

# **BANK MANAGEMENT SYSTEM**

## **A PROJECT REPORT**

*Submitted by*

**ARUNKUMAR A**



*In partial fulfillment for the award of the course*

**Of**

**FULL STACK JAVA DEVELOPER**

**IN**

**SOFTWARE DEVELOPER**

**NEXT GEN,**

**CUDDALORE – 607001**

**MARCH 2025**



This is to certify that the project report entitled "**BANK MANAGEMENT SYSTEM**" is the bonafide work of **ARUNKUMAR A** who carried out the project work under my supervision.

**Submitted by:**

**ARUNKUMAR A**

- **Project Source Code:** <https://github.com/Arunkumar7824/Bank-Management-System>
- **LinkedIn** : <https://www.linkedin.com/in/arunkumar5059/>
- **GitHub** : <https://github.com/Arunkumar7824>
- **Email** : arunkumar.a.5059@gmail.com

**Date:** \_\_\_\_\_

# ABSTRACT

The **Bank Management System** is a Java-based application developed to simulate the core functionalities of a banking system. It provides a user-friendly interface for customers to perform essential banking operations such as creating an account, depositing money, withdrawing money, checking balance, and viewing transaction history.

This project is developed using **Java** for the front-end with **Swing (GUI)**, and **MySQL** as the back-end database. It ensures secure transactions and provides an efficient way to manage customer data and transaction details.

The system aims to reduce the manual effort involved in traditional banking and make daily operations faster, more secure, and reliable. By incorporating validation, error handling, and database connectivity, the application serves as a practical model of real-world banking systems.

## TABLE OF CONTENTS

| CHAPTER<br>NO | TITLE                                    | PAGE<br>NO |
|---------------|--|------------|
|               | ABSTRACT                                 | iii        |
| 1             | INTRODUCTION                             | 5          |
| 2             | SYSTEM REQUIREMENTS                      | 7          |
| 3             | SOFTWARE ANALYTICS                       | 8          |
|               | 3.1 Account Creation Module              |            |
|               | 1. Application Form ( Personal Details ) | 8          |
|               | 2. Personal Details                      | 10         |
|               | 3. Additional Details                    | 11         |
|               | 3.2 ATM Module                           |            |
|               | 1. Deposit                               | 15         |
|               | 2. Withdraw                              | 18         |
|               | 3. Fast Cash                             | 21         |
|               | 4. Mini Statement                        | 23         |
|               | 5. Change Pin                            | 26         |
|               | 6. Check Balance                         | 29         |

# INTRODUCTION

The **Bank Management System** is a Java-based desktop application designed to automate and simplify core banking operations. The system provides an interactive and user-friendly interface for both customers and banking staff, enabling them to manage bank-related tasks efficiently and securely.

The project is developed using **Java** with **Swing** for the Graphical User Interface (GUI), and **MySQL** as the backend database to store customer and transaction information. This ensures a structured, scalable, and reliable system for handling banking operations.

The system is broadly divided into two main modules:

## 1. Account Creation Module

In this module, a new user can create a bank account by filling out three structured forms:

- **Application Form:** Captures personal details such as name, DOB, address, etc.
- **Additional Details:** Gathers extra information like income, occupation, and PAN details.
- **Account Details:** Allows the user to choose the account type and services, generating the account number and PIN.

## 2. ATM Simulation Module

Once the account is created, the user can access ATM functionalities using their PIN. The ATM module includes the following features:

- **Deposit:** Allows the user to add money to their account.
- **Withdraw:** Enables withdrawal of a specific amount.
- **Fast Cash:** Provides quick withdrawal options with fixed denominations.
- **Check Balance:** Displays the current balance in the account.
- **Mini Statement:** Shows the recent transaction history.
- **Change PIN:** Lets the user securely update their ATM PIN.

# SYSTEM REQUIREMENTS

## Software Requirements

- **Operating System:** Windows 10
- **Programming Language:** Java
- **User Interface:** Java Swing (GUI Framework)
- **Database:** MySQL
- **Database Connectivity:** JDBC (Java Database Connectivity)
- **IDE Used:** NetBeans IDE
- **Additional Tools:** MySQL Workbench (for managing database)

These software tools provide the required environment to develop, execute, and test the **Bank Management System** efficiently.



# SOFTWARE ANALYTICS



Create Account Here

[Click Here](#)



Use ATM Here

[Click Here](#)

*Home Page*

## 1. Account Creation Module

The **Account Creation Module** is the first and most important part of the Bank Management System. It allows a new user to open a bank account by entering all necessary details through three structured and connected forms. Each form plays a crucial role in gathering, validating, and storing data in the **MySQL database** using **JDBC connectivity**.

### 1.1 Application Form

This is the first step of the account creation process. It collects the user's **personal details**, which are essential to initiate a new bank account.

Application Form

**APPLICATION FORM**

Form NO : 3467      **Personal Details**      Page 1

Name : Arunkumar

Father's Name : Aruljothi

Gender : ☒ Male ☐ Female ☐ Others

Date of Birth : Mar 8, 2002

Email address : arunkumar.a.505

Marital Status : ☐ Married

Address : 64 A, Angalamma

City : Panruti

State : TamilNadu

Pin Code : 607108

back      Next

### Application Form (Personal Details)

#### Details Collected:

- Full Name
- Father's Name
- Gender
- Date of Birth
- Email Address
- Marital Status
- Address
- City
- State
- Pincode

## Buttons Used:

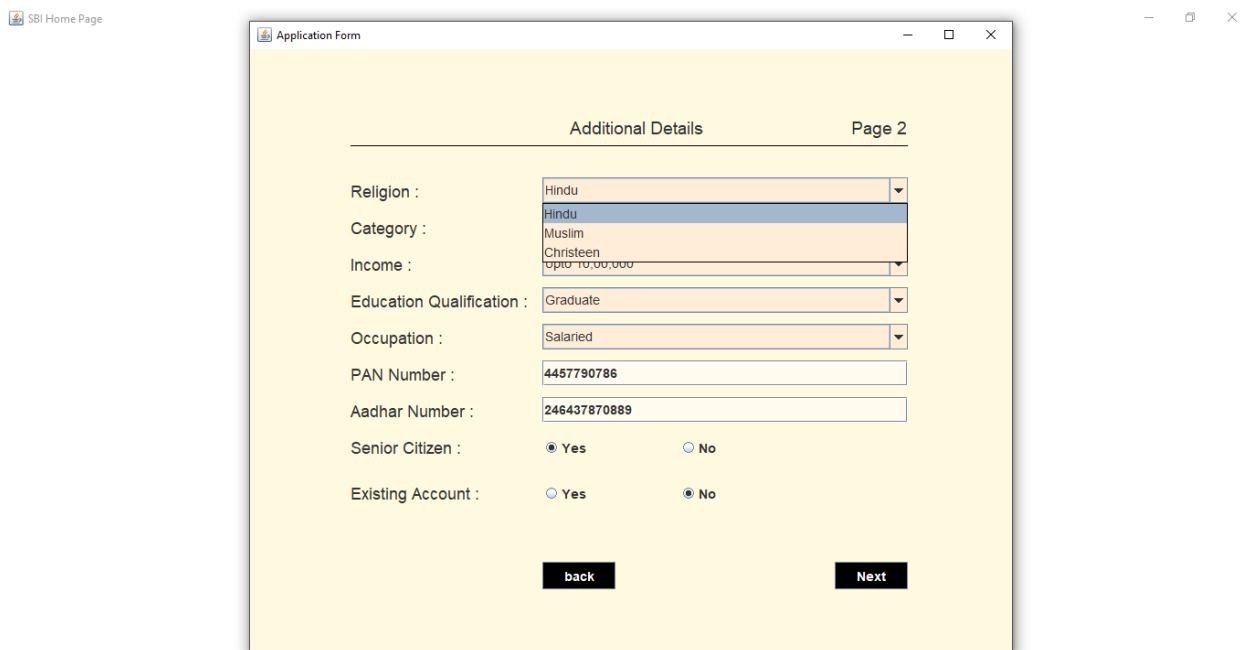
- **Back:** Navigates to the previous screen (if any).
- **Next:** Proceeds to the **Additional Details** form.

## Functionality:

- ✓ Once the user fills out all fields and clicks **Next**, the entered data is validated.
- ✓ If valid, the data is stored in a **MySQL table** (e.g., application\_form) through JDBC.
- ✓ The user is then redirected to the next form.

## 1.2 Additional Details Form

This is the second step of the account creation. It collects extra information to strengthen the user's banking profile and for verification purposes.



The image shows a screenshot of a web browser window with the title 'SBI Home Page' and a smaller window titled 'Application Form'. The 'Application Form' window displays the 'Additional Details' section, labeled 'Page 2'. The form contains the following fields and options:

- Religion :
- Category :
- Income :
- Education Qualification :
- Occupation :
- PAN Number :
- Aadhar Number :
- Senior Citizen : ☒ Yes ☐ No
- Existing Account : ☐ Yes ☒ No

At the bottom of the form, there are two buttons: 'back' and 'Next'.

*Additional Details Form*

## Details Collected:

- Religion
- Category (e.g., BC,MBC,BMC, SC, ST)
- Income
- Educational Qualification
- Occupation
- PAN Number
- Aadhaar Number
- Senior Citizen (Yes/No)
- Existing Account (Yes/No)

## Buttons Used:

- **Back:** Returns to the **Application Form**.
- **Next:** Proceeds to the **Account Details** form.

## Functionality:

- ✓ On clicking **Next**, the entered data is validated.
- ✓ If valid, the data is stored in a **MySQL table** (e.g., additional\_details) using JDBC.
- ✓ The user is then redirected to the final account setup form.

## 1.3 Account Details Form

This is the final form where the user selects the account type, required services, and submits the account creation request.

**Account Details** Page 3

**Account Type:**

☒ Saving Account ☐ Fixed Deposit Account

☐ Current Account ☐ Recurring Deposit Account

**Card Number:** XXXX XXXX XXXX 4256  
(Your 16-digit Card Number)

**PIN:**  
(4-digit password)

**Services Required:**

☐ ATM Card ☒ Internet Banking

☐ Mobile Banking ☐ EMAIL Alerts

☐ Cheque Book ☐ E-Statement

☐ I declare that the above details are correct.

**back** **submit** **cancel**

**Message**

✓ Card No : 4207 1910 6005 4256  
✓ Pin No : 5698

**OK**

### Account Details Form

#### Details Collected:

- Account Type (e.g., Savings, Current, Fixed Deposit, Recurring Deposit)
- Required Services (e.g., ATM Card, Internet Banking, Mobile Banking, Email Alerts, Cheque Book, E-Statement)
- Auto-generated Account Number
- Auto-generated ATM PIN

#### Buttons Used:

- **Back:** Returns to the **Additional Details** form.
- **Submit:** Saves all the information into the **MySQL table** (e.g., account\_details) and confirms account creation.
- **Cancel:** Cancels the account creation process and exits the form.

## Functionality:

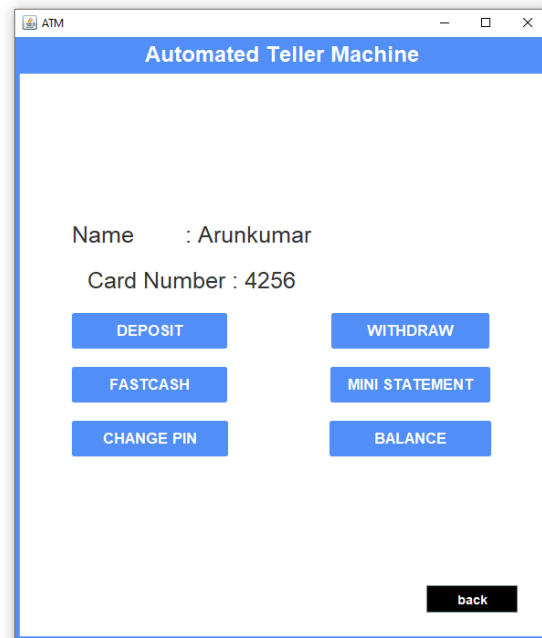
- ✓ Upon clicking **Submit**, the form gathers the user's selected options.
- ✓ A unique **Account Number** and a secure **ATM PIN** are generated automatically.
- ✓ All the collected data is saved in the **MySQL database** using JDBC.
- ✓ A confirmation message is shown to the user upon successful account creation.
- ✓ If **Cancel** is clicked, the form is reset or closed without saving data.

## Backend Integration

- ❖ All three forms are connected to the **MySQL database** using **JDBC**.
- ❖ Each form's data is saved in separate tables (application\_form, additional\_details, and account\_details).
- ❖ Proper validation and exception handling are used to ensure secure and error-free data entry.
- ❖ The user interface is built using **Java Swing**, making the forms interactive and user-friendly.

## 2. ATM Module

The **ATM Module** in this project acts as the **user interface** for performing various banking transactions. This screen is displayed **after the user has created an account** by filling out three forms. All the entered data (Name, Card Number, PIN, etc.) is stored in the **MySQL database**, and this module retrieves the required user information using **JDBC**.



### Automated Teller Machine

#### ◆ Display User Details:

- Automatically fetches the **latest user's name** from the accountPage1 table.
- Retrieves the **Card Number** from the accountPage3 table.
- These details are displayed at the top of the ATM screen using **Swing labels**.

#### ◆ Six Functional Buttons:

Each button opens a **separate JFrame (page)** with its own code and logic.

- **Deposit** – Opens the deposit window to add money.
- **Withdraw** – Opens the withdrawal window to remove money.
- **FastCash** – Instantly withdraws fixed amounts (₹100, ₹500, ₹1000, etc.).

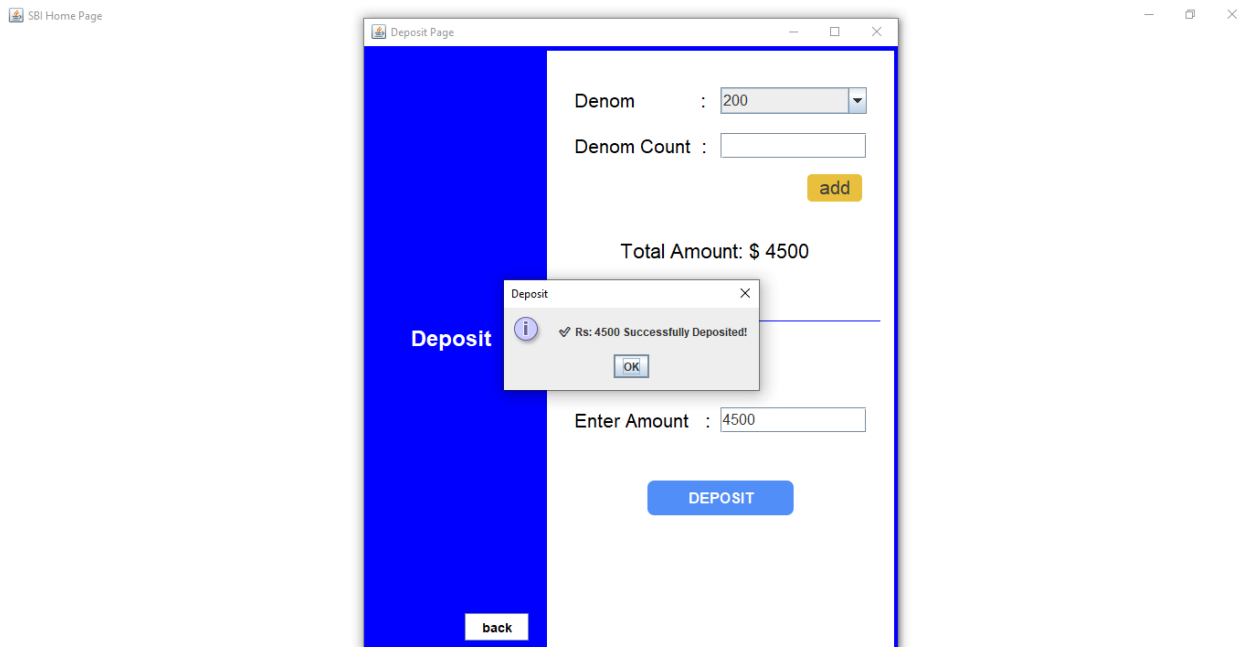
- **Mini Statement** – Shows the latest transactions made by the user.
- **Change PIN** – Allows the user to securely update their ATM PIN.
- **Check Balance** – Displays the user's current account balance.

### ◆ **Back Button:**

- The **Back** button is used to return to the previous screen or exit the ATM module.

## 2.1 DEPOSIT

The Deposit module is a key component of the Bank Management System developed using Java Swing and MySQL. It allows users to securely deposit cash into their account using a graphical interface that mimics ATM denomination handling.



*Deposit*



## Module Flow:

### 1. Deposit Button:

- On clicking the "Deposit" button from the userATM interface, a new frame titled "**Deposit**" is launched.
- This frame is implemented using JFrame and JPanel for organized component placement and styling.

### 2. Denomination Section:

- The user is presented with:
  - A **Denomination Text Field** (denomTextField) to input a denomination value (e.g., 100, 200, 500).
  - A **Denomination Count Text Field** (denomCountTextField) to input the count of that denomination.
  - An "**Add**" **Button**, which, when clicked:
    - Multiplies the denomination with its count.
    - Adds it to a **running total**, which is displayed below using a JLabel or uneditable JTextField.

### 2. Total Amount Display:

- As the user adds denominations, the calculated total deposit amount is shown dynamically.
- This ensures clarity and prevents manual calculation errors.

### 3. Manual Entry and Validation:

- Below the denomination section, a text field labeled "**Enter Your Amount**" is provided (enterAmountField).
- On clicking the "**Deposit**" **Button**, the following validations occur:
  - If the manually entered amount matches the calculated total:
    - The amount is **added to the existing balance** in the accountPage3 table (MySQL database).
    - A corresponding **entry is made in the mini-statement table** for record-keeping.
    - A success dialog or message is shown.
  - If the amounts do not match:
    - An error dialog is displayed, such as "**Entered amount does not match total denomination amount. Please check again.**"

#### 4. Database Integration:

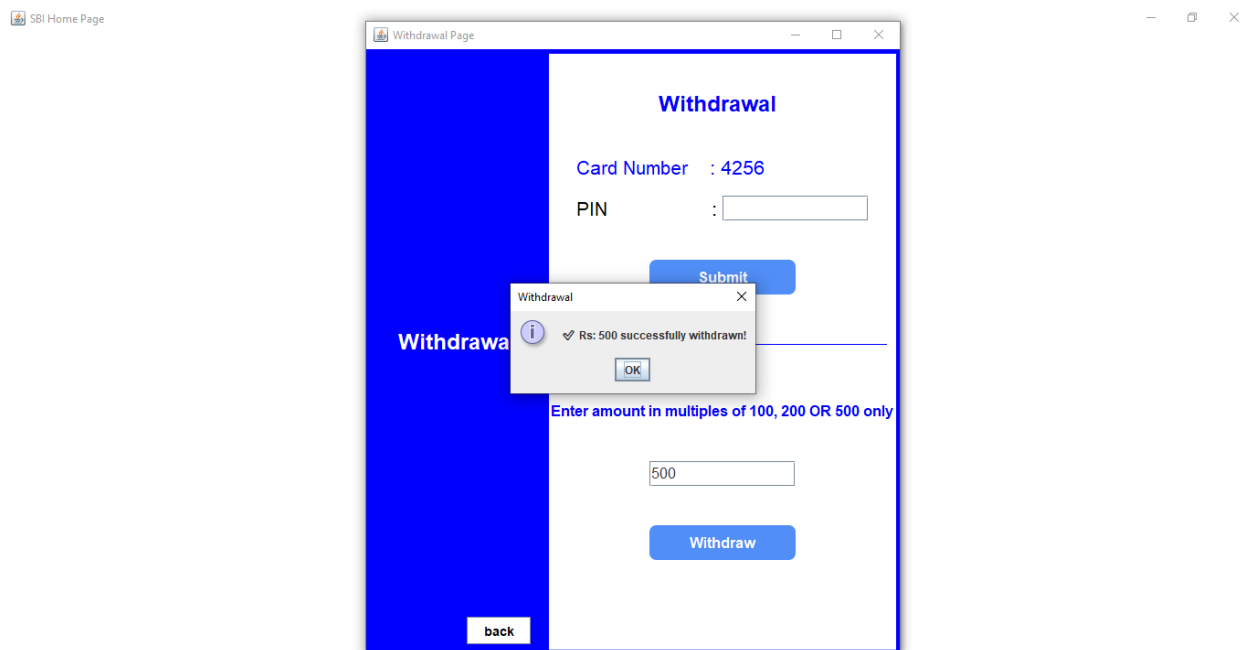
- On successful validation:
  - The balance column is updated using an UPDATE query in JDBC.
  - A new mini-statement record is inserted using an INSERT query.
- MySQL tables involved:
  - accountPage3 (stores account balance)
  - ministatement (stores transaction logs)

## ← BACK Back Navigation

- A "**Back**" button is provided in the FastCash options frame.
- When clicked, it allows the user to **return to the ATM main page**, improving navigation and user control.

## 2.2 WITHDRAW

The **Withdraw Module** of the Bank Management System is designed to provide a secure and interactive way for users to withdraw cash from their account. The user interface is built using **Java Swing** components, and data operations are handled using **MySQL**. The system ensures that only authenticated users can access the withdrawal feature and supports withdrawal in limited denominations for realistic ATM behavior.



Withdraw

## ◆ User Interface Design

- When the user clicks the **"Withdraw"** button on the main ATM interface, a **new frame** is launched using JFrame.
- This frame is styled using **JPanel layouts** to organize the UI elements in a clean and professional manner.
- The top section of the frame displays the **card number** of the currently logged-in user.
  - This **card number is automatically fetched from the MySQL database**, using the user's session or login credentials.

## 🔒 PIN Verification Mechanism

- The frame includes a **PIN number input field** where users must enter their 4-digit PIN.
- By default, the **withdrawal amount input section and action buttons are disabled** to prevent unauthorized access.
- After entering the PIN, the user clicks the **"Submit"** button.
  - If the PIN is **correct** (validated by checking the value in the accountPage3 table in the database), the **hidden/disabled section becomes enabled**, allowing the user to proceed.
  - If the PIN is **incorrect**, an error message is displayed: *"Invalid PIN. Please try again."*

## 💰 Amount Input & Validation

- Once the PIN is verified, the user is allowed to enter the **amount** they wish to withdraw.

- The system only allows amounts that are **multiples of ₹100, ₹200, or ₹500**, which simulates real ATM currency availability.
- Input validation is performed:
  - If the amount is **not a valid denomination**, the system shows an error:  
*"Only ₹100, ₹200, and ₹500 notes are available. Please enter a valid amount."*
  - If the amount **exceeds the available balance**, an error is shown:  
*"Insufficient Balance."*

## **Back Navigation**

- A **"Back"** button is provided in the FastCash options frame.
- When clicked, it allows the user to **return to the ATM main page**, improving navigation and user control.

## **Database Update and Transaction Record**

- On successful validation:
  - The entered amount is **deducted** from the user's current balance in the **accountPage3** table.
  - The transaction is recorded in a **mini-statement/history table** for audit purposes.
- A message is shown to the user:  
*"Transaction Successful. Please collect your cash."*

## 2.3 FASTCASH

The **Fast Cash Module** of the Bank Management System is designed to provide users with a **quick and convenient method** to withdraw commonly used cash amounts with just one click. This feature simulates real-world ATM functionality by offering predefined withdrawal options and instant cash delivery after secure verification.



*Fast Cash*

### ◆ User Interface Design

- When the user clicks the "**FastCash**" button on the ATM home screen, a **new frame** opens using JFrame.
- The frame is built using **Java Swing** with JPanel to maintain a structured and user-friendly design layout.

- At the top of the frame, the **user's card number is automatically displayed**, fetched in real-time from the **MySQL database**.

## PIN Verification Mechanism

- Below the card number, there is an **input field** for the user to enter their **PIN number**.
- After entering the PIN, the user must click the **"Submit"** button to verify.
  - If the **PIN is correct**, a **new Fast Cash frame** is launched.
  - If the **PIN is incorrect**, the system displays an error: *"Incorrect PIN. Please try again."*

## Fast Cash Amount Selection

- Once the PIN is verified, the user is directed to a new frame that contains **seven FastCash buttons** representing commonly withdrawn amounts:
  - ₹100, ₹200, ₹500, ₹1000, ₹2000, ₹5000, and ₹10000
- These amounts are chosen to match **realistic ATM denomination options**, making the user experience fast and intuitive.

## Instant Withdrawal

- When the user clicks any of the FastCash buttons:
  - The system checks if the user has **sufficient balance** in their account.

- If yes:
  - The selected amount is **immediately deducted** from the user's balance in the accountPage3 table.
  - A record of the withdrawal is added to the **mini-statement table** for transaction history.
  - A confirmation message appears:  
*"₹<amount> has been withdrawn successfully. Please collect your cash."*
- If the balance is insufficient, an error is displayed:  
*"Insufficient balance. Please try with a lower amount."*

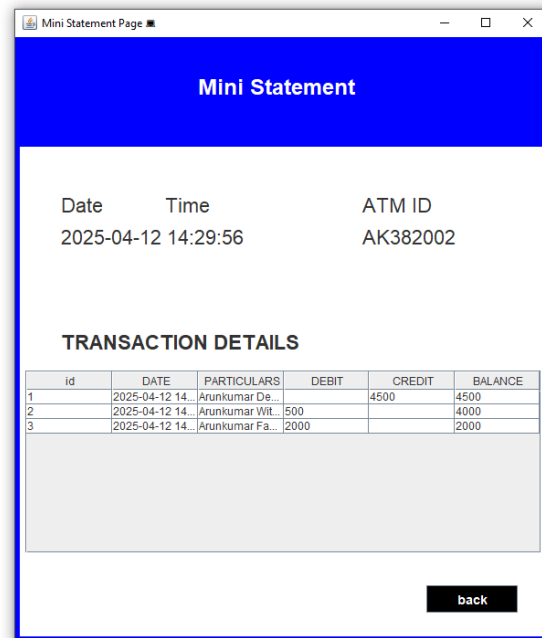
## **Back Navigation**

- A "**Back**" button is provided in the FastCash options frame.
- When clicked, it allows the user to **return to the ATM main page**, improving navigation and user control.

## 2.4 Mini Statement

The **Mini Statement Module** in the Bank Management System allows users to securely view their recent transactions, including **deposits** and **withdrawals**, along with essential details like **ATM ID**, **date and time**, and **card number**. This module enhances **user transparency** and helps keep track of account activity.





## User Interface Design

- When the user clicks the **"Mini Statement"** button on the ATM home screen, a **new frame** opens using JFrame.
- The UI layout is designed with JPanel, creating a neat and structured view of the cardholder's transaction history.
- The **card number is displayed automatically**, fetched securely from the **MySQL database** using JDBC.

## PIN Number Verification

- The interface provides a **PIN input field** where users must enter their **4-digit PIN**.
- After entering the PIN, the user clicks the **"Submit"** button:

- If the PIN is **correct**, a **new frame** appears, showing the mini statement.
- If the PIN is **incorrect**, the system shows a message:  
*"Incorrect PIN. Please try again."*

## **Mini Statement Details**

After successful PIN verification, the system displays a new frame that includes:

- **Current Date and Time** – Generated using Java's date and time API (LocalDateTime).
- **ATM ID** – A unique identifier for the ATM (can be hardcoded or dynamic depending on your system).
- **Recent Transactions:**
  - Shows both **deposits** and **withdrawals** from the mini statement table.
  - Displays information such as **amount**, **type** (Deposit/Withdrawal), and **date/time of transaction**.

## **Back Button Functionality**

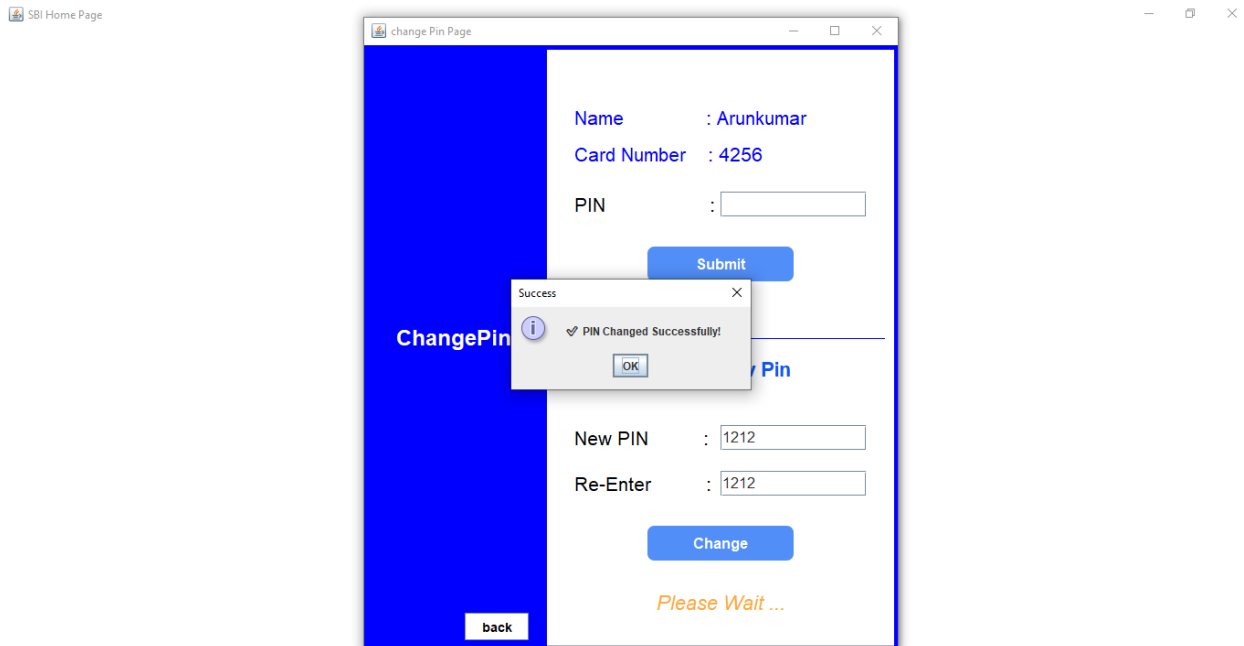
- A **"Back"** button is available to return to the **ATM main menu** or previous screen.
- This improves navigation and allows users to perform other transactions seamlessly.

## **Database Connectivity**

- The card number and PIN are matched with values in the accountPage3 table.
- Recent transactions are fetched from the **mini statement table** using a SELECT query.
- Transactions are displayed in **reverse chronological order** to highlight the latest activity.

## 2.5 Change Pin

The **Change PIN Module** of the Bank Management System allows users to securely update their ATM PIN. This ensures that users can manage their account privacy and maintain account security in case their current PIN is compromised or needs to be updated.



Change Pin

## User Interface Design

- When the user clicks the **“Change PIN”** button on the ATM main screen, a new frame opens using JFrame.
- The GUI is designed with JPanel for a structured and clean layout.
- The **card number is automatically displayed**, fetched from the **MySQL database** using JDBC.

## 🔒 PIN Number Verification

- A **text field** is provided where the user must enter their **current 4-digit PIN**.
- The submit button checks the entered PIN with the one stored in the database.
- Until a correct PIN is entered, the rest of the form remains in **disabled mode** (fields and buttons are disabled).

## ✓ Enabling Change PIN Panel

- If the entered PIN is **correct**, the hidden content (i.e., the “Change PIN” section) becomes **enabled**:
  - Two text fields are shown:
    - **New PIN**
    - **Re-enter New PIN**
- These fields are used to ensure the new PIN is entered correctly and confirmed.

## ❓ Validation Logic

- Both PIN fields must:

- Be **identical**
  - Contain exactly **4 digits** only (numeric)
- If the validation passes:
- The new PIN is updated in the **database** using an UPDATE SQL query.
  - A success message is shown using JOptionPane:  
*"PIN changed successfully!"*
- If validation fails:
- Appropriate error messages are displayed:
    - *"PINs do not match."*
    - *"PIN must be 4 digits only."*

## ↻ Back Navigation Option

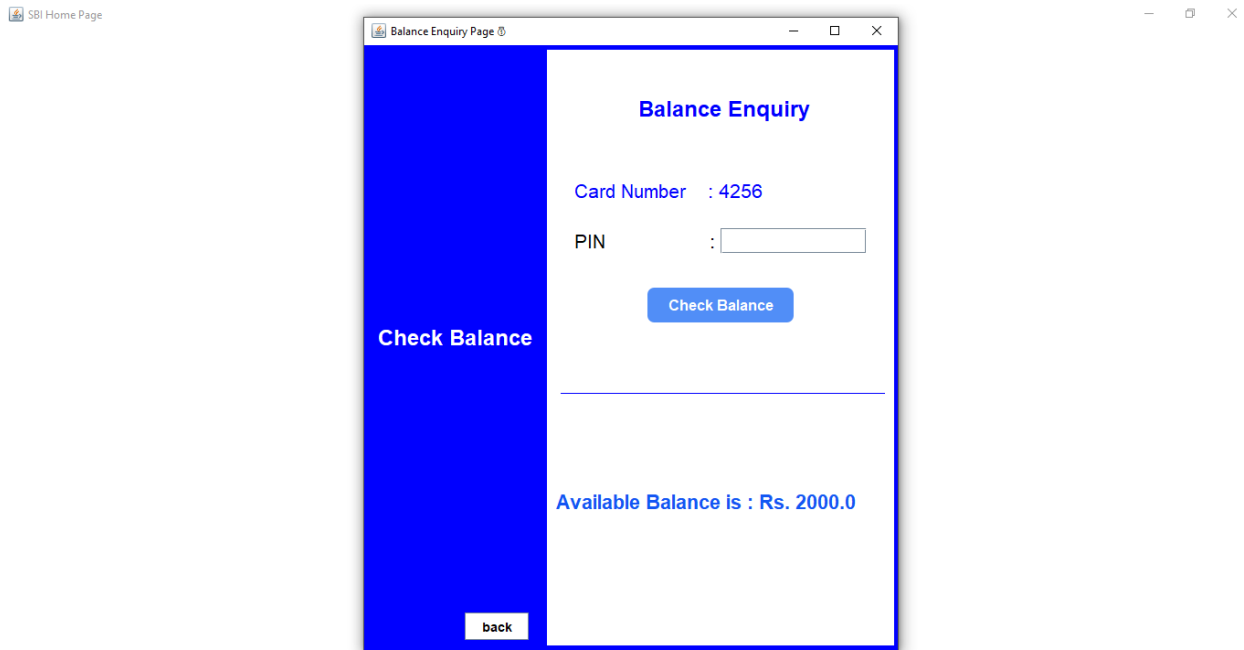
- Optionally, a **Back** button is provided to return to the ATM main menu without making changes.

## ☛ Database Connectivity

- Card number is retrieved from the logged-in session or from a previous frame.
- PIN verification and update operations interact with the **accountPage3** table using:
- SELECT query for PIN verification
  - UPDATE query for PIN change

## 2.6 Check Balance

The **Balance Enquiry Module** allows ATM users to securely check their account balance after successful PIN verification. It is a simple and essential part of the ATM system that ensures only authorized users can view sensitive financial information.



Check Balance

### User Interface & Design

- When the user clicks the "**Balance**" button on the ATM home screen, a new **Balance frame** is opened.
- This frame is built using **Java Swing** components, especially using **JPanel** for structured layout and styling.

- The **card number** is automatically fetched from the database using the logged-in session or from the previous frame and displayed at the top of the screen in a **non-editable text field**.

## PIN Verification Phase

- The frame contains a **PIN entry text field** where the user must enter their current PIN.
- A **Submit** button is provided to validate the PIN against the database.
- **Initially**, the section that displays the balance is **hidden or disabled** using `.setEnabled(false)` or `.setVisible(false)` to prevent unauthorized access.

## ✓ Unlocking Balance Info After PIN Verification

- When the correct PIN is entered and submitted:
  - ✓ A **SQL SELECT query** checks the card number and PIN combination from the database:  

```
SELECT balance FROM accountPage3 WHERE card_number = ?  
AND pin = ?
```
  - ✓ If the PIN is correct:
    - The disabled balance content becomes **enabled/visible**.
    - A message like:  
  
“**Available Balance: ₹0.00**” is displayed in a large bold JLabel.

## ⚠ Error Handling

### ➤ Incorrect PIN:

- Shows a JOptionPane message: *"Incorrect PIN. Please try again."*

### ➤ Empty PIN Field:

- Displays a warning: *"Please enter your PIN."*

## 🗄 Database Integration

- The application uses **MySQL** to manage user data.
- The balance is fetched from the accountPage3 table.
- The system uses **PreparedStatement** to prevent SQL injection and ensure secure database queries.