Marcus Rappenborg Kjærsgaard
13-12-2025

Multi-Currency Wallet Simulator
Usability Testing Design (not executed)

1. Purpose

This document defines a usability test design for the Multi-Currency Wallet Simulator. The goal is to evaluate whether a first-time user can complete the core workflows (wallet creation, deposit, withdrawal, exchange, and transaction history) and correctly understand the system's feedback (success states, error states, and what changed).

2. Scope

In scope (core user workflows):

- Create a wallet in a chosen currency
- Deposit money into a wallet
- Withdraw money from a wallet (including insufficient funds case)
- Exchange money between two wallets
- View and interpret transaction history

Out of scope:

- Visual preferences (colors, "pretty UI") beyond whether it supports task completion
- Long-term use patterns (weeks/months of usage)
- Accessibility compliance audit (this test can still surface obvious accessibility barriers, but it is not a full audit)

3. Target users

The test targets users similar to the expected audience of the project:

- Students or casual users with basic web-app familiarity
- No special financial or currency-exchange knowledge required

Recommended sample: 3–5 participants (sufficient to find the most common usability issues in a small system).

4. Test method and setup

Method: Moderated, task-based usability test with think-aloud.

Participants are asked to say what they expect to happen and what they believe happened after each action.

Environment:

- Laptop or desktop, modern browser (Chrome/Edge/Firefox)
- Application running in a stable "test mode" setup (to avoid unpredictable external factors)
- Session length: ~15 minutes per participant

Materials:

- This script (tasks + prompts)
- A simple results sheet for time, success, errors, and notes
- Post-task rating questions (see section 6)

5. Roles

Moderator responsibilities:

- Read instructions, encourage think-aloud
- Avoid leading the participant toward "correct" actions
- Capture errors, hesitation points, and participant expectations

Participant responsibilities:

- Perform tasks as naturally as possible
- Think aloud when uncertain and after seeing results or errors

6. Measures

6.1 Performance measures (objective)

Collected per task:

1. **Time on task (seconds)** – time from task start until completion or abandonment
2. **Task success** – Completed / Partially completed / Not completed
3. **User errors** – count of incorrect actions (e.g., wrong wallet selected, wrong field used, repeated attempts)
4. **Hesitation / stuck time** – noticeable pauses where the participant is unsure what to do next
5. **Number of attempts** – how many tries were needed before success (if any)

6.2 Preference measures (subjective)

Collected after each scenario using a 1–7 Likert scale (1 = strongly disagree, 7 = strongly agree):

- "This scenario was easy to complete."
- "The UI was easy to understand while doing the task."
- "The feedback/messages were clear."
- "I understood what changed in the system after my actions."
- "The system felt fast enough for this task."
- "I could complete the task without unnecessary steps."
- "I feel confident using this feature again without help."

After the full test (overall):

- "Overall, the system was easy to use."
- "I would trust the system to correctly reflect my actions."

7. Test script

7.1 Moderator introduction (read aloud)

"Thanks for joining. Today we're testing the wallet simulator. This is a test of the system, not of you. Please think aloud as you work-say what you expect to happen and what you're looking for.

If something is confusing, it's useful information. I may ask 'What are you thinking?' but I won't tell you what to do."

7.2 General rules

- The moderator should not correct mistakes during tasks.
- If the participant is stuck for ~60 seconds, the moderator may ask:

"What would you do next if you were alone?"

- If still stuck, the moderator can provide a minimal hint and record that assistance was needed.

8. Scenarios and tasks

Scenario 1 - Create wallet and deposit (happy path)

Context: "You want to start tracking money in a DKK wallet and add funds."

Tasks:

1. Create a new wallet in **DKK**.
2. Deposit **100.00 DKK** into the wallet.
3. Confirm that the wallet balance changed correctly.
4. Find where the system shows the deposit was recorded (transaction or history view).

Success criteria:

- Wallet created successfully
- Deposit succeeds

- Updated balance is visible and understandable
- Participant can locate a record of the deposit

What the moderator records:

- Whether the participant understands where "current balance" is shown
- Whether the participant expects an entry in history and can find it

Scenario 2 - Withdrawal with insufficient funds (negative path)

Context: "You try to withdraw more than you have and need to understand the result."

Tasks:

1. Attempt to withdraw an amount that should fail (e.g., more than the current balance).
2. Identify what the system communicates (status/error message) and whether the balance changed.
3. Withdraw a valid amount that should succeed.
4. Confirm the balance changed appropriately.

Success criteria:

- The failing withdrawal is clearly communicated as a failure
- The participant understands why it failed (e.g., insufficient funds)
- No unintended balance change occurs on the failed withdrawal
- A successful withdrawal can be completed afterward

What the moderator records:

- Whether the participant interprets failure messages correctly
- Whether the participant can explain whether "nothing happened" vs. "something happened but failed"

Scenario 3 - Exchange between wallets (happy path with dependency)

Context: "You want to convert DKK to USD and confirm the results."

Tasks:

1. Create a **USD** wallet (if it does not already exist).
2. Exchange **10.00 DKK → USD** from the DKK wallet to the USD wallet.
3. Verify that both wallets were updated (source decreased, target increased).
4. Find the exchange transaction in the transaction history and interpret it.

Success criteria:

- The participant completes the exchange
- The participant can verify both sides of the exchange
- The participant can locate and understand the exchange transaction

What the moderator records:

- Whether the participant understands which wallet is source vs. target
- Whether the participant can interpret credited amount and currency

Scenario 4 - Transaction history interpretation (understanding + traceability)

Context: "You need to confirm what happened earlier."

Tasks:

1. Open transaction history for the DKK wallet.
2. Find the most recent **deposit**, **withdrawal**, and **exchange** transactions.
3. For each, explain:
    - transaction type
    - status (success/failure)
    - what changed (balance before/after, credited amount, etc., if shown)

Success criteria:

- Participant can find transaction history
- Participant can correctly interpret at least the type + status for each transaction
- Participant can explain outcomes in plain language

What the moderator records:

- Confusion about statuses or terminology
- Whether ordering and filtering (if present) helps or blocks comprehension

9. Post-test interview questions (short)

Ask these after all scenarios:

1. “What was the most confusing part of the system?”
2. “Where did you feel least confident about what happened?”
3. “If you could change one thing, what would it be?”
4. “Was anything missing that you expected to be able to do?”

10. Data capture template (for the moderator)

For each scenario:

- Start time: ___ End time: ___ Time on task: ___
- Result: Completed / Partial / Not completed
- Errors observed (count + notes): ___
- Hesitation/stuck moments (notes): ___
- Participant rating (1–7):
  - Ease: ___
  - Clarity of feedback: ___
  - Confidence: ___