ATM System

1. Introduction

The **ATM System** is a Java-based application that provides essential banking functionalities such as user login, deposit, withdrawal, balance inquiry, and PIN-based authentication. The system uses **JDBC** to connect to a MySQL database and **Swing** for the graphical user interface (GUI).

2. Features

- User authentication with account number and PIN
- Cash withdrawal with balance verification
- Deposit money into an account
- Check account balance
- Transaction history tracking
- PIN change functionality
- Database connectivity using JDBC
- GUI interface using Java Swing

3. Technologies Used

- **Programming Language**: Java
- **Database**: MySQL
- **GUI Framework**: Java Swing
- **Database Connectivity**: JDBC
- **Layout Used**: GridLayout (for main GUI)

4. Database Structure

4.1 Users Table (users1)

Column Name	**Data Type**	**Description**
id	INT (PK)	Unique identifier
account_number	VARCHAR	Unique account number
pin	VARCHAR	User's PIN (hashed)
balance	DOUBLE	User's account balance

4.2 Transactions Table (transactions1)

Column Name	Data Type	Description
id	INT (PK)	Unique transaction ID

account_number	VARCHAR	Account related to transaction
amount	DOUBLE	Amount of transaction
transaction_type	VARCHAR	Type: Deposit or Withdrawal
timestamp	TIMESTAMP	Date and time of transaction

5. Code Explanation

5.1 Database Initialization

The initializeDB() method establishes a connection with the MySQL database using **JDBC**.

```
private void initializeDB() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/atm?useSSL=false&serverTimezone=UTC";
        String user = "root";
        String password = "root";
        conn = DriverManager.getConnection(url, user, password);
        System.out.println("✅ Connected to database successfully.");
    } catch (Exception e) {
        System.out.println("❌ Database Connection Failed!");
        e.printStackTrace();
    }
}
```

5.2 GUI Setup

The createGUI() method initializes the graphical interface using **Java Swing** components like JFrame, JTextField, JButton, and JTextArea. The layout used is **GridLayout** to structure the UI elements efficiently.

```
private void createGUI() {
    frame = new JFrame("ATM System");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new GridLayout(8, 2, 5, 5));

    frame.add(new JLabel("Account Number:"));
    accountField = new JTextField();
    frame.add(accountField);

    frame.add(new JLabel("PIN:"));
    pinField = new JPasswordField();
    frame.add(pinField);

    JButton loginButton = new JButton("Login");
    frame.add(loginButton);

    frame.add(new JLabel("Amount:"));
    amountField = new JTextField();
    frame.add(amountField);

    JButton depositButton = new JButton("Deposit");
    JButton withdrawButton = new JButton("Withdraw");
    JButton balanceButton = new JButton("Check Balance");
    JButton changePinButton = new JButton("Change PIN");

    frame.add(depositButton);
    frame.add(withdrawButton);
    frame.add(balanceButton);
    frame.add(changePinButton);

    outputArea = new JTextArea(5, 30);
    outputArea.setEditable(false);
    frame.add(new JScrollPane(outputArea));

    frame.setVisible(true);
}
```

5.3 User Login

The loginUser() method checks the account number and PIN in the database.

```
private void loginUser() {
    String accountNumber = accountField.getText();
    String pin = new String(pinField.getPassword());

    try {
        String query = "SELECT * FROM users1 WHERE account_number=? AND pin=?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, accountNumber);
        pstmt.setString(2, pin);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            outputArea.setText("✅ Login Successful!");
        } else {
            outputArea.setText("❌ Invalid credentials!");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

5.4 PIN Change Feature

The changePin() method allows the user to update their PIN.

```
private void changePin() {
    try {
        String accountNumber = accountField.getText();
        String newPin = new String(pinField.getPassword());

        String updatePin = "UPDATE users1 SET pin = ? WHERE account_number = ?";
        PreparedStatement pstmt = conn.prepareStatement(updatePin);
        pstmt.setString(1, newPin);
        pstmt.setString(2, accountNumber);

        pstmt.executeUpdate();


        outputArea.setText("✅ PIN changed successfully.");

    } catch (Exception e) {

        e.printStackTrace();

        outputArea.setText("❌ PIN change failed!");

    }

}
```

Conclusion

This **ATM System** provides a functional and secure banking experience using Java, Swing, and MySQL. It includes features like **PIN authentication, withdrawals, deposits, balance inquiry, and PIN change functionality**, making it a complete ATM-based application.