Mortgage Monitor

Katherine Mulder

User Acceptance

Test Plan

Table of Contents

1 Introduction 1

1.1 Purpose 1

1.2 Background 1

2 Scope 2

2.1 User acceptance testing 2

2.2 Approach 2

2.3 Defect management 3

2.4 Risks 3

2.5 Issues 4

2.6 Dependencies 4

2.7 Assumptions 4

3 System under test 5

3.1 Application/system under test 5

4 Test case script 6

4.1 Test description 6

4.2 Initial conditions 7

1 Introduction

1.1 Purpose

This document describes a test plan for conducting user acceptance testing. The user acceptance objectives are:

1. verify function accuracy: ensure the system accurately calculates initial mortgage repayment, mortgage maturity, amortization schedules, and incremental payments based on user inputs.

2. Validate override payments: confirm that system correctly processes and calculates mortgage details when users input override payments.

3. assess user experience: evaluate the ease of use and overall user experience, ensuring the interface is easy to use and intuitive.

4. test interact graphics: ensure the interact graphical representation of interest and principal amortization is accurate and enhances user understanding.

1.2 Background

The Mortgage Calculator project provide users with a comprehensive tool to manage their mortgage details. Users can input their mortgage information, including the principal, term, interest rate, and deposit, to calculate their initial mortgage repayment, mortgage maturity date, and an amortization table. The system also supports incremental updates, allowing users to input override payments and see the impact on their mortgage.

The user acceptance test plan will cover various aspects of the system, including functional accuracy, usability, and performance of interactive features.

2 Scope

2.1 User acceptance testing

2.1.1 In scope

Functionality under test
initial mortgage calculation
mortgage maturity
override payment calculation
amortization table
increment planning table
graphics for Amortization
user settings

2.2 Approach

We will focus our testing efforts on the most important parts of the system that could cause the most problems if they do not work correctly, including ensuring the mortgage calculations are accurate, the override payments are processed correctly, and the interactive graphs work well. In addition to following set test cases, we will also perform some free-form testing to find unexpected issues, allowing testers to explore the system based on their understanding and intuition, which can help identify problems that structured tests might miss. This approach is especially useful for checking how easy the system is to use.

The main goals for UAT are to ensure the system works as expected and is ready for real use, confirm that the application is easy to use and provides a good experience for users, verify that all mortgage calculations, including payments, maturity dates, and schedules, are accurate, and test the interactive graphs and other dynamic features to ensure they perform well. We will plan and design our tests by first reviewing the project requirements to identify what needs to be tested, then developing detailed test cases that cover all functionalities, including potential problem areas, and finally preparing all the necessary data to simulate real-world use of the system.

We will ensure all parts of the system are tested by checking all features like mortgage calculations, override payments, and interactive graphs, evaluating usability, performance, and security, and using a checklist to make sure all requirements are tested. We will use test automation to make the testing process more efficient, automating tests that need to be run frequently to ensure new changes do not break existing functionality, and automating tests to check how well the system handles load

and stress. However, we will perform manual testing for areas that need human judgment, such as exploring the system freely and evaluating ease of use, because automation is not suitable for these tasks.

This approach ensures we thoroughly test the Mortgage Calculator/Monitor project, making sure it works well and is easy to use before we release it.

2.3 Defect management

Defect management for user acceptance testing will follow a structured approach. we will log any issues they find in a tracking system, providing details like a unique ID, description, steps to reproduce, severity, assigned tester, and screenshots. Defects will be categorized by severity into critical, high, medium, and low. Logged defects will be assigned to developers who can best handle the issue. Developers will work on fixing the defects and update the tracking system with the status and details of the fix. Once a defect is fixed, testers will re-check it to ensure it's resolved. If it's fixed, the defect will be marked as "Closed." If not, it will be reopened for further investigation. Regular reports will be generated to show the number of defects found, fixed, and pending, and these reports will be shared with stakeholders. We will review and improve the defect management process based on feedback and lessons learned during testing. This approach ensures all issues are tracked, prioritized, and resolved effectively, leading to a stable system ready for release.

2.4 Risks

A risk is something that could impact the test effort and can be described in terms of the likelihood of it occurring and the potential consequences if it does.

Ref	Risk description	Likelihood	Consequence	Rating	Controls – mitigation
1	Incorrect mortgage calculations affecting users' financial decisions	Likely	Severe	Very high	Thorough validation of calculation algorithms
2	System crashes or slow performance under load	Possible	Major	High	Load testing and performance optimization.
3	User interface issues affecting usability	Possible	Moderate	Medium	Usability testing and iterative design improvements based on user feedback.
4	Integration issues with external services	Possible	Moderate	Medium	Rigorous testing of integrations, having fallback mechanisms, and regular updates to handle external changes.

Ref	Risk description	Likelihood	Consequence	Rating	Controls – mitigation
5	Security vulnerabilities exposing sensitive user data	Unlikely	Severe	High	Regular security audits, use of secure coding practices, and implementing data encryption.

2.5 Issues

An issue is something that has been identified as currently affecting, or likely to affect test delivery capability, and that needs to be resolved or addressed.

Ref	Issue description	Urgency	Consequence	Rating	Controls – mitigation
1	Incomplete test data affecting test coverage	Imminent	Moderate	Medium	prioritize critical test cases.
2	Delays in receiving feedback from stakeholders	Intermediate	Minor	Low	Establish a clear communication plan and regular update meetings to ensure timely feedback.

2.6 Dependencies

The mortgage monitor project depends on several key factors. First, we need a stable version of the application with all features working properly. Second, we need accurate test data that covers a variety of mortgage scenarios, including different principals, terms, interest rates, deposits, and override payments.

We also need key stakeholders and end-users available to test the system and give feedback. Having the right testing tools, like defect tracking systems and performance testing tools, is essential.

Clear and constant communication between the development team, testing team, and stakeholders is important to quickly resolve any issues that come up. Additionally, any third-party services or integrations the application uses must be reliable and accessible during the testing period to ensure everything works as expected.

By managing these dependencies well, we can make sure the UAT process goes smoothly and results in a reliable and user-friendly mortgage monitor application.

2.7 Assumptions

We assume that will deliver a stable version of the application on schedule, with all intended features implemented and functioning correctly. It is also assumed that the test data provided will be comprehensive and representative of real-world scenarios, including various combinations of mortgage details such as principal, term, interest rate, deposit, and override payments.

We are assuming that key stakeholders and end-users will be available and willing to participate in the testing process, providing timely and constructive feedback. Additionally, we assume that all necessary testing tools and resources, such as defect tracking systems and performance testing tools, will be available and operational throughout the UAT period.

3 System under test

3.1 Application/system under test

The primary application under test is the Mortgage Monitor. In addition to this, several other systems are required to perform user acceptance testing. The table below lists the application and systems,

Application/system
Mortgage monitor
PostgreSQL Database
Web Server (Flask)

Additional information:

- No special hardware such as simulators or static generators is required for this UAT. Testing will be performed on standard computing hardware suitable for running web applications and databases.
- Testing will focus on functionality and usability of the Mortgage Monitor application. Any known limitations or constraints of the application or systems under test will be documented, and testing will be adjusted accordingly. For example, if certain features are still under development or certain integrations are not yet available, these areas will be noted and excluded from testing.

4 Test case script

4.1 Test description

Scenario and overview:	This test case checks if setting up a new mortgage in the web application works correctly. the scenario involves entering different mortgage details, making sure the system handles the data properly, and checking that all calculations and visuals are accurate. this is important to give users correct mortgage information and projections.
Requirements:	The system must collect these inputs: mortgage name, initial interest rate, initial term (in years), initial principal, deposit, extra costs, start date, comments, payment override enabled (boolean), monthly payment override (if applicable), fortnightly payment override (if applicable), increment amounts for different principal scenarios, and increment percentages for projection scenarios. it must check all user inputs for correctness and format, calculate the initial monthly and fortnightly payments based on user inputs, and calculate and store the mortgage maturity date. additionally, the system must create an amortization table showing each payment, interest, principal reduction, and remaining balance, calculate projected monthly and fortnightly payments based on increment amounts and percentages, and make visualizations of payment schedules and principal reduction using a library like plotly or matplotlib. the system must show the calculated initial payment breakdown, mortgage maturity date, amortization table, and visualized payment schedules and principal reduction, and log the initial transaction in the database as the creation of the mortgage.
Objectives:	The goals are to check that all required fields are collected correctly, make sure all user inputs are checked properly, and confirm the correct calculation of initial payments and the mortgage maturity date. The test should also check the creation of a correct amortization table, verify correct calculation of projected payments based on increments, and ensure accurate and visually appealing graphs. Additionally, it should confirm that all outputs are shown correctly and that the initial transaction is logged in the database.
Documents referenced:	The documents referenced include the functional requirements document for the mortgage application, user interface design specifications, data validation rules documentation, financial calculation formulas for mortgage payments, and documentation for Plotly and Matplotlib for creating visualizations.
Design notes:	While designing this test case, it's important to consider edge cases such as very large numbers or special characters. Tests should include both numerical and text input fields, cover all possible combinations of missing

	and invalid data, and include scenarios for both monthly and fortnightly payment overrides. It's also crucial to ensure that graphs are responsive and easy to use.
Notes for testers:	Testers should make sure the testing environment is correctly set up with the latest version of the application. Pay close attention to error messages to ensure they match the requirements and record any differences or unexpected behaviours' in detail. If a test case fails, provide as much information as possible to help with fixing the issue. Finally, make sure the database correctly logs the initial transaction.

4.2 Initial conditions

Step no.	Set-up action	Required result	Complete (tick)
1.	Ensure the testing environment is set up with the latest version of the web application.	The testing environment should have the latest version of the web application installed and running	✓
2.	Collect all necessary test data and inputs, including valid and invalid data sets.	Test data and inputs should be prepared and available for testing.	✓
3.	Verify access to the database and ensure it is reset to its initial state.	The database should be accessible and reset to a clean state before testing begins.	✓
4.	Ensure all referenced documents are available and accessible for reference.	All necessary documents (functional requirements, UI design specifications, data validation rules, etc.) Should be available for testers to reference.	✓