Banking System - Project Documentation

1. Project Overview

Project Name: Banking System
Course: Object-Oriented Programming 1 (Second Semester)
Development Environment: Linux (Fedora), Visual Studio Code
Language: Java 21
GUI Framework: JavaFX 21
Database: MySQL 8.0

Purpose: A desktop banking application demonstrating OOP concepts with separate interfaces for administrators and customers. Implements secure authentication, transaction management, loan processing, and account operations.

---

2. Tools & Technologies Used

Development Tools

- **IDE:** Visual Studio Code with Java Extension Pack
- **JDK:** OpenJDK 21.0.8
- **Database:** MySQL Server 8.0
- **Version Control:** Git

Libraries & Dependencies

1. **JavaFX SDK 21.0.9** - GUI framework for desktop interface
   - Location: `/home/uthman/Downloads/openjfx-21.0.9_linux-x64_bin-sdk/javafx-sdk-21.0.9`
2. **MySQL Connector/J 8.0.33** - JDBC driver for database connectivity
   - File: `lib/mysql-connector-j-8.0.33.jar`
3. **jBCrypt 0.4** - Password hashing library
   - File: `lib/jbcrypt-0.4.jar`

---

3. Project Structure

```
BankingSystem/
├── .vscode/                # VS Code configuration
│   ├── launch.json         # Run configurations
│   └── settings.json       # Project settings
├── lib/                    # External JAR files
│   ├── jbcrypt-0.4.jar
│   └── mysql-connector-j-8.0.33.jar
├── src/                    # Source code
│   ├── models/             # Data models (User, Customer, Account, etc.)
│   ├── database/           # DAO classes for database operations
│   ├── views/              # JavaFX UI screens
│   ├── utils/              # Helper classes (PasswordHasher, Validator, SessionManager)
│   └── main/application/   # Main entry point
└── bin/                    # Compiled classes (auto-generated)
```

Package Breakdown

models/ - Contains all entity classes representing database tables

- User (abstract), Admin, Customer
- Account (abstract), SavingsAccount, CurrentAccount
- Transaction, Loan, Overdraft

database/ - Data Access Objects (DAO) for database operations

- DatabaseConnection, UserDAO, CustomerDAO, AccountDAO, TransactionDAO, LoanDAO, OverdraftDAO

views/ - JavaFX UI components

- LoginView, AdminDashboard, CustomerDashboard

utils/ - Cross-cutting utilities

- PasswordHasher (BCrypt), SessionManager (Singleton), Validator

---

4. Database Design

Tables Overview

users - Authentication (username, hashed password, role)
customers - Personal information (name, email, phone, address, DOB)
accounts - Bank accounts (account number, type, balance, status)
transactions - Transaction history (type, amount, balance_after, date)
loans - Loan applications (amount, interest rate, duration, status)
overdrafts - Overdraft requests (limit, status)

Relationships

- One user → one customer
- One customer → many accounts
- One account → many transactions

- One account → many overdrafts
- One customer → many loans

Key Constraints

- Usernames must be unique
- Account numbers must be unique
- Savings accounts: minimum $100 balance
- Overdrafts: only for current accounts, max $5,000
- Passwords stored as BCrypt hashes (255 chars)

---

5. OOP Concepts Implemented

Inheritance

- **User** (abstract) → **Admin**, **Customer**
- **Account** (abstract) → **SavingsAccount**, **CurrentAccount**

Encapsulation

- All fields are `private`
- Access through `public` getters/setters
- Immutable fields marked as `final`

Abstraction

- Abstract classes: User, Account
- Abstract methods: login(), logout(), deposit(), withdraw()

Polymorphism

- Method overriding: Admin and Customer implement login() differently
- Different withdrawal logic for savings vs current accounts

---

6. Key Features

Admin Features

1. **Create Customer** - Creates user account, customer record, and initial bank account
2. **Add Account to Customer** - Adds additional accounts (savings/current) to existing customers
3. **View All Customers** - Lists all customers with their details
4. **Approve Loans** - Review and approve/decline pending loan applications
5. **Approve Overdrafts** - Review and approve/decline overdraft requests
6. **Change Password** - Update admin password with security verification

Customer Features

1. **View Balance** - Display all accounts with current balances
2. **Deposit Money** - Add funds to account with transaction recording
3. **Withdraw Money** - Withdraw funds with balance and minimum balance checks
4. **Apply for Loan** - Submit loan application with amount and duration
5. **Request Overdraft** - Request overdraft protection (current accounts only)
6. **View Statement** - See transaction history (requires password re-entry)
7. **Terms & Conditions** - View banking terms

8. **Change Password** - Update password with security verification

---

7. Security Features

Password Security

- **BCrypt Hashing:** All passwords hashed before storage (not plain text)
- **Password Strength:** Minimum 8 characters required
- **Re-authentication:** Viewing statements requires password re-entry
- **Session Management:** Tracks logged-in user, automatic logout

Data Validation

- Email format validation
- Phone number format validation
- Amount validation (positive numbers only)
- Username validation (alphanumeric, 3-20 chars)

SQL Injection Prevention

- All database queries use PreparedStatement
- User inputs sanitized and parameterized

---

8. Design Patterns Used

DAO (Data Access Object) Pattern

Separates database operations from business logic. Each entity has its own DAO class:

- UserDAO, CustomerDAO, AccountDAO, TransactionDAO, LoanDAO, OverdraftDAO

Singleton Pattern

SessionManager - Ensures only one instance manages user session across the application

MVC-Inspired Architecture

- **Models:** Data classes (User, Account, etc.)
- **Views:** JavaFX UI classes (LoginView, AdminDashboard, CustomerDashboard)
- **Controllers:** DAO classes handle business logic and data operations

---

9. How the Application Works

Application Flow

1. Startup

- Main.java launches JavaFX application
- LoginView displayed

2. Login

- User enters username and password
- UserDAO.validateLogin() checks credentials (BCrypt comparison)
- UserDAO.getUserRole() determines if admin or customer
- SessionManager stores user info
- Redirect to appropriate dashboard

3. Admin Operations

- Create customer → Creates user, customer record, and account
- Approve loans → Updates loan status in database
- View customers → Queries database and displays list

4. Customer Operations

- Deposit → Updates account balance + records transaction
- Withdraw → Checks balance, updates account + records transaction
- View statement → Requires password, shows transaction history

5. Logout

- Clears session
- Returns to login screen

---

10. Business Rules Implemented

Account Rules

- **Savings Account:**
  - Minimum balance: $100
  - Can apply for loans
  - Cannot have overdraft
- **Current Account:**
  - No minimum balance
  - Can request overdraft (max $5,000)
  - Suitable for frequent transactions

Transaction Rules

- Deposits: Any positive amount
- Withdrawals: Must have sufficient balance
- All transactions recorded with timestamp and balance_after

Loan Rules

- Interest rate: 5.5% (set by bank, not customer)
- Status: Pending → Approved/Declined (by admin)
- Recorded with application date and processed date

Overdraft Rules

- Only for current accounts
- Maximum limit: $5,000
- Requires admin approval

---

11. Database Connection

Configuration:

- URL: `jdbc:mysql://localhost:3306/banking_system`
- Driver: `com.mysql.cj.jdbc.Driver`
- Connection managed by DatabaseConnection class
- Uses PreparedStatement for all queries

Example:


java

```java
Connection conn = DatabaseConnection.getConnection();
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setString(1, username);
ResultSet rs = stmt.executeQuery();
```

---

12. Installation & Setup

Prerequisites

1. Install JDK 21
2. Install MySQL Server
3. Install VS Code with Java Extension Pack
4. Download JavaFX SDK 21
5. Download MySQL Connector and jBCrypt JARs

Database Setup


sql

```sql
CREATE DATABASE banking_system;
USE banking_system;
-- Run all table creation scripts
-- Create default admin user
INSERT INTO users (username, password, role)
VALUES ('admin', '$2a$10$...', 'admin');
```

VS Code Configuration

- Configure `settings.json` with JavaFX path
- Configure `launch.json` with VM arguments
- Add JAR files to lib folder

Running the Application

- Press F5 in VS Code
- Login with admin credentials: admin / admin123

---

13. Testing Scenarios

Test Case 1: Create Customer

1. Login as admin
2. Click "Create Customer"
3. Fill form with test data
4. Verify customer created in database
5. Login as new customer

Test Case 2: Deposit & Withdraw

1. Login as customer
2. View initial balance
3. Deposit $500
4. Verify balance updated
5. Withdraw $200
6. Check transaction history

Test Case 3: Loan Application

1. Customer applies for loan
2. Logout
3. Login as admin
4. Approve/decline loan
5. Login as customer
6. Verify loan status

---

14. Known Limitations

1. **No password recovery** - Users cannot reset forgotten passwords
2. **No interest calculation** - Savings accounts don't automatically calculate interest
3. **No email notifications** - No automated emails for approvals/transactions
4. **Single admin account** - Only one admin can exist
5. **No account statements export** - Cannot export statements to PDF
6. **No transaction disputes** - No mechanism to dispute transactions
7. **Limited reporting** - No advanced analytics or reports

---

15. Future Enhancements

Possible Improvements

1. **Interest Calculation** - Automatic interest for savings accounts
2. **Fixed Deposits** - Time-locked accounts with higher interest
3. **Account Transfers** - Transfer money between accounts
4. **Email Notifications** - Alert users of transactions
5. **Two-Factor Authentication** - Enhanced security
6. **Mobile App** - Develop mobile version

7. **Advanced Reports** - Generate PDF statements and analytics
8. **Multiple Admins** - Support for multiple admin accounts
9. **Account Freeze** - Admin can freeze suspicious accounts
10. **Transaction Limits** - Daily/weekly withdrawal limits

---

16. Conclusion

This Banking System successfully demonstrates:

- **OOP principles** through inheritance, encapsulation, and polymorphism
- **Database integration** using JDBC and MySQL
- **GUI development** with JavaFX
- **Security practices** including password hashing and session management
- **Software design patterns** (DAO, Singleton, MVC-inspired)

The application provides a solid foundation for understanding enterprise-level application development while maintaining simplicity suitable for academic purposes.

Developer: Uthman
Project Completion Date: November 2025
Total Development Time: ~15 hours
Lines of Code: ~2,500+
Database Tables: 6
Java Classes: 20+