance:** After we know the right fields are showing, check how fast they load. This makes sure users don’t get stuck waiting.
3. **The last step is browser compatibility.** Run the same tests in Chrome, Firefox, and Safari to see if the behavior is consistent across all browsers.

This order makes sense because we need to first confirm the logic works, then make sure it’s fast, and finally check that it works everywhere.