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

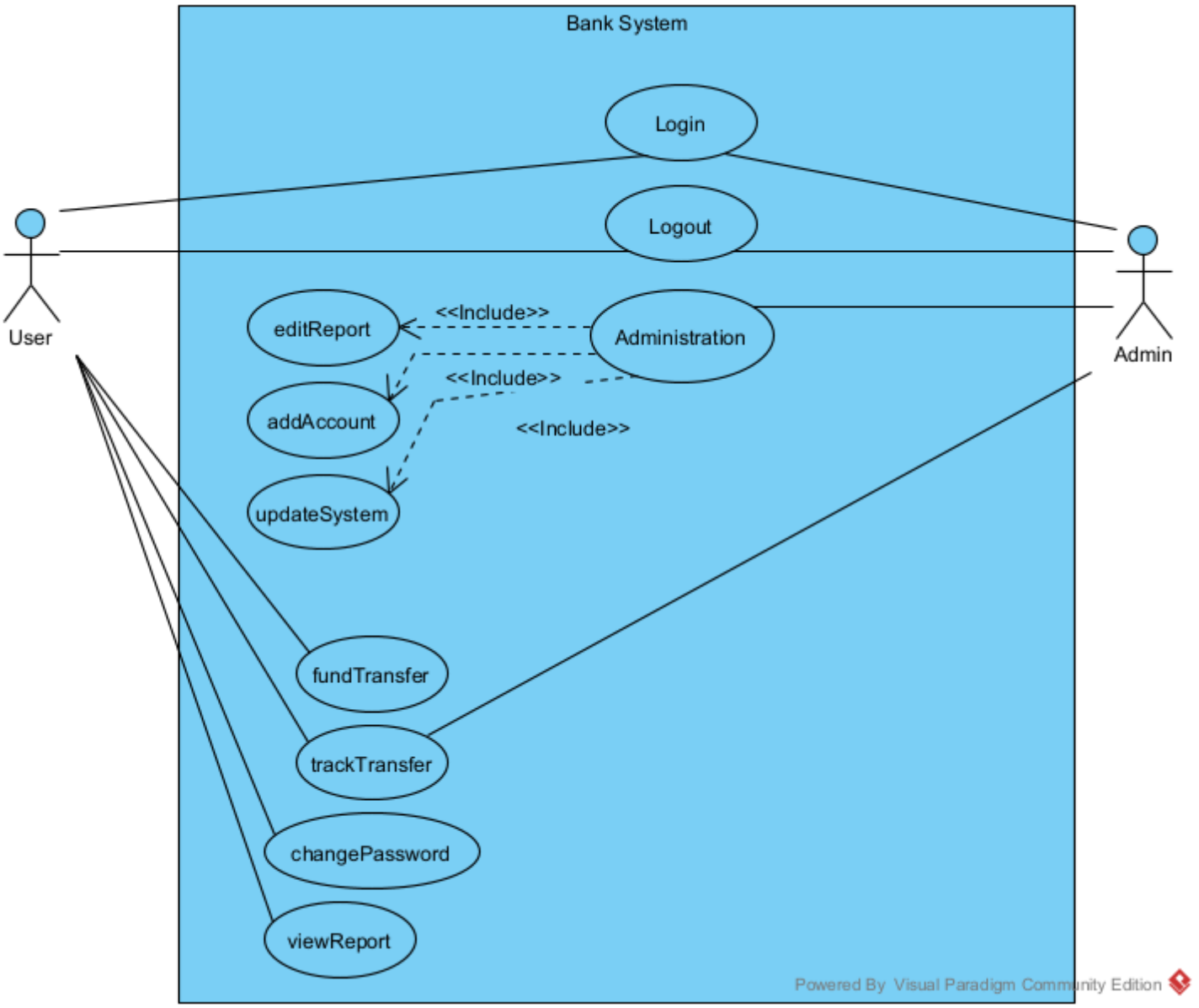

Figure 1: Basic use case of banking system

As shown in Figure 1, 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

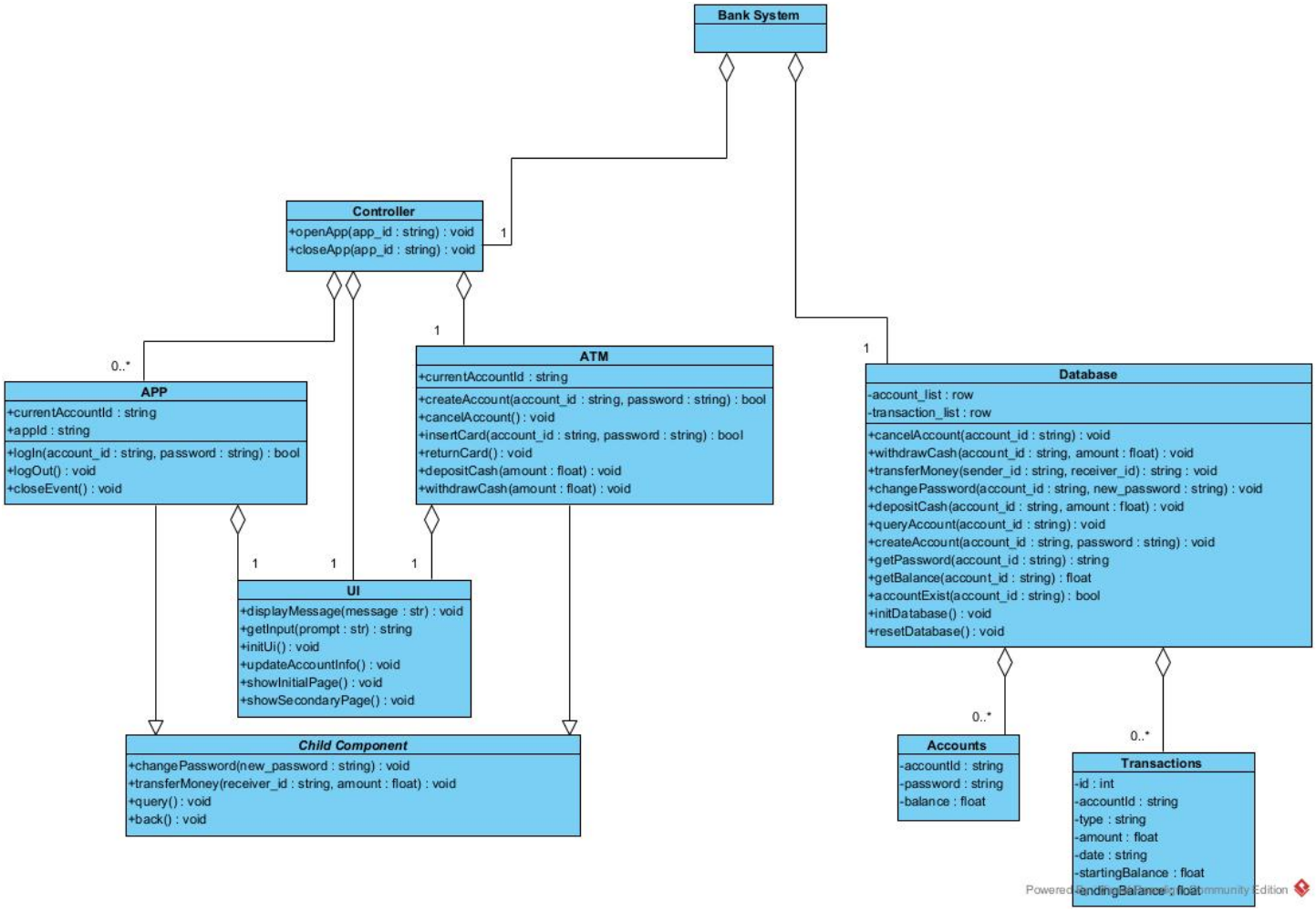

Figure 2: Basic class diagram of banking system

As shown in Figure 2, 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@account_id@password**
   - Constraint1: Account IDs should consist of 10 pure digits.
   - Constraint2: No duplicate account IDs or usernames.
   - Constraint3: Password length and complexity: minimum of 8 characters, including uppercase letters, lowercase letters, and numbers.
   - Constraint4: Sufficient processing time.
2. **cancel_account**
   - Constraint1: Ensure the account balance is zero before closing the account.
   - Constraint2: The account must not be logged in on the app.
   - Constraint3: Sufficient processing time.
3. **insert_card@account_id@password**
   - Constraint1: Validity and existence of the account IDs and correctness of password.
   - Constraint2: Sufficient processing time.
4. **return_card**
   - Constraint1: Ensure the card has been inserted in ATM before returning.
   - Constraint2: Check that no other operations are taking place in ATM.
   - Constraint3: Sufficient processing time.
5. **deposit_cash@amount**
   - Constraint1: Minimum and maximum values for the deposit amount (0.00 <= && <= 50000.00).
   - Constraint2: Check that no other operations are taking place in ATM.
   - Constraint3: Check for any unfinished transactions at APP before the card is returned.
   - Constraint4: Update the database.
   - Constraint5: Sufficient processing time.
6. **withdraw_cash@amount**
   - Constraint1: Minimum and maximum values for the withdraw amount (0.00 <= && <= 50000.00).
   - Constraint2: Sufficient account balance for withdrawal.
   - Constraint3: Check that no other operations are taking place in ATM.
   - Constraint4: Check for any unfinished transactions at APP before the card is returned.
   - Constraint5: Update the database.
   - Constraint6: Sufficient processing time.

4.2 APP Functionality

1. **log_in@account_id@password**
   - Constraint1: Validity and existence of the account IDs and correctness of password.
   - Constraint2: The app to log in is opened.
   - Constraint3: One account can only be logged into one APP.
   - Constraint5: Sufficient processing time.
2. **log_out**
   - Constraint1: Check if the user is logged in.
   - Constraint2: Check that no other operations are taking place in APP.
   - Constraint3: Sufficient processing time.

4.3 Both (Shared Functionality)

1. **change_password@new_password**
   - Constraint1: Correct format for the new password
   - 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.
   - Constraint5: In the same UI, make sure there are no other actions performed, and in different UI, make sure there are no other actions related to the transaction.
   - Constraint6: Sufficient processing time.
2. **transfer_money@receiver_id@amount)**
   - Constraint1: Minimum and maximum values for the transfer amount (0.00 <= && <= 50000.00).
   - Constraint2: Validity and existence of the receiver_id.
   - Constraint3: Ensure sufficient account balance, prevent duplicate transfers.
   - Constraint4: Update the database.
   - Constraint5: In the same UI, make sure there are no other actions performed, and in different UI, make sure there are no other actions related to the transaction.
   - Constraint6: Sufficient processing time.
   - Constraint7: Cannot transfer to your own.
3. **query**
   - Constraint1: The state of account should be logged in APP or inserted-card in ATM.

- Constraint2: Show the account's balance, transaction logs(including, Transaction Time, Transaction Type, Amount, Start Balance, End Balance) and password.
- Constraint3: Check that no other operations are taking place.
- Constraint4: Sufficient processing time.

4. **back**
    - Constraint1: Return to the initial page

4.4 Controller Functionality

1. **open_app@app_id**
    - Constraint1: The app_id must be a positive integer 1 <= && <= 99 and must not duplicate the app_id of an open APP.
    - Constraint2: open the $i + 1_{th}$ APP, where the i is the num of APP opened.
2. **close_app@app_id**
    - Constraint1: The app_id must be a positive integer 1 <= && <= 99 and the corresponding APP must already be open.
3. **reset**
    - Constraint1: Delete all the data stored in database.

4.5 Failure Cases (Errors)

Consider the following tests when operations above fail:

- Provide clear and correctly corresponding error messages to help users understand the issue.
- Ensure system consistency after the error occurs (e.g., transaction rollback).