OOPs Capstone Project Report
Spring 2025

Banking Management System

Name	Sap ID	Cohort
Bhavya Bhardwaj	500120630	B10
Abhinav Sharma	500123588	B9
Kshitij Amrit	500120238	B10
Nivedit Joshi	500122784	B9
Divyanshi Rawat	500119363	B10
Priyanshu Sarma	500126599	B10
Suhana	500122035	B10

UPES
UNIVERSITY OF TOMORROW

Index:-

1.Introduction

Banking Management System in Java

This project is a straightforward yet powerful Banking Management System, built using Java and designed with Swing for an intuitive Graphical User Interface (GUI). It leverages Object Serialization for efficient file handling, ensuring that user data remains safe and persistent.

What Can Users Do?

With this system, users can:

- **Sign up** by creating an account with key details like name, account number, password, account type (Savings or Current), and an initial deposit.
- **Log in securely** using their account number and password.
- **Manage transactions**, including deposits, withdrawals, and checking their account balance.
- **Apply for loans**, such as House Loans, Car Loans, and Education Loans, with predefined interest rates.
- **Repay loans** in full or in part, depending on their available balance.
- **Track financial activity** through a detailed transaction history that records every deposit, withdrawal, loan request, and repayment.

How Does It Keep Data Safe?

- User details and transaction records are stored using **serialization**, ensuring data persistence even after closing the application.
- A **log file (bankingdetails.txt)** is maintained to record important actions like registrations, logins, transactions, and logouts.

What Makes the Experience Smooth?

- A **colourful gradient background**, created with custom Java 2D graphics, adds a visual appeal.
- **Robust error handling and validation** keep the system running smoothly, even when users enter incorrect information.
- **Seamless UI transitions** make navigation effortless between login, registration, and dashboard interfaces.

2.Objectives

This Banking Management System aims to:

Develop a user-friendly banking application with an intuitive graphical interface.

Enable essential banking operations such as deposits, withdrawals, balance inquiries, and transaction history tracking.

Provide loan management features, allowing users to request and repay various types of loans.

Ensure secure data storage through file-based persistence using serialization, keeping user details intact even after closing the application.

Apply key programming concepts, including object-oriented principles, exception handling, and event-driven programming, to create a robust and efficient system.

3.Java Technologies Used

Java Swing (for GUI design)

- Java Swing was used to design the complete graphical user interface (GUI) of the application.
- Components like JFrame, JPanel, JButton, JLabel, JTextField, JPasswordField, JComboBox, JScrollPane, and JOptionPane were utilized to create a responsive and user-friendly banking interface.
- Layout managers like GridLayout were used for structured arrangement of components.

Java IO (FileOutputStream, FileInputStream, ObjectOutputStream, ObjectInputStream, BufferedWriter, FileWriter) for file handling

- Java I/O classes were employed to handle reading from and writing to files.
- **ObjectOutputStream** and **ObjectInputStream** were used for saving and loading the list of users (ArrayList<User>) through **serialization**.
- **BufferedWriter** and **FileWriter** were used to maintain a transaction and activity log in a text file (bankingdetails.txt).

Serialization (saving and loading User objects)

- The User class implements Serializable to allow the object's state (user details, balances, transaction history) to be saved to a file and retrieved later.
- This enables **persistent storage** without needing a database.

Collections Framework (ArrayList) to manage user accounts

- ArrayList<User> was used to dynamically store and manage multiple user accounts in memory.
- This collection simplifies tasks like searching, adding, and updating user records.

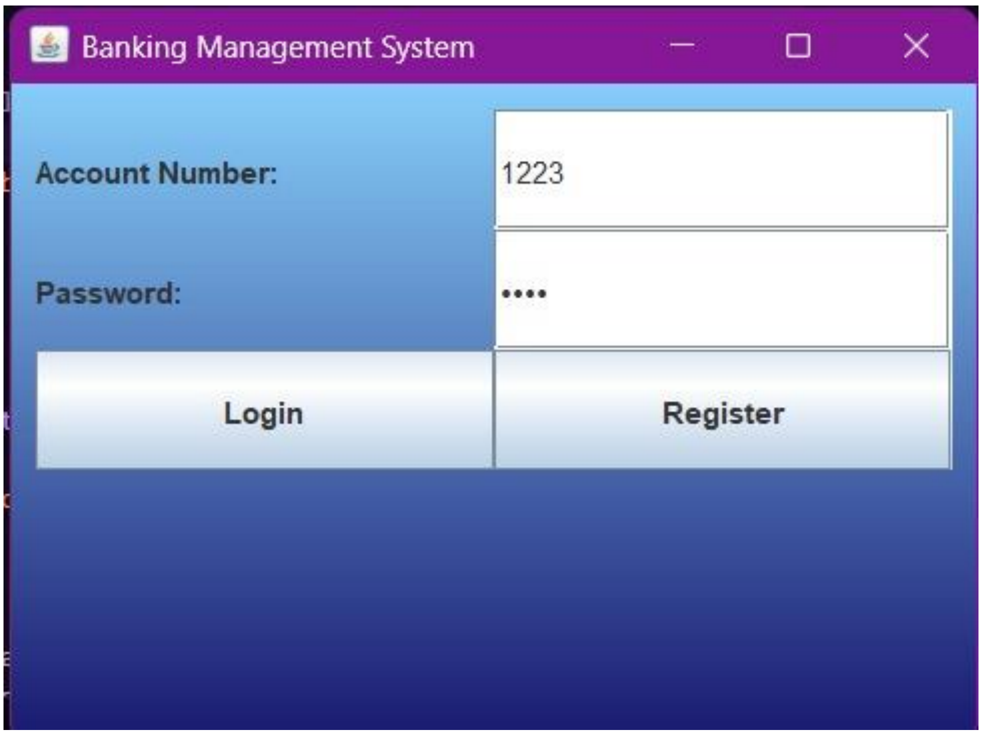
Exception Handling (try-catch blocks)

- Robust **error handling** was implemented using try-catch blocks to manage runtime exceptions like NumberFormatException, IOException, and ClassNotFoundException.
- This ensures smooth user experience even if incorrect input or file errors occur.

4.Project Demonstration

Login/Register

- New users can **register** by providing details such as their name, account number, initial deposit, password, and account type (Savings or Current).
- Existing users can **log in** securely by entering their registered account number and password.
- The system validates credentials and ensures that account numbers are unique during registration.

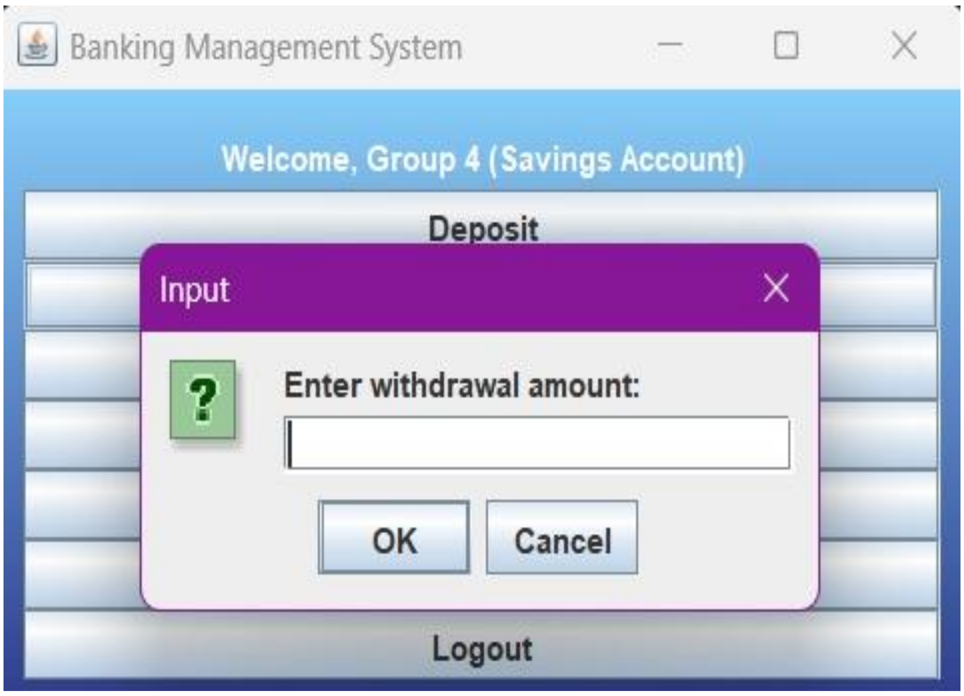


Account Number: 1223

Password: ••••

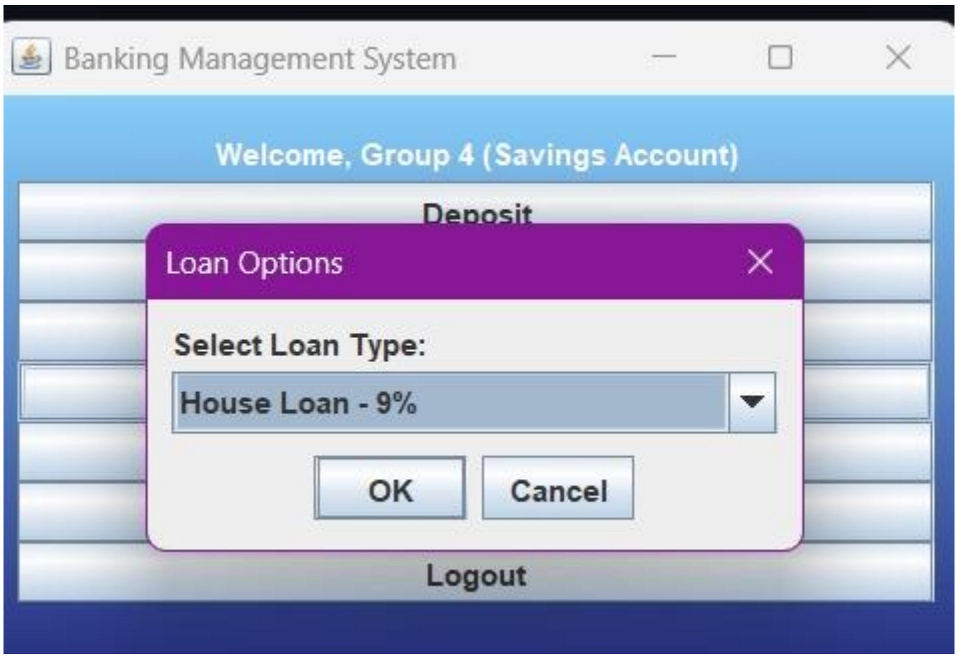
Deposit/Withdraw

- After logging in, users can **deposit** money into their account to increase their balance or **withdraw** money if sufficient balance is available.
- Proper validations are in place to prevent invalid transactions, and every action is recorded in the user’s transaction history.



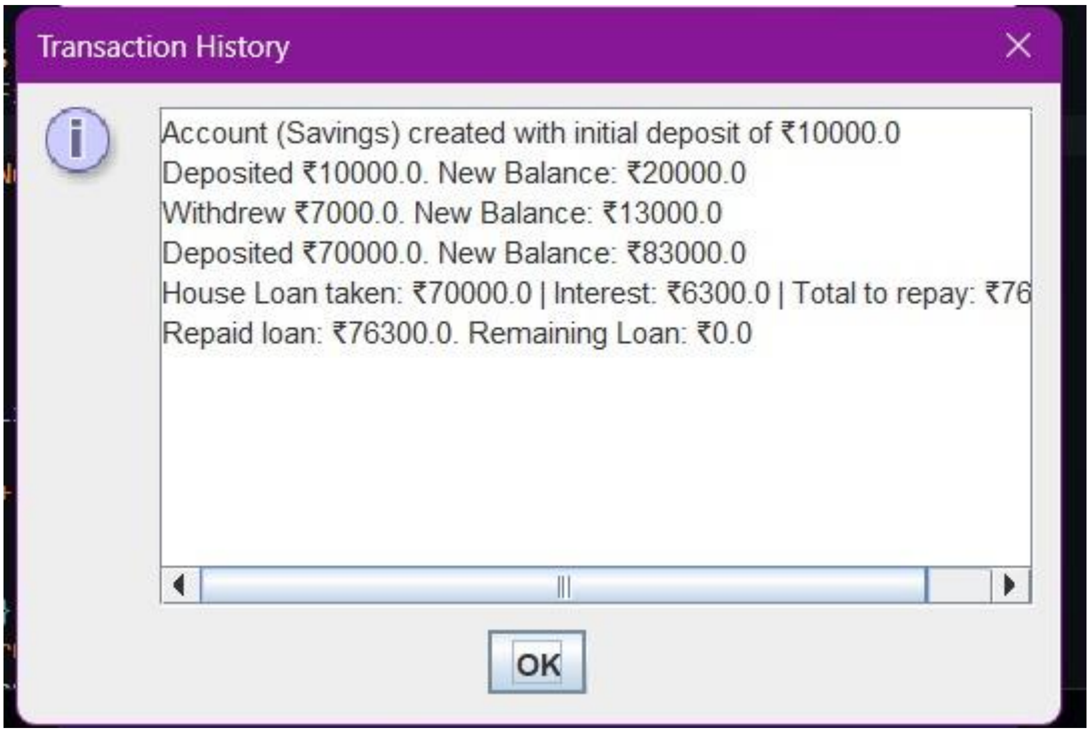
Loan Management

- Users can **apply for different types of loans**:
  - **House Loan** (9% interest)
  - **Car Loan** (9.5% interest)
  - **Education Loan** (8% interest)
- Upon taking a loan, the loan amount (plus calculated interest) is added to the user's outstanding loan balance, and the principal amount is credited to their account.
- Users can later **repay** their loans partially or completely.



Transaction History

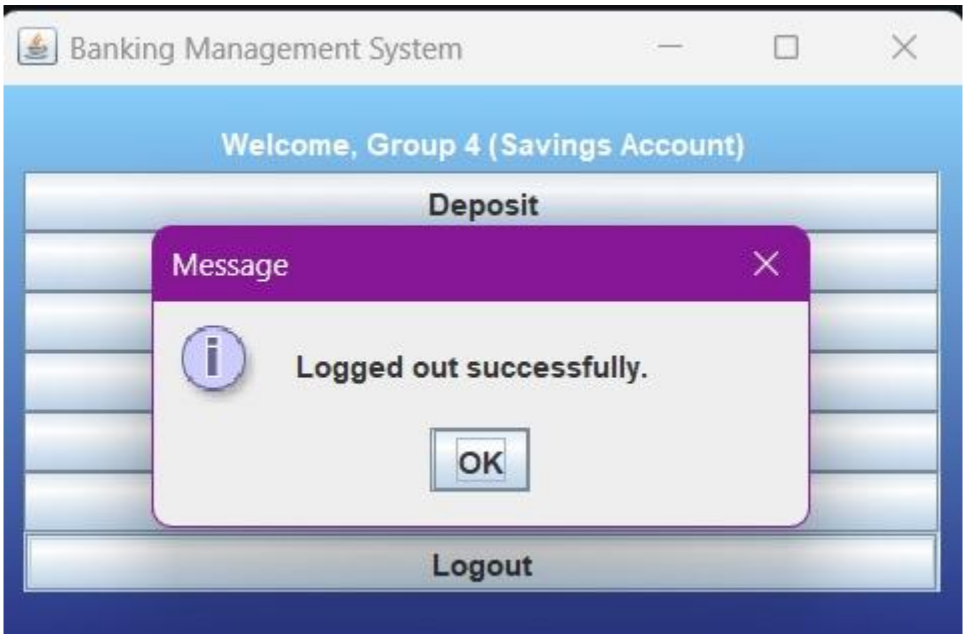
- Users can view a detailed **transaction history** showing all deposits, withdrawals, loans taken, repayments, and account activities.
- The history is displayed in a user-friendly, scrollable format.



```
Account (Savings) created with initial deposit of ₹10000.0
Deposited ₹10000.0. New Balance: ₹20000.0
Withdrew ₹7000.0. New Balance: ₹13000.0
Deposited ₹70000.0. New Balance: ₹83000.0
House Loan taken: ₹70000.0 | Interest: ₹6300.0 | Total to repay: ₹76
Repaid loan: ₹76300.0. Remaining Loan: ₹0.0
```

Logout and Data Saving

- When a user logs out or completes any important transaction, all data (account details, balances, transaction history) is **automatically saved** using object serialization.
- This ensures that user information is preserved between sessions, maintaining data integrity and security.



Logged out successfully.

5.Table of Capstone Project Topics Used

Table of Capstone Project Topics

Sr No	**Topics**	**Implementation Status**
1.	Java Swing GUI Development	✓
2.	Event Handling	✓
3.	File Handling using Streams	✓
4.	Object Serialization/Deserialization	✓
5.	Exception Handling	✓
6.	Collections (ArrayList)	✓
7.	Multithreading	✗
8.	Networking (Sockets)	✗
9.	JDBC (Database Connectivity)	✗
10.	Advanced Graphics/Animation	✓ (basic gradient background)