Banking System Requirement Specification

Group 15

YuHang Zhu

April 13 . 2024

1 Introduction

1.1 Purpose

The objective of the bank system project is to design and develop a comprehensive and efficient banking system that caters to the needs of both account holders and banking facilities. The system encompasses various features such as a database to store all account data, checking and saving accounts, an interface for a mobile application, and an interface specifically designed for ATMs.

1.2 Scope

The scope of the bank system project encompasses the following core functionalities:

1. Database Management: Implementation of a centralized database to store comprehensive account data, including information related to checking and saving accounts. This database will ensure efficient data management and retrieval for account-related transactions.

2. Account Management: Provision of features for managing checking and saving accounts, including the ability to open and close accounts as per customer preferences and requirements.

3. Mobile Application Interface: Development of a user-friendly interface accessible through a mobile application. This interface will enable customers to perform essential banking tasks such as transferring money to other individuals and managing account status.

4. ATM Interface: Integration of an intuitive interface specifically designed for Automated Teller Machines (ATMs). Through this interface, customers will be able to conveniently deposit and withdraw cash, as well as query their account details securely.

2 Use Case Diagram

As shown in Figure ??, the banking system allows users and admins to log in and out, perform fund transfers, track transfers, change passwords, and view reports. Additionally, admins can perform administrative tasks such as editing reports, adding accounts, and updating system settings.

3 Class Diagram

As shown in Figure **??**, subclasses APP and ATM under BankingSystem are responsible for implementing basic functions, the UI class handles interface presentation and input processing, and the Database class stores accounts and their associated transaction information.

4 Functional Boundaries and Limitations

4.1 ATM Functionality

1. **create_account@password**
    - Constraint1: Account IDs should consist of 10 digits.
    - Constraint2: No duplicate account IDs or usernames.
    - Constraint3: Password length and complexity (e.g., Way 1. minimum of 8 characters, including uppercase letters, lowercase letters, and numbers. Way 2. consist of exactly 6 digits).
2. **close_account**
    - Constraint1: Ensure the account balance is zero before closing the account.
    - Constraint2*: User identity verification (e.g., through password) should be performed before closing the account.
    - Constraint3: The account must not be logged in on the app.
3. **insert_card@id**
    - Constraint1: Validity and existence of the account IDs.
    - Constraint2*: Check if the card is blacklisted or frozen.
4. **return_card**
    - Constraint1*: Check for any unfinished transactions at APP before the card is returned.
    - Constraint2: Ensure the card has been inserted in ATM before returning.
5. **deposit_cash@num**
    - Constraint1: Minimum and maximum values for the deposit amount (e.g $0.01 $\sim$ $50000.00).
    - Constraint2*: Check if the cash capacity of the ATM is sufficient to store the new cash.
    - Constraint3*: Check for any unfinished transactions at APP before the card is returned.
    - Constraint4: Update the database.
6. **withdraw_cash@num@password**
    - Constraint1: Sufficient account balance for withdrawal.
    - Constraint2: Password verification.
    - Constraint3*: Availability of cash inventory in the ATM.
    - Constraint4*: Check for any unfinished transactions at APP before the card is returned.
    - Constraint5: Update the database.

4.2 APP Functionality

1. **log_in@id@password#app_id**
    - Constraint1: Correct format for account ID, password and app ID.
    - Constraint2: The app to log in is opened.
    - Constraint3.1*: If login fails, repeatedly request the correct ID and password.
    - Constraint3.2*: Limit on the number of login attempts (to prevent brute-force attacks), multi-factor authentication.
2. **log_out#app_id**
    - Constraint1: Check if the user is logged in.
    - Constraint2: Correct format for app ID.
    - Constraint3*: Ensure all unsaved data is saved before logout.
3. **close_app#app_id**
    - Constraint1: Correct format for app ID.
    - Constraint2: Check if the application is in use (e.g log out the using account).
    - Constraint3*: Ensure all unsaved data is saved before closing.

4.3 Both (Shared Functionality)

1. **change_password@new_password(#app_id)**
    - Constraint1: Correct format for the new password and app ID.
    - Constraint2: Avoid using the old password as the new password, multi-factor authentication.
    - Constraint3: The state of account should be logged in APP or inserted-card in ATM.
    - Constraint4: Update the database.
2. **transfer_money@receiver_id@num(#app_id)**
    - Constraint1: Minimum and maximum values for the transfer amount (e.g \$0.01 $\sim$ \$50000.00).
    - Constraint2: Valid state of account and correct format for the app ID and account ID.
    - Constraint3: Ensure sufficient account balance, validity of the receiver's account, prevent duplicate transfers.
    - Constraint4: Update the database.
3. **query(#app_id)**
    - Constraint1*: Limit on query frequency (to prevent excessive querying).
    - Constraint2: Valid state of account and correct format for the app ID and account ID.
    - Constraint3: Get the account's balance, transaction log and *password.

4.4 other

1. **open_app**
   - Constraint1: open the $i + 1_{th}$ APP, where the i is the num of APP opened
2. **reset**
   - Constraint1: Only need to delete all the data stored in database.

4.5 Failure Cases (Errors)

Consider the following tests when operations above fail:

- Provide clear error messages to help users understand the issue.
- *Ensure system consistency after the error occurs (e.g., transaction rollback).
- *Log all failed attempts for future analysis and improvement.

NOTE : (* represents the optimal functional limitation we could consider later.)

WeeklyReport/BankSysUseCase.png

Figure 1: Basic use case of banking system

WeeklyReport/class_diagram.png

Figure 2: Basic class diagram of banking system