JAVA BANKING APPLICATION – COMPLETE PROJECT MANUAL

1. Introduction

This Banking Application is a console-based mini project developed using Core Java. It demonstrates Object-Oriented Programming (OOP), exception handling, secure password hashing, file-based persistence, and modular system design. The application allows users to register, login, deposit money, withdraw money, check balance, and view transaction history.

2. System Objectives

- Provide a simple banking simulation using Java
- Implement secure authentication using SHA-256
- Use serialization for data storage
- Demonstrate OOP concepts in a real-world scenario
- Provide clean modular code for learning

3. Technologies Used

- Java (Core Java)
- OOP Concepts
- SHA-256 Password Hashing
- Java Serialization
- HashMap, ArrayList
- VS Code / IntelliJ / Eclipse

4. System Architecture

The system is divided into three layers:

1. Presentation Layer – Console-based user interface
2. Business Logic Layer – Banking operations
3. Data Layer – users.db storage using serialization

5. Modules of the System

- User Registration
- User Login
- Deposit Module

- Withdrawal Module
- Balance Inquiry
- Transaction History

6. OOP Concepts Used

- Encapsulation – private attributes with getters/setters
- Abstraction – hiding complex logic inside classes
- Modularity – separate classes for User, BankSystem, Utils
- Constructors – initializing user objects

7. Data Storage (Serialization)

Serialization converts Java objects into byte streams and stores them in users.db. On program start, the file is loaded back into objects using ObjectInputStream.

8. Security Implementation

Passwords are hashed using SHA-256. This ensures passwords cannot be reversed, providing secure authentication.

9. Working of the System

1. User starts application
2. Chooses Register or Login
3. After successful login User Menu opens
4. User performs transactions
5. All operations saved in data file
6. User logs out

10. UML Diagrams (Concept Description)

- Use Case Diagram – Shows user and system interactions
- Class Diagram – User, BankSystem, Utils relationships
- Activity Diagram – Process flow from login to operations
- DFD Level 0 – High-level system flow

11. Test Cases

TC1: Register new user ® Successful

TC2: Register existing user ® Error

TC3: Login with wrong password ® Denied

TC4: Withdraw amount > balance ® Error

TC5: Deposit valid amount ® Updated balance

TC6: View transactions ® Full list displayed

12. Advantages

- Lightweight and simple
- Secure password handling
- Persistent data storage
- Modular and extendable design

13. Limitations

- No graphical interface
- Not suitable for multi-user concurrency
- File-based storage not scalable

14. Future Enhancements

- Integration with MySQL database
- GUI using JavaFX/Swing
- Admin panel
- OTP verification
- Online banking features

15. Viva Questions and Answers

Q1. What is Serialization?

A: Converting objects into byte streams for storage.

Q2. Why use SHA-256?

A: It is a non-reversible secure hashing algorithm.

Q3. What is OOP?

A: Object-Oriented Programming: Abstraction, Encapsulation, Inheritance, Polymorphism

Q4. Why HashMap?

A: Fast lookup of user accounts using key-value mapping.

16. Conclusion

The Java Banking Application successfully demonstrates console-based system development using Core Java principles. It is secure, scalable, and an excellent foundation for future upgrades like GUI or database integration.

17. Developer Details

Name: Atharv Ramesh Kulkarni

GitHub: github.com/kulatharv

College: Sinhgad Institute of Technology,Lonavala