3. A test plan document, outlining how your tests are organized (in directories or whatever), how they will be run (as shell scripts), and how the output will be stored and organized for reporting and comparison with later runs (make text file printouts of any directory structures and script files created)

Front End Banking System - Requirements Test Plan

1. Purpose and Scope

A complete, requirements-based test plan for the Front End of the banking System. The goal is to verify that every required behaviour in the system is checked for accuracy, performance and precision:

- Normal Operations

- Bounding conditions

- Errors and invalid inputs

- Standard vs Admin usages for users

So far this document is only for the front end. No back end logic and integration is assumed at this stage. As per requirement no programs or code is implemented here. Only test inputs, expected outputs, and organized structures are defined.

All tests are designed to check every vulnerability the front end may have. Each one will have its own single input stream file with outputs also recorded.

2. Test Strategy Overview

<u>Black Box, Requirements Driven test structure</u>

- Tests are created directly from the customer requirements and formal constraints

- So far only observable behavior matters, since the back end is not implemented

<u>Session Based testing</u>

- Each test file represents one complete Front End session:

    - >=0 invalid attempts

    - Valid login

- >=0 transactions
- Invalid transaction
- logout

Output comparisons

- each test produces a bank account transaction file

- output files are compared against expected results

3. Directory/ File structure

```
frontend-tests/
├── plan/
│   ├── test_cases.pdf
│   └── test_plan_document.pdf
│
├── input/
│   ├── login/
│   ├── withdrawal/
│   ├── transfer/
│   ├── paybill/
│   ├── deposit/
│   ├── create/
│   ├── delete/
│   ├── disable/
│   ├── changeplan/
│   ├── logout/
│   └── robustness/
│
├── expected-output/
│   ├── login/
│   ├── withdrawal/
│   ├── transfer/
│   ├── paybill/
│   ├── deposit/
│   ├── create/
│   ├── delete/
│   ├── disable/
│   ├── changeplan/
│   ├── logout/
```

```
│ └── robustness/
│
├──actual-output/
│ ├── withdrawal/
│ ├── transfer/
│ ├── paybill/
└── ...
├── accounts/
│ ├── bank_accounts.txt
│ └── accounts_transaction_files.txt
│
└── scripts/
├── run-all-tests.sh
├── run-category.sh
└── diff-results.sh
```

4. Test Input and Output Organization

Input files

- located in input/

- Each file:

    - Plain text

    - One transaction or response per line

    - Represents a complete front end session

Naming convention:

<requirement>_<scenario>_<id>.txt (N/A is ambiguous for standard or admin)

Example:

withdrawal_standard_max_11.txt

Expected output files

- Located in expected-output/

- One expected transaction log per input test

- File name matches input file

Example:

expected-output/withdrawal_standard_max_11.txt

Bank Account files

Stored in accounts/ and copied into the runtime directory before each test.

Scenarios:

- Multiple users

- Disabled accounts

- Zero-balance accounts

- Maximum balance accounts

- END_OF_FILE

5. Test Run Plan

Execution model

Tests are executed non-interactively:

frontend < input/testfile.txt > actual-output/testfile.out

Scripts

- run-all-tests.sh
    - Iterates through all input directories
    - Runs every test file
    - Captures output
- run-category.sh <category>
    - Runs only one transaction category

- diff-results.sh

    - Compares actual-output/ against expected-output/

    - Reports pass/fail per test

Actual Storage

```
actual-output/
├── withdrawal/
│ ├── actual_withdrawal_N/A_valid_account_10.txt
│ ├── actual_withdrawal_standard_max_11.txt
│ └── actual_withdrawal _N/A_negative_12.txt
├── transfer/
├── paybill/
└── ...
```

Each test run is timestamped for regression testing into a singular file for each and every frontend requirement. These actual outputs will be dated not the files.

6. Requirement Problems

Deposit visibility - It wasn't specified in the requirements whether or not that the deposited money should display the updated account balance or balance before the deposit was made. It did however say that the deposited new balance can not be used until after the current session.

Multiple accounts under one name - Requirements never specified how many accounts can be under one name

Case sensitivity - Requirements never specified if the company name for the bill payee had to be case sensitive similar to the account holders name in bank accounts file or account transactions file

Transaction files - Never specified whether a transaction file should still be made if all you do is login and logout

7. Traceability Matrix (Summary)

Requirement	Test Category
Login rules	login/
Privilege enforcement	robustness/, create/, delete/
Transaction limits	withdrawal/, transfer/, paybill/

File formatting	expected-output/*
Error handling	robustness/