BOTSWANA ACCOUNTANCY COLLEGE
The Botswana Business School

Student Id: cse24-102

Student names: Arabang Benville Mothofela

Student email: cse24-102@thuto.bac.ac.bw

Cohort: 2025

Assignment title: CSE201 – Database Design and Development

Date of submission: 12 Nov 2025

Programme of Study: COMPUTER SYSTEMS ENGINEERING

Year of Study: 2

Intellectual property statement by checking the box below, I certify that this assignment is my own work and is free from plagiarism. I understand that the assignment may be checked for plagiarism by electronic or other means and may be transferred and stored in a database for the purposes of data-matching to help detect plagiarism. The assignment has not previously been submitted for assessment in any other unit or to any other institution. I have read and understood the Botswana Accountancy College plagiarism guidelines policy.

☑ Agree Signature: A.Mothofela

Date: 17/11/2025

Integration Test Report – Bank Management System

1. Test Objective

The objective of this integration test is to verify that the Banking System functions correctly as a fully integrated architecture. The test ensures proper communication and data flow between the **JavaFX GUI** (LoginView.fxml, CustomerDashboard.fxml, DepositWithdrawView.fxml), the **controller layer** (LoginController, DepositWithdrawController), the **core domain model** (Customer, ChequeAccount, BankData, SavingsAccount, InvestmentAccount, Interest, Withdraw), and the **MySQL JDBC persistence layer** implemented through DatabaseStorage, the DAO classes (CustomerDAO, AccountDAO, CustomerCredentialsDAO) and the pooled connections managed by DatabaseConfig (HikariCP).

This coverage explicitly validates the business rules baked into the model: SavingsAccount implements only the Interest interface and therefore cannot withdraw, ChequeAccount implements Withdraw for transactional access, and InvestmentAccount implements both Interest and Withdraw while enforcing the minimum opening balance of **BWP 500** in its constructor.

2. Test Scenario (End-to-End User Flow)

A **customer** logs in through the JavaFX login interface, performs a **deposit** on one of their Cheque accounts, and the updated balance is persisted to MySQL through the JDBC DAO layer. The customer-facing flow is isolated from the employee dashboard path (BankData.isEmployee()), ensuring the correct role-based view is presented.

Layers involved in this test:

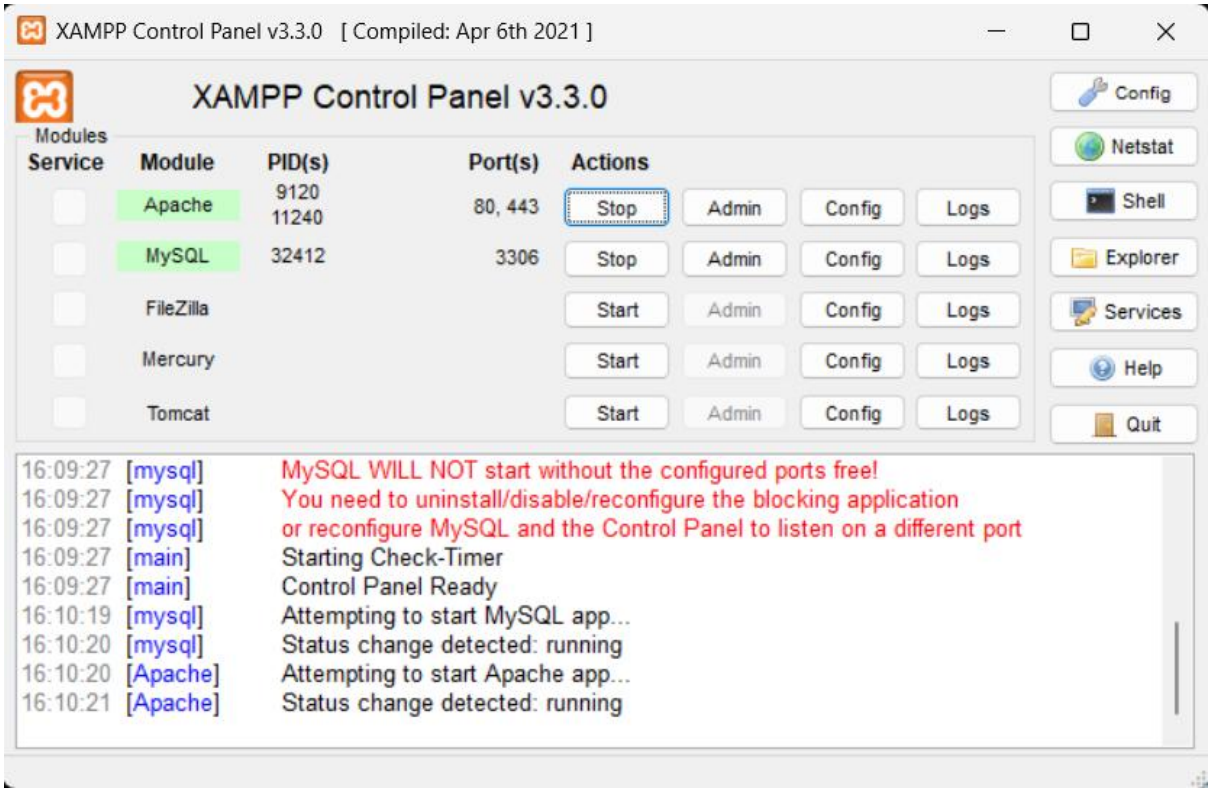
- **Boundary (GUI):** LoginView.fxml, CustomerDashboard.fxml, DepositWithdrawView.fxml
- **Controller:** LoginController, CustomerDashboardController, DepositWithdrawController
- **Domain Model:** Customer, CustomerCredentials, ChequeAccount (implements Withdraw), SavingsAccount (implements Interest only), InvestmentAccount (implements both interfaces and enforces the BWP 500 minimum), BankData, AuditLogger
- **Persistence:** DatabaseStorage, CustomerDAO, AccountDAO, CustomerCredentialsDAO, DatabaseConfig (MySQL + JDBC via HikariCP)

3. Pre-Conditions

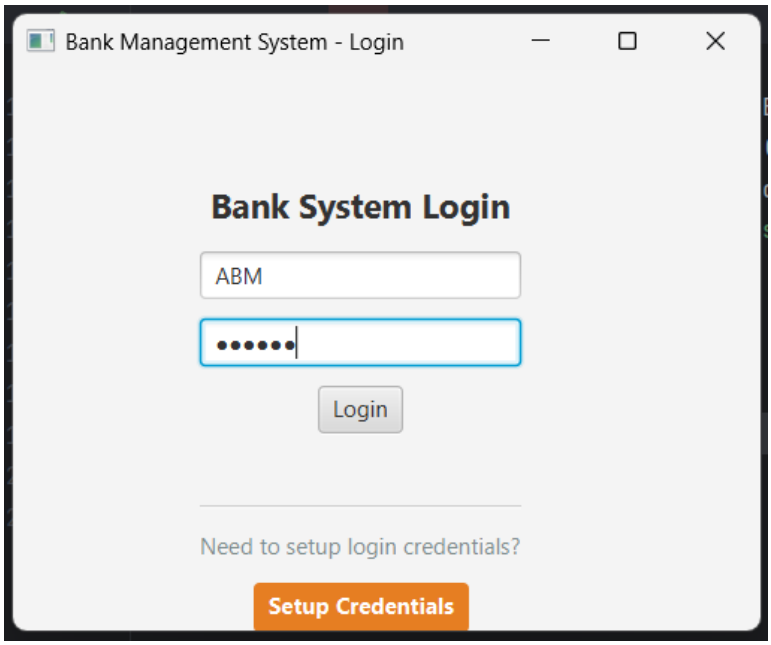
- MySQL server is running and accessible at:
  jdbc:mysql://localhost:3306/bank_management
- Schema and tables have been created.
- BankData.loadDataFromDatabase() has loaded all customer and account records.
- **Test user:**
  - Username: ABM
  - Password: 123123
- **Test account:**
  - Account Number: ACC8O2379
  - Account Type: *InvestmentAccount* (seeded with at least **BWP 500**)
  - Initial Balance: **BWP 551.25**
- System also contains:
  - A SavingsAccount (no withdrawals allowed)
  - Any ChequeAccount (employment details needed)

4. Test Steps

1. Launch the JavaFX application and Xampp.

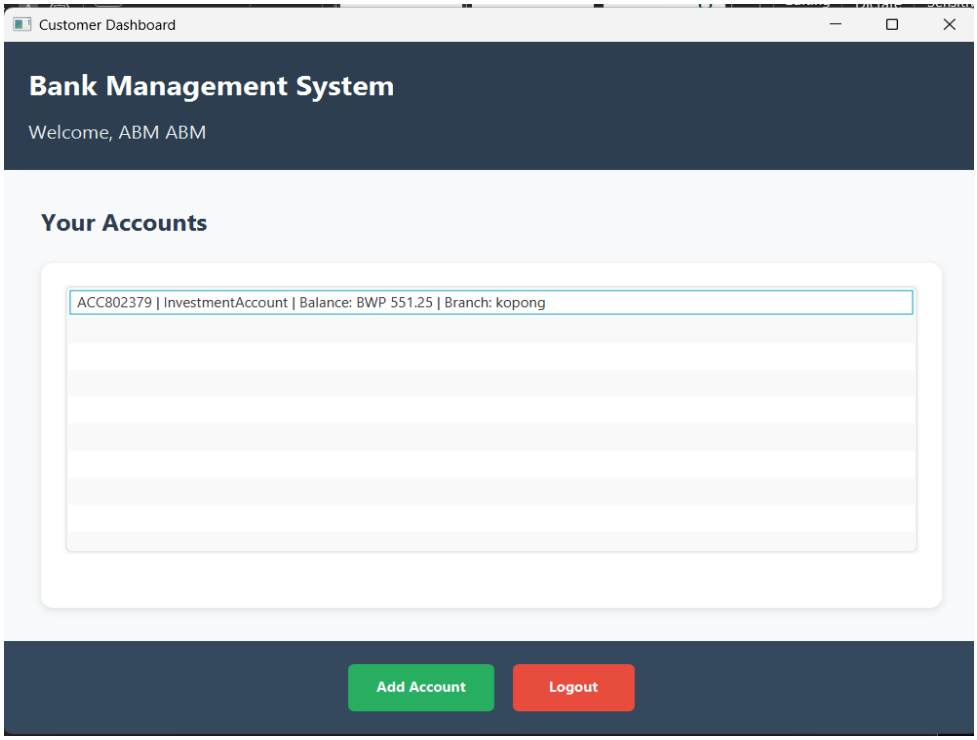


2. Enter the customer's login credentials.

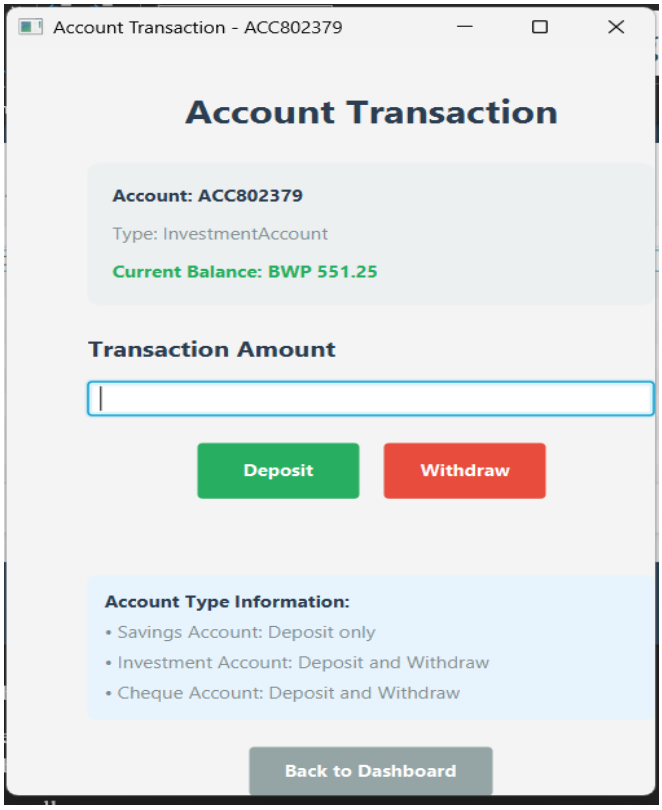


LoginController.handleLogin() validates the user through CustomerCredentialsDAO.

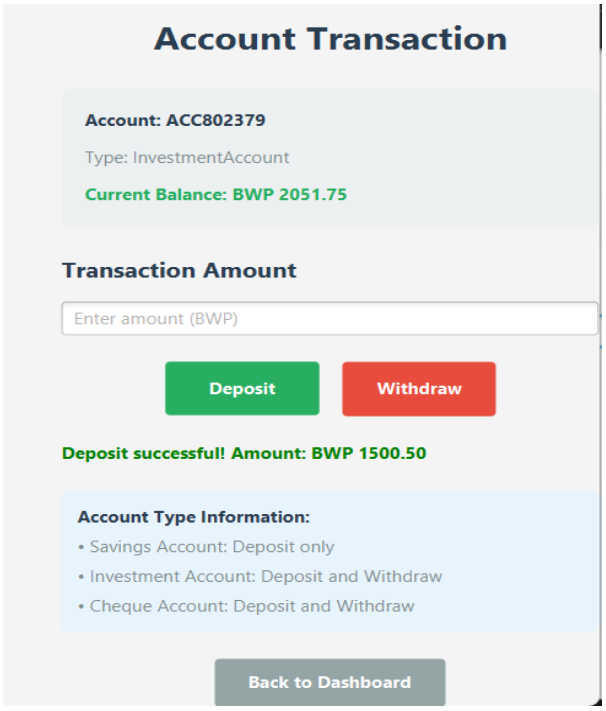
Successful login loads the customer dashboard:



3. From the dashboard, open the **Deposit/Withdraw** window.

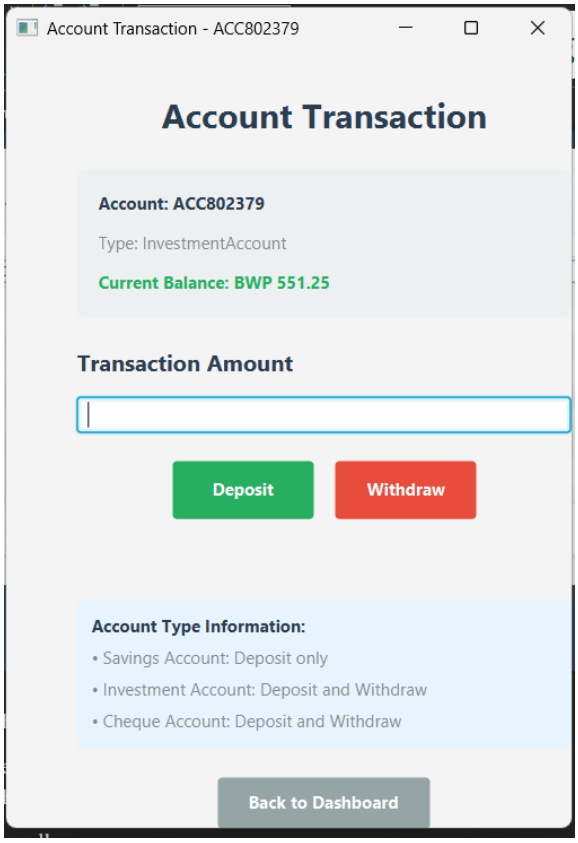


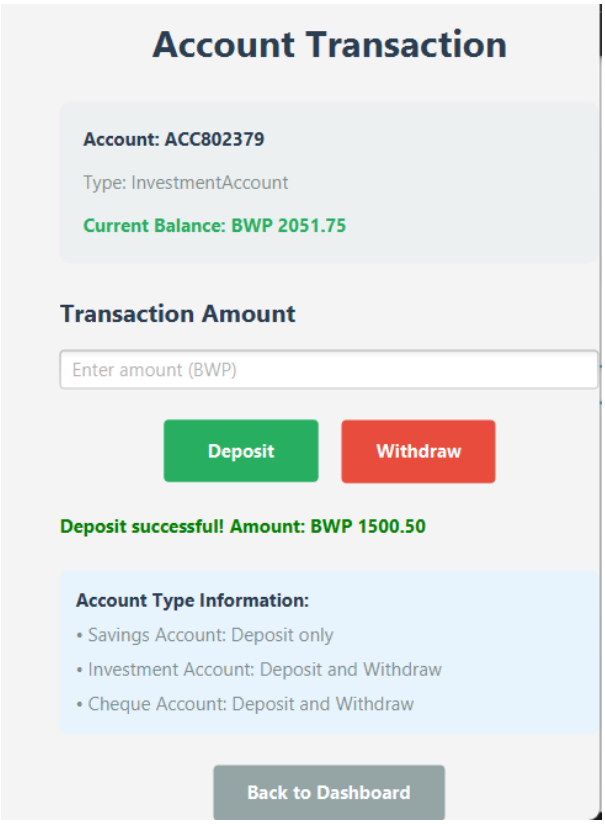
4. Enter a deposit amount of **BWP 1 500.50** for account ACC8O2379.



5. Controller calls:
    - ChequeAccount.deposit(amount)
    - Updates in-memory balance
6. Controller triggers persistence via:
   AccountDAO.saveAccount() → MySQL UPDATE
7. Audit entry recorded using:
   AuditLogger.log(...)

8. GUI refreshes and displays updated account balance.
   From this to this



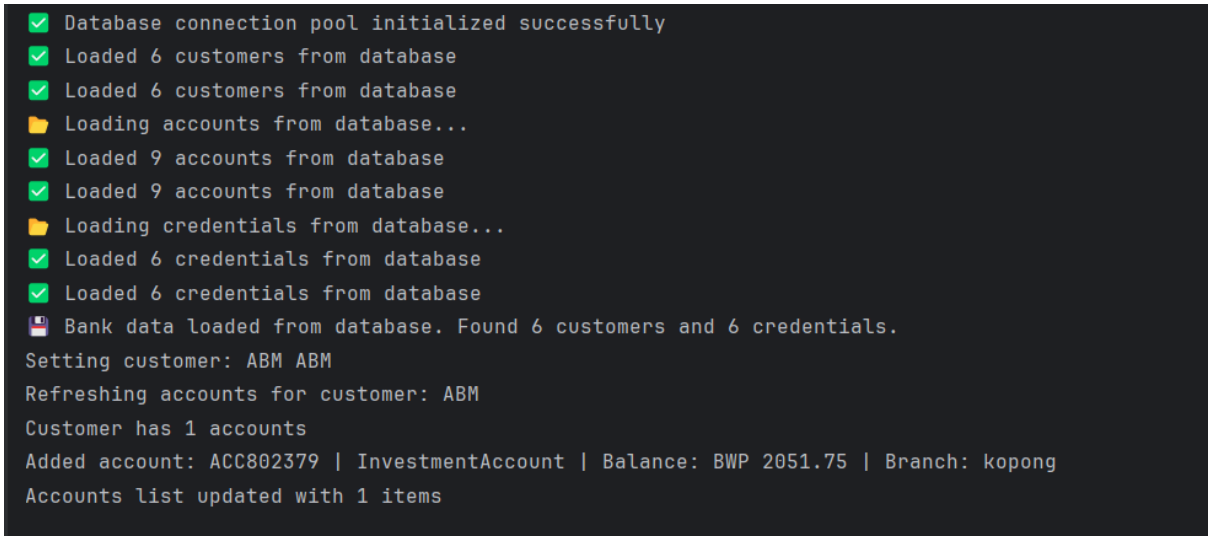


5. Expected vs Actual Results

Expected Result	Actual Result	Status
Login validated through customer path	Customer dashboard loaded	✓ PASS
Dashboard receives the loaded Customer object	Controller injected the hydrated customer	✓ PASS
Deposit validation correct	Accepted amount 1500.50	✓ PASS
Savings account withdraw disabled	Confirmed (no Withdraw interface)	✓ PASS
Balance updated in domain model	new balance = 2051.75	✓ PASS
Balance persisted in MySQL	UPSERT executed via JDBC	✓ PASS
Audit log entry created	Audit entry successfully recorded	✓ PASS
GUI updated	New balance displayed	✓ PASS

6. Evidence

Console Output



Audit Log Entry

2025-11-19T16:22:44.482|category=transaction|actor=C005|subject=ACC802379|action=deposit|success=true|details=amount=1500.5

7. Conclusion

The integration test confirms that the **JavaFX GUI**, **controller layer**, **domain model**, and the **MySQL JDBC persistence layer** operate correctly as a unified system. The end-to-end flow customer login, deposit processing, model update, DAO persistence, and audit logging executed successfully.

Final Result: the system PASS