

1 Multi-Currency Wallet Simulator (SRS v1)

2

Summary Review Report

3

1. Review Overview

4 Formal document review of the Software Requirements Specification (SRS) v1 for the *Multi-Currency*
5 *Wallet Simulator* web-application.

6 Objectives:

- 7 • Identify ambiguities, inconsistencies, and missing requirements that could hinder implemen-
8 tation and test design.
- 9 • Assess the **testability** of both functional and non-functional requirements, including error
10 handling, usability, and performance aspects.
- 11 • Provide input for revising SRS v1 into a clearer, more testable SRS v2.

12 Scope of review:

- 13 • Entire SRS v1 document:
14 ○ 3 pages with line numbering.

15 Meeting Information:

- 16 • Date: 21st November 2025
17 • Time / Duration: 1-hour 15min
18 • Consolidation and report writing: ≈ 15min

19

2. Participants and Roles

| Name | Role | Responsibility |
|--------|-----------|---|
| Marcus | Manager | Decided to review the SRS, defined scope and objectives |
| Marcus | Author | Write SRS v1, responsible for rework/producing SRS v2 |
| Marcus | Moderator | Planned and guided the review process, accepted/adjusted identified defects |

| | | |
|-------------------------------------|-----------|--|
| Victor | Scribe | Performed systematic review, identified defects and testability issues |
| Isabella, ChatGPT, (Marcus), Claude | Reviewers | Performed systematic review, identified defects and testability issues |

1 3. Defined Severity Levels

2 Defects were classified using four severity levels:

- 3 ▪ **Critical**: Defect that can cause seriously wrong behaviour, unsafe assumptions, or block meaningful test design.
- 4 ▪ **Major**: Defect with significant impact on clarity or completeness; different teams might reasonably implement different behaviours based on the current wording.
- 5 ▪ **Minor**: Defect that slightly harms clarity, precision, or consistency, but is unlikely to cause major behavioural divergence.
- 6 ▪ **Cosmetic**: Stylistic, formatting, or spelling issues that do not affect behaviour or test design.

10 4. Defect Categories

11 Each defect is tagged with one or more categories:

- 12 ▪ **A** - Ambiguity / Clarity
- 13 ▪ **M** - Missing / Incomplete Requirement
- 14 ▪ **C** - Consistency / Contradiction
- 15 ▪ **T** - Testability / Measurability
- 16 ▪ **S** - Scope / Assumptions
- 17 ▪ **F** - Formatting / Structure

18 These categories help analyse where requirements quality is weakest (e.g. many issues in testability vs. many in scope/assumptions).

1 5. Defect Summary by Severity

| Severity | Count | IDs |
|----------|-------|---|
| Critical | 0 | |
| Major | 7 | D-01, D-04, D-05, D-07, D-12, D-17, D-19 |
| Minor | 15 | D-02, D-03, D-06, D-08–D-11, D-13–D-16, D-18, D-20–D-22 |
| Cosmetic | 0 | |

2 Total defects identified: 22

3 No Critical defects were found; the main issues are Major (clarity and testability) and Minor (refine-
4 ments and scope).

5 6. Overall Reflections and Recommendations

6 Strengths of SRS v1:

- 7 ▪ Clear domain and scope: a limited, simulation-only multi-currency wallet system with three
8 currencies.
- 9 ▪ Functional behaviour is generally well-structured.
- 10 ▪ Error handling is conceptually sound: invalid input and external API failure do not modify
11 balances.
- 12 ▪ Non-functional aspects explicitly mention usability, performance, and testability, which is
13 very relevant for a testing-focused course.

14 Main weaknesses identified:

- 15 1. Ambiguous or incomplete core concepts
 - 16 ○ User / wallet ownership model (single vs multi-user).
 - 17 ○ Wallet status state machine and whether “closed” is a final state.
 - 18 ○ Detailed behaviour of exchanges (rounding, same-currency transfers).

1 **2. Non-functional requirements not measurable enough**

- 2
 - 3 ○ Usability phrased as “understandable without prior training” without performance or preference measures.
 - 4 ○ Performance described as “reasonable response time” without concrete thresholds.
 - 5 ○ Stress testing mentioned without defined load or acceptance criteria.

6 **3. Scattered or implied definitions**

- 7
 - 8 ○ “Invalid input” not fully tied to all validation rules.
 - 9 ○ “Sensitive payment information” not concretely defined.
 - 10 ○ Architecture constraints duplicated and mixed with testability.

11 **Recommended next steps:**

12 **1. Address all Major defects first**

- 13
 - 14 ○ This will significantly improve clarity and testability of both functional and non-functional requirements.

15 **2. Refine Minor defects where effort is reasonable**

- 16
 - 17 ○ Especially those that clarify scope and test conditions (same-currency exchanges, browser support, persistence, stress tests).

18 **7. Detailed Defect List**

19 Locations follow the format [page, line] as in the SRS (e.g. p. 1, l. 27).

20 The Title gives a short label for each defect; Recommendation is a concise suggestion for rework.

21 See next pages.

| ID | Severity | Category | Location | Title | Comment / Description | Recommendation |
|------|----------|----------|---------------------------|-------------------------------|--|--|
| D-01 | Major | S, A | p. 1, l. 5–7; 14–18 | User / wallet ownership model | “The user” is used without defining whether the system is <i>single-user</i> or <i>multi-user</i> or how wallets are owned. | Explicitly define whether the simulator is single-user or multi-user and how wallets are owned/isolated. |
| D-04 | Major | M, T | p. 1, l. 28; p. 2, l. 1–4 | Deposit validation rules | Validation rules are described for <i>withdrawals/exchanges</i> , but not clearly for <i>deposits</i> and initial deposits (e.g. 0 or negative amounts). | Make explicit how the system handles non-positive deposit and initial-deposit requests (reject + error). |
| D-05 | Major | T, M | p. 2, l. 11–15 | Exchange rounding strategy | Rounding is said to be “clear and deterministic” but the exact rule, precision, and rounding step are not specified. | Specify the exact rounding rule, precision (e.g. 2 decimals), and when rounding is applied in the calculation. |
| D-07 | Major | M, A | p. 2, l. 16–20 | Wallet status state machine | Allowed status transitions are given only as examples, and it is not explicit that “closed” is a final state. | Define the allowed status transitions and state clearly that “closed” is a terminal state with no further changes. |

| | | | | | | |
|------|--------------|------|---------------------|--------------------------------------|---|--|
| D-12 | Major | T, A | p. 3, l. 2–4 | Error re-response definition | The phrase “clear error response” is used without specifying HTTP status codes or response body structure. | Define HTTP status code range and a minimal JSON structure for invalid input errors (e.g. errorCode + message). |
| D-17 | Major | T, A | p. 3, l. 12–14 | Usability goal measurability | “Understandable without prior training” is a qualitative goal with no measurable success criteria for usability tests. | Replace “understandable without prior training” with measurable usability criteria (task success rate, time, errors). |
| D-19 | Major | T, A | p. 3, l. 15–17 | Performance target vagueness | “Reasonable response time” and “typical operations” are vague; no concrete response time targets are given. | Define concrete response-time targets for key operations in the intended local/demo environment. |
| D-02 | Minor | M, F | p. 1, l. 12–13 | Architecture: overview vs constraint | The architecture (frontend, API, DB, FX API) is described, but it is unclear if this is a strict constraint or just an overview. | Clarify that this architecture is a design constraint (or move it to a dedicated constraints subsection and reference it). |
| D-03 | Minor | M, S | p. 1, l. 4–7; 11–21 | Persistence and data lifetime | The use of a relational database is mentioned, but there is no explicit statement about persistence across restarts or retention. | State whether wallets and transactions persist across application restarts and whether there is any retention policy. |

| | | | | | | |
|------|--------------|------|------------------------|-------------------------------------|--|--|
| D-06 | Minor | A, S | p. 2, l. 3–7; 11–12 | Same-currency exchange behaviour | Exchanges “optionally in a different currency” are defined, but behaviour for same-currency transfers (e.g. DKK→DKK) is not specified. | Specify behaviour for same-currency transfers (e.g. effective rate 1.0, no external API call) to avoid ambiguity. |
| D-08 | Minor | S, A | p. 2, l. 16–20 | Who changes wallet status | The SRS does not state whether status changes are done by end users, admins, or automatically, or through which interface. | Define whether status changes are initiated by end-users, admins, or system logic, and through which interface. |
| D-09 | Minor | A, T | p. 2, l. 21–22 | Stored amounts for exchanges | For exchange transactions, it is unclear whether both source and target amounts (with currencies) are stored or only one amount. | Require that exchange transactions store both the debited and credited amounts with their currencies, in addition to balances. |
| D-10 | Minor | M, T | p. 2, l. 25–27 | Transaction history ordering/scope | The history view is mentioned but the sort order and which transactions (source, target, failed) are included are not defined. | Define sort order (e.g. newest first) and whether the history includes both source and target roles and failed transactions. |
| D-11 | Minor | A, T | p. 1, l. 28 | “Validated” initial deposit wording | “Validated” initial deposit is referenced without explicitly stating which validation rules apply. | Tie “validated” explicitly to the deposit validation rules (and any extra constraints, if they exist). |

| | | | | | | |
|------|--------------|------|----------------|--|--|---|
| D-13 | Minor | M, C | p. 3, l. 2–4 | Definition of “invalid input” | Examples of invalid input are given (“such as...”) but it is not clearly linked to all validation rules in Section 3. | Link “invalid input” to all violations of the validation rules in Section 3 and clarify that the list given is non-exhaustive. |
| D-14 | Minor | T, A | p. 3, l. 5–8 | Cached exchange rate behaviour | The system may reuse “recently obtained” rates, but “recently” and cache behaviour are not defined beyond “no invented rates”. | Clarify that caching is optional, but any cached rate must come from a real prior API response (no invented values). |
| D-15 | Minor | A, T | p. 3, l. 5–7 | “Data that cannot be processed” | The phrase “data that cannot be processed” is broad and does not give examples of what is considered unprocessable. | Give examples of unprocessable data (e.g. invalid JSON, missing fields, unsupported currencies, non-positive or non-numeric rates). |
| D-16 | Minor | S, T | p. 3, l. 9–10 | Scope of “sensitive payment information” | “Sensitive payment information” is not defined, and it is unclear whether DB, logs, and client-side storage are all included. | Define which real-world payment data must never be accepted or stored and extend this to DB, logs, and client storage. |
| D-18 | Minor | S, T | p. 3, l. 12–13 | “Modern desktop browser” scope | “Modern desktop browser” is vague; supported browsers/OS versions for testing are not specified. | Specify at least a minimal supported browser/OS set (e.g. Chrome + Edge on Windows, Chrome on macOS). |

| | | | | | | |
|------|-------|------|-----------------------|---|--|---|
| D-20 | Minor | T, M | p. 3, l. 17–18 | “Testable with automated tools” vagueness | Saying it “shall be testable with automated tools” adds little without stating which properties enable automation. | Specify properties that support automation, such as stable API endpoints and a way to create/reset test data. |
| D-21 | Minor | C, F | p. 3, l. 19–22 | Duplicated architecture constraint | The architecture constraint appears both in the overview and non-functional sections, creating duplication and consistency risk. | Avoid duplication by keeping architecture in one section and referencing it from the non-functional/testability part. |
| D-22 | Minor | T, M | p. 3, l. 17–18; 21–22 | Undefined stress test conditions | “Stress performance tests” are mentioned without specifying load levels or acceptance criteria. | Define a simple stress scenario (e.g. requests/sec or concurrent users) and acceptance criteria (no crashes, no data corruption). |