Assumptions

This document details the main assumptions and design choices made during the development of the Acme Teller System. These choices were made to fulfil the project brief while ensuring technical reliability and logical consistency.

1. System Scope & Users

The system is used exclusively by bank tellers, not customers directly. Only one teller uses the system at a time (no concurrent multi-user access). Teller authentication is out of scope, and teller access is assumed to be authorised by the bank. The system runs as a local Java console (CLI) application, not a web or GUI system

2. Authentication

Requirement: "The authentication process does not need real-world encryption or secure login."

Assumption: We assume that tellers are already authenticated on their workstations. Therefore, "Customer Authentication" in this application specifically refers to identifying the customer by their Customer ID. We do not require customer PINs or passwords, as tellers verify identity using physical photo ID and address documents. The authentication process is logical rather than encrypted or password-based, as permitted by the brief.

For proof of authentication, the documents comply with the simple UK format. For example, the passport needs 9 digits, the National Insurance Number needs to start with 2 characters followed by 6 digits, and the last character must be a letter.

3. Account Number & Sort Code Generation

Requirement: "Guarantee uniqueness... Handle collisions."

Account numbers can be generated randomly by the system, with uniqueness guaranteed by checking existing accounts before assignment.

Assumption: We used a random 8-digit number generator. To ensure uniqueness, the system checks the database before assigning a number. We assume a limit of 100 re-tries is sufficient to find a unique number. If 100 attempts fail (which is statistically nearly impossible), the system throws an error to prevent an infinite loop.

Sort codes are hard-coded based strictly on account type and cannot be modified after creation.

4. ISA Interest Calculation

Requirement: "Calculate annual interest at 2.75% APR based on the average annual balance, or a simplified equivalent."

ISA interest is calculated using a simplified annual average balance model rather than daily compounding.

Assumption: We have implemented the "simplified equivalent" approach, which means the system calculates interest at a rate of 2.75% based on the current balance at the time when the teller initiates the "Apply Interest" process.

To prevent excessive applications of interest (applying multiple times in a single day), the system enforces a strict "Once per Calendar Year" rule. It checks the database for any 'Interest'

transaction associated with the account that has a timestamp within the current year (YYYY). If such a transaction is found, the operation will not be conducted.

5. Business Account Types

Requirement: "Require verification of an eligible business type. Reject excluded types (Enterprise, PLC, Charity, Public Sector)."

Assumption: We assume that the only eligible types of business structures are "Sole Trader" and "Ltd." The system strictly validates user input against this list.

When a user selects a specific type (e.g., "Ltd"), the system saves the account type as "Business (Ltd)" in the database. The system logic utilises keyword matching (specifically, looking for "Business") to ensure that all relevant business rules (such as fees and overdrafts) apply, regardless of the specific subtype suffix.

6. Annual Fees (Business Accounts)

Requirement: "Apply the annual fee of £120 automatically."

Assumption: The fee is charged immediately upon account creation. If the opening balance is below £120, the system permits the account to be created but places it into a negative balance right away. This approach ensures the bank maintains fee revenue and aligns with real banking practices, where account fees can result in an initial debit balance.

7. Data Persistence

Requirement: "Data should be stored in a persistent format... File-based storage (JSON, CSV, or serialised objects)."

Assumption: We used SQLite for persistence, fulfilling the requirement for persistent file storage while providing enhanced reliability, data integrity, and query capabilities compared to flat CSV or JSON text files. It also enables robust relationship management among Customers, Accounts, and Transactions.

8. Scope Of Transactions

Assumption: Per the brief's focus on "essential operations".

We have implemented Deposits, Withdrawals, interest applications, Direct Debits, and Standing Orders.

9. Scheduled Payments

Assumption: The Teller can manually trigger the "End-of-Day Processing" option. The SQL query selects only payments with a next payment date on or before today. Once a payment is successfully processed, the system promptly updates the next payment date to a future date (e.g., one month later). If a Teller accidentally runs the processing job twice in one day, the second run will find no eligible records, thus preventing any possibility of double-charging.

10. Logging

Assumptions: Logs can be recorded in local text files after the program has finished running. All significant events, such as customer creation, account opening, financial transactions, and

scheduled payment execution, are logged immediately as they occur. This approach ensures that an audit trail is maintained, even if the program terminates unexpectedly or crashes.