## 1. Test Input and Output Organization

The test cases are well-organized in a structured folder system to ensure clarity and ease of execution. The organization follows a hierarchical structure, separating inputs, expected outputs, and transaction logs.

### Folder Structure (Generated using tree from Homebrew)

(current\_accounts\_file.txt) (Shared input file for all test cases)

inputs

│ ├── 00\_login\_inputs

│ ├── 01\_withdrawal\_inputs

│ ├── 02\_transfer\_inputs

│ ├── 03\_paybill\_inputs

│ ├── 04\_deposit\_inputs

│ ├── 05\_create\_inputs

│ ├── 06\_delete\_inputs

│ ├── 07\_changeplan\_inputs

│ ├── 08\_disable\_inputs

│ ├── 09\_logout\_inputs

outputs

│ ├── 00\_login\_outputs

│ ├── 01\_withdrawal\_outputs

│ ├── 02\_transfer\_outputs

│ ├── 03\_paybill\_outputs

│ ├── 04\_deposit\_outputs

│ ├── 05\_create\_outputs

│ ├── 06\_delete\_outputs

│ ├── 07\_changeplan\_outputs

│ ├── 08\_disable\_outputs

│ ├── 09\_logout\_outputs

transaction\_outputs

│ ├── 01\_withdrawal\_transaction\_outputs

│ ├── 02\_transfer\_transaction\_outputs

│ ├── 03\_paybill\_transaction\_outputs

│ ├── 04\_deposit\_transaction\_outputs

│ ├── 05\_create\_transaction\_outputs

│ ├── 06\_delete\_transaction\_outputs

│ ├── 07\_changeplan\_transaction\_outputs

│ ├── 08\_disable\_transaction\_outputs

│ ├── 09\_logout\_transaction\_outputs

### 

### 

### Organization Approach

1. Shared current\_accounts\_file.txt  
   This file remains constant for all test cases unless explicitly modified for specific tests.
2. Test Inputs (inputs/)
   * Each category of transactions is stored in its own subfolder.
   * Test files are named consistently, e.g., changeplan01.inp, withdrawal05.inp, etc.
3. Expected Outputs (outputs/)
   * Each test case has a corresponding expected output .out file stored in its respective transaction folder.
4. Transaction Outputs (transaction\_outputs/)
   * Each test case has an expected transaction log file .etf stored in the transaction folder.

## 2. Test Run Plan

The following automated test execution strategy ensures smooth running of test cases.

### Execution Steps

1. Prepare the Testing Environment
   * Ensure the current\_accounts\_file.txt is available and correct.
   * Ensure the program can read inputs from inputs/.
   * Clear previous output files (if necessary).
2. Batch Execution of Tests
   * Run all test cases sequentially using a test runner script:
     + ./run\_tests.sh
   * This script:
     + Loops through all .inp files in inputs/
     + Runs them through the banking system program
     + Stores outputs in the outputs/ and transaction\_outputs/ directories.
3. Validation of Test Results
   * Compare actual output files against expected output files.
   * Use a diff tool to highlight mismatches:
     + diff outputs/expected\_output.out outputs/actual\_output.out
   * If differences are found, flag the test as failed.
4. Logging Results
   * Each test case will be marked as:
     + Test Passed: withdrawal\_01
     + Test Failed: deposit\_03 (Output mismatch)
   * Store logs in test\_results.log.
5. Re-run Failed Tests (if necessary)
   * Identify failed tests and run them individually.