Loan Calculator Test - Angelo Facecchia

# Test Approach for the Loan Calculator

For this exercise, I will try to cover the user interface functionality and the proper operation of the calculation API.

**1. Functional Testing Approach**

Functional tests will verify that the calculator meets the specified requirements:

* **Manual UI Testing**: Manually test the user interface to verify that the calculator opens correctly, allows for the selection of loan amount and period, and updates the monthly payment and APRC in real time.
* **Save Behavior Validation**: Ensure that changes are not saved unless the save button is clicked.

**2. Integration Testing for the "calculate" Endpoint**

* Using Postman to test various scenarios for the "calculate" endpoint, including valid and invalid inputs, boundary conditions, and error simulation.
* API tests will verify that the endpoint returns correct values for the monthly payment and APRC and handles any errors appropriately.

**3. Automation with Playwright**

* Automate UI tests for the calculator using Playwright framework. I will automate
  + Opening the modal and selecting options.
  + Real-time updates of the APRC and monthly payment after a change.

# Test Scenario for the Loan Calculator

The main scenarios to include for UI testing:

* **Case 1: Modal Opening**

Verify that the calculator opens automatically when the page is loaded and can also be triggered by clicking the “Jätka” button.

* **Case 2: Loan Amount and Period Selection**

Verify that the user can select different loan amounts and periods, ensuring the values are entered correctly.

* **Case 3: Real-Time Update**

Verify that when changing the loan amount or period, verify that the calculator sends a request to the "calculate" endpoint and updates the monthly payment and APRC accordingly.

* **Case 4: Saving Changes**

Verify that changes are not saved unless the user clicks the save button.

* **Case 5: Close Button Functionality**

Verify that the calculator closes correctly and no save the changes.

**Test Case 1:**

* ***Description***:

The following test case will check that the calculator is opened automatically when the page is load or using the “Jätka” button.

* ***Precondition:***

The <https://taotlus.bigbank.ee/> page is opened.

* ***Test Steps:***

|  |  |
| --- | --- |
| *Description* | *Expected* |
| Check that the ‘Vali sovib summa ja periood’ modal is opened | The modal is opened |
| Check the UI of modal meets the UI requirements. | The UI requirements have been met |
| Click on ‘X’ icon on the right corner or clicking ‘Jätka’ button. | The modal is closed |
| Click on ‘Laenusumma’ on the right corner of the page | The ‘Vali sovib summa ja periood’ modal is opened |

**Test Case 2:**

* ***Description***:

The following test case will check that the user can select different loan amounts and period using the slider or editing the text field.

* ***Precondition:***

The <https://taotlus.bigbank.ee/> page is opened.

* ***Test Steps:***

|  |  |
| --- | --- |
| *Description* | *Expected* |
| Add a numeric value in the ‘Laenusumma’ field. | The numeric value has been added. |
| Add a numeric value in the ‘Periood’ field. | The numeric value has been added. |
| Try to add a value in the ‘Laenusumma’ field using the slider | The value has been added using the slider. |
| Try to add a value in the ‘Periood’ field using the slider | The value has been added using the slider. |

**Test Case 3:**

* ***Description***:

Verify that when changing the loan amount or period, verify that the calculator

sends a request to the "calculate" endpoint and updates the monthly payment

and APRC accordingly.

* ***Precondition:***

The <https://taotlus.bigbank.ee/> page is opened using the inspector mode.

* ***Test Steps:***

|  |  |
| --- | --- |
| *Description* | *Expected* |
| Try to add a numeric value in the ‘Laenusumma’ field and/ or in the ‘Periood’ field. | The numeric value has been added and the ‘calculate’ endopoint has been triggered in the ‘Network’ tab. |
| Check that the ‘calculate’ endopoint is visible. | The ‘calculate’ endpoint is visible and the values are correct. |
| Try to add a value in the ‘Laenusumma’ field and/ or in the ‘Periood’ field using the slider | The numeric value has been added and the ‘calculate’ endopoint has been triggered in the ‘Network’ tab. |
| Check that the ‘calculate’ endopoint is visible. | The ‘calculate’ endpoint is visible and the values are correct. |

**Test Case 4:**

* ***Description***:

Verify that changes are not saved unless the user clicks the save button.

* ***Precondition:***

The <https://taotlus.bigbank.ee/> page is opened.

* ***Test Steps:***

|  |  |
| --- | --- |
| *Description* | *Expected* |
| Try to add a numeric value in the ‘Laenusumma’ field and/ or in the ‘Periood’ field. | The numeric value has been added and the ‘calculate’ endopoint has been triggered in the ‘Network’ tab. |
| Click on the ‘X’ icon on the right corner of modal | The modal is closed and the data are not saved. |
| Click on ‘Laenusumma’ on the right corner of the page | The ‘Vali sovib summa ja periood’ modal is opened |
| Try to add a numeric value in the ‘Laenusumma’ field and/ or in the ‘Periood’ field and click on ‘Jätka’ button | The modal is closed and the value added in the ‘Laenusumma’ is displayed. |

**Test Case 5:**

* ***Description***:

Verify that the calculator closes correctly and no save the changes.

* ***Precondition:***

The <https://taotlus.bigbank.ee/> page is opened.

* ***Test Steps:***

|  |  |
| --- | --- |
| *Description* | *Expected* |
| Try to add a numeric value in the ‘Laenusumma’ field and/ or in the ‘Periood’ field. | The numeric value has been added and the ‘calculate’ endopoint has been triggered in the ‘Network’ tab. |
| Click on the ‘X’ icon on the right corner of modal | The modal is closed and the data are not saved. |

# API verification of ‘calculate’ endpoint for the Loan Calculator

The test scenarios used for ‘calculate’ endpoint are:

**Valid Input - Standard Values**

**Description**:

Send a request with standard values (e.g., amount=5000, period=60) within typical bounds.

**Expected Outcome**:

The API should return a valid response with correct monthly payment and APRC values.

**Reasoning**:

This tests the normal functionality of the endpoint with average values to ensure it calculates correctly under standard conditions.

**Boundary Values**

**Description**:

Test the minimum and maximum values for the amount and period fields (e.g., amount=500, period=12 and amount=30000, period=120).

**Expected Outcome**:

The API should return correct calculations for both low and high boundary values.

**Reasoning**:

Boundary testing helps ensure that the endpoint handles edge cases within acceptable limits. It confirms that the API behaves as expected at the extreme ends of input.

**Invalid Input - Amount or Period Too Low**

**Description**:

Test with values below the minimum limits (e.g., amount=10, period=1).

**Expected Outcome**:

The API should return an error response or validation message indicating that the input is outside acceptable limits.

**Reasoning**:

Ensures that the API enforces limits and handles invalid input gracefully, preventing calculations with unrealistic values.

**Invalid Input - Amount or Period Too High**

**Description**:

Test with values above the maximum limits (e.g., amount=100000, period=300).

**Expected Outcome**:

The API should return an error message indicating that the input exceeds the allowed limits.

**Reasoning**:

Similar to the previous case, this confirms the API’s ability to handle out-of-bounds input and provide appropriate error messages.

**Zero or Null Values**

**Description**: Send amount=0 or period=0, or null values (if the API accepts nulls).

**Expected Outcome**:

The API should return an error message or default to minimum values, depending on its configuration.

**Reasoning**:

Tests the API’s handling of potentially problematic inputs, ensuring that zero or null values are managed properly.

**Negative Values**

**Description**:

Send negative values for amount and period (e.g., amount=-5000, period=-12).

**Expected Outcome**:

The API should return an error, as negative values are not valid for loan calculations.

**Reasoning**:

Negative testing helps verify that the API does not accept unrealistic values, enhancing data validation and reliability.

***Test Collection and Report Result:***

For the test collectionand result please see the files attached:

* **Test Collection**: *BigBank - Loan Calculator API test.postman\_collection.json*
* **Test Report:** *BigBank - Loan Calculator API test.postman\_test\_run.json*

# Automated tests of the feature

These automated test cases cover essential UI functionalities for the loan calculator:

1. **Opening and closing** the calculator modal.
2. **Real-time updates** of the monthly payment when the loan amount changes.
3. **Boundary testing** for minimum and maximum values of the loan amount field.

**Test Case 1: Open and Close the Calculator Modal  
  
Objective**: Verify that the calculator modal opens and closes correctly. **Steps**:

1. Navigate to the calculator page with default parameters.
2. Locate the calculator modal and verify that it is visible.
3. Locate the close button within the modal.
4. Click the close button.
5. Verify that the modal is no longer visible after closing.

**Expected Result**: The calculator modal should be visible upon loading the page, and it should close successfully when the close button is clicked.  
**Reasoning**: This test ensures that the modal can be opened and closed properly, verifying basic functionality for user interaction.

**Test Case 2: Adjust Loan Amount and Check Real-Time Updates**

**Objective**: Verify that the monthly payment updates in real-time when adjusting the loan amount.  
**Steps**:

1. Navigate to the calculator page with a custom parameter for amount set to 10000.
2. Locate the loan amount input field and set the value to 10000.
3. Locate the monthly payment display field.
4. Verify that the monthly payment field is populated and displays a value (indicating that it updated in response to the loan amount change).

**Expected Result**: The monthly payment field should show a calculated value, indicating that real-time updates work correctly when adjusting the loan amount.  
**Reasoning**: This test ensures that the loan calculator dynamically updates the monthly payment field based on the selected loan amount, providing real-time feedback to users.

**Test Case 3: Check Minimum and Maximum Values for Loan Amount**

**Objective**: Verify that the loan amount input field correctly accepts and displays the minimum and maximum allowed values.  
**Steps**:

1. Navigate to the calculator page with default parameters.
2. Locate the loan amount input field.
3. Set the loan amount to 500 (the minimum value) and verify that the input displays 500.
4. Set the loan amount to 30000 (the maximum value) and verify that the input displays 30000.

**Expected Result**: The loan amount field should display 500 when set to the minimum value and 30000 when set to the maximum value.  
**Reasoning**: Boundary value testing ensures that the loan calculator handles both minimum and maximum values correctly, providing accurate inputs for users and preventing errors with out-of-bounds values.