# RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM 602 105


## CS23333 OOPS Using Java

Laboratory Record Note

Name :

Year / Branch / Section

: University Register ..

College Roll No. : .

Semester .

Academic Year : .

JALAKSHMI ENGINEERING COLLEGE

**n Autonomous Institution**

## BONAFIDE CERTIFICATE

Name: ……

Academic Year: …… Semester: …… Branch: ……

Register No.

*Certified that this is the bonafide record of work done by the above student in the…… Laboratory during the academic year 2025- 2026*

Signature of Faculty in-charge

Submitted for the Practical Examination held on……

Internal Examiner

External Examiner

| EX.NO | DATE | NAME OF THE EXPERIMENT | GITHUB QR |
|---|---|---|---|
| 1 | | I/O, Data Types, Operators | |
| 2 | | Control Structures | |
| 3 | | Arrays | |
| 4 | | Strings | |
| 5 | | Classes & Objects | |
| 6 | | Inheritance | |
| 7 | | Interface | |
| 8 | | Exceptions | |
| 9 | | Collections | |
| 10 | | Collections | |
| 11 | | Project | |
| 12 | | Lambda | |

# ABSTRACT

The main objective of the project is to develop online Banking system for banks. In present system all banking work is done manually. User have to visit bank to Withdrawal or Deposit amount. In present bank system it is also difficult to find account information of account holder. In this bank management system we will automate all the banking process. In our bank management system user can check his balance online and he can also transfer money to other account online. In this Software you can keep record for daily Banking transactions. The main purpose of developing bank management system is to design an application, which could store bank data and provide an interface for retrieving customer related details with 100% accuracy.

This Digital bank management system also allow user to add new customer account, delete account and user can also modify existing user account information. Using this system user can also search any individual account in few seconds. Using our bank management system user can also check any translation in any account. Our system also provide security check to reduce fraud. The system will check the user's existence in the database and provide the set of services with respect to the role of the user.

## ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman MR. S. MEGANATHAN and the chairperson DR. M. THANGAM MEGANATHAN for their timely support and encouragement. We are greatly indebted to our respected and honorable principal Dr. S. N. MURUGESAN for his able support and guidance. Our heartfelt thanks to our Head of Department Dr. E. M. MALATHY and Deputy Head Dr. J. MANORANJINI for their constant encouragement throughout our project. We also extend our sincere gratitude to our internal guide Mr . N. KARTHIKEYAN for her valuable guidance and motivation during the completion of this project. Finally, we thank our family members, friends, and all staff members of the Computer Science and Engineering department for their continuous support.

## TABLE OF CONTENTS

# CHAPTER 1 - INTRODUCTION

The main object of this system is to provide a secure system. Our system is password protected and it only allows authorized user to access various functions available in the system.

Our system will help the user to Locate any A/C wanted by the user. It will Reduced manual work as most of the work done by computer. As all the manual work will be done automatically so it will increase work speed and reduce time consumption to complete any bank related work. It will also increase the work efficiency as few employees can handle more customers. This will reduced the manual workload and give information instantly.

The Project Banking system has been made to automate the Banking system. Through this bank management system user can manage all bank account activity like deposit money, withdraw money, transfer money from one account to another account, online payment etc. Using this bank management system user can check his account detail online like balance in account, bank statement etc. The Administrator can check bank account with a login can work out with A/C holders of the bank can withdraw/ deposit cash / cheque /DD to/from their accounts. This system is also help bank user to create New account easily. The project makes a sincere effort to provide all the below-mentioned features to meet the requirements of the bank.

In this project we have automate the bank process like Account Opening, Daily Transactions, Loan Sanctions, Account Maintenance. In this bank management system use can also search record of a particular Account Holder.

Using this system user can manage following account type

Savings Account

Current Account

Fixed Deposit Account

Recurring Deposit Account

Loan Account

# CHAPTER 2 -SYSTEM SPECIFICATIONS

Hardware Specifications:
Processor: Intel i5
RAM: 16 GB
Hard Disk: 500 GB

Software Specifications:
Operating System: Windows 11
Front-End: Java Script
Back-End: MySQL and JDBC (Java Database Connectivity)
Language: Java

## CHAPTER 3 - MODULE DESCRIPTION

A Digital Bank Management System project for a DBMS mini-project can be broken down into modules like **Account Management**, **Transactions**, **Customer Services**, and **Admin/Employee Management**. These modules handle core banking functions such as creating and managing customer accounts, processing deposits and withdrawals, providing additional services like loans or bill payments, and allowing bank staff to manage records and employee information.

User Management Module

This module handles all user-related data and functionalities.

- Customer Registration: Allows new users to create accounts by providing personal details like name, address, and contact information.
- Login/Logout: Provides secure access to the user's account with an authentication system.
- User Profile: Allows users to view and update their personal information.
- Admin Panel: Provides an interface for bank staff to manage user accounts

Account Management Module

This module is for managing customer bank accounts.

- Account Creation: Enables the creation of different account types (e.g., savings, current) with automated account number generation.
- Balance Inquiry: Allows users to check their current account balance.
- Account Details: Lets users view account-specific information such as account type and status.

Transaction Management Module

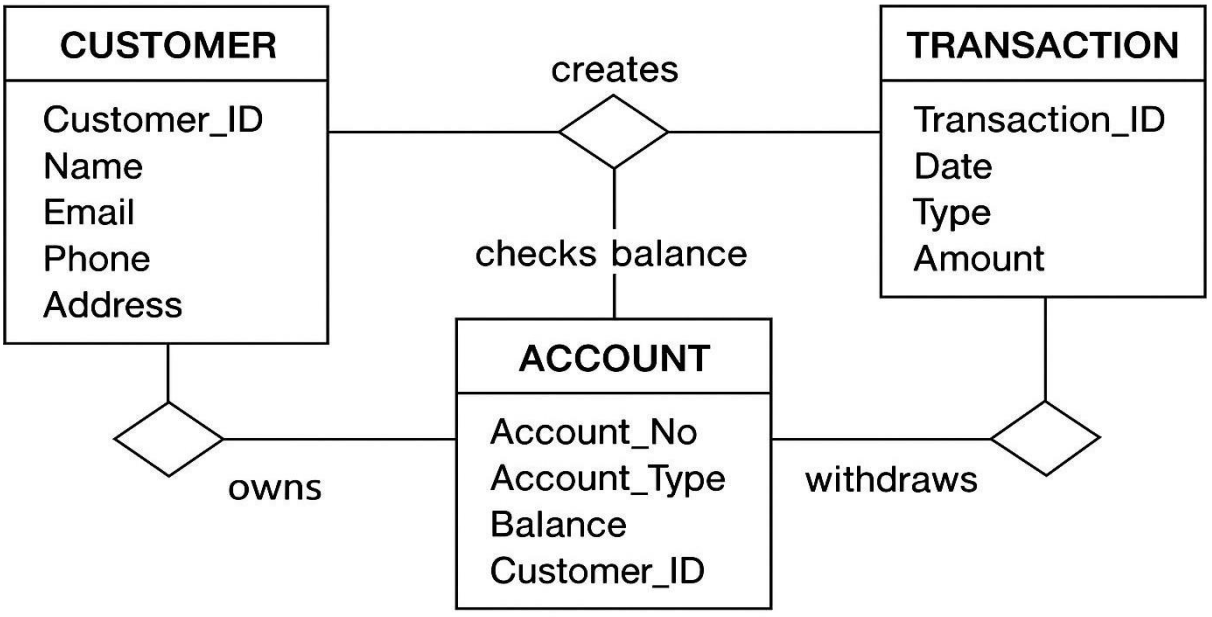
This module tracks and processes all financial transactions.

- Fund Transfer: Allows users to transfer money between accounts, including internal transfers.

- Deposit/Withdrawal: Supports basic banking operations for adding or removing funds.
- Transaction History: Provides a record of all past transactions, including deposits, withdrawals, and transfers.
- Transaction Reports: Generates reports showing transaction activity for a given period.
    - Bill Payment Module: Allows users to schedule and pay bills online.
    - Two-Factor Authentication (2FA): A critical security feature requiring two forms of identification for transactions or account access.
    - Data Analytics and Reporting: Provides insights into banking operations and customer behavior.
    - Card Management: Allows customers to manage debit and credit card-related services.

# CHAPTER – 4 ARCHITECTURE DIAGRAM & ER DIAGRAM

**4.1** ENTITY-RELATIONSHIP




4.1 ER Diagram

**4.2** ER DIAGRAM TO RELATIONAL MODEL

Show how each entity and relationship is converted into tables.

Entities and their Attributes are :

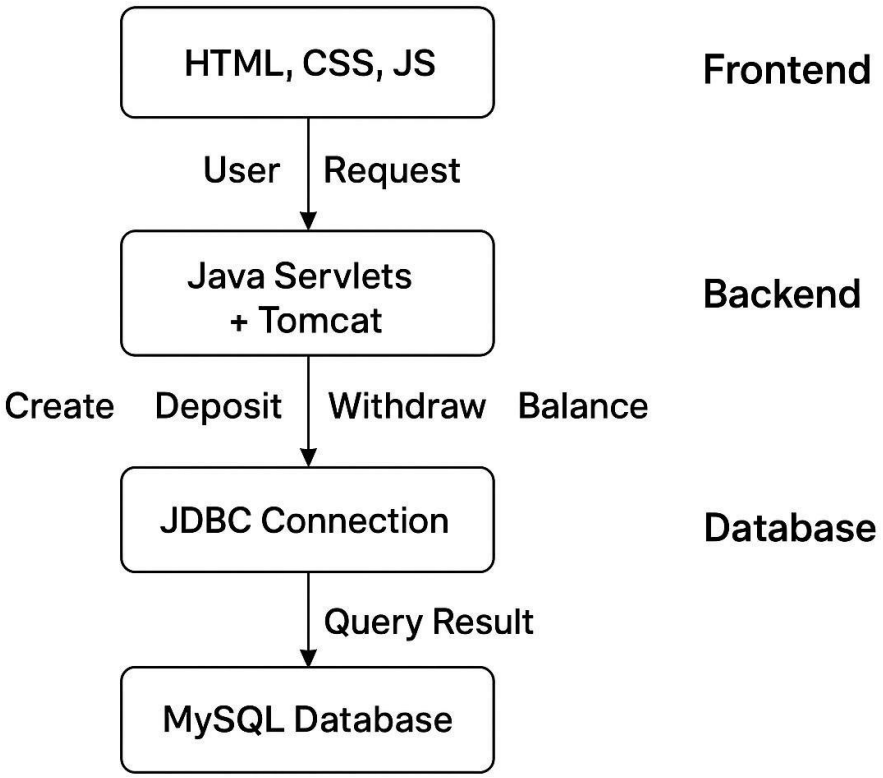
- Bank Entity : Attributes of Bank Entity are Bank Name, Code and Address.
  Code is Primary Key for Bank Entity.
- Customer Entity : Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.
  Customer_id is Primary Key for Customer Entity.
- Branch Entity : Attributes of Branch Entity are Branch_id, Name and Address.
  Branch_id is Primary Key for Branch Entity.
- Account Entity : Attributes of Account Entity are Account_number, Account_Type and Balance.
  Account_number is Primary Key for Account Entity.
- Loan Entity : Attributes of Loan Entity are Loan_id, Loan_Type and Amount.
  Loan_id is Primary Key for Loan Entity.

**4.2** RELATIONAL ALGEBRA

EXPRESSIONS DEFINITION

Relational Algebra is a formal language used to retrieve and manipulate data stored in relational databases. It uses mathematical operations like select, project, join, union, and rename to perform queries on tables.

**4.2** RELATIONAL SCHEMA DIAGRAM



**4.4** CONSTRAINTS USED IN THE PROJECT , In the Digital Bank Management System, various database constraints are applied to maintain **data integrity**, **accuracy**, and **consistency** across all tables. The commonly used constraints are **PRIMARY KEY Constraint, FOREIGN KEY Constraint, NOT NULL Constraint, CHECK Constraint, UNIQUE Constraint.**

CONSTRAINTS USED IN THE PROJECT

- **Customer_id**: `PRIMARY KEY`. This ensures every customer has a unique identifier and is the main way to reference them.
- **Name**: `NOT NULL`. A customer's name is a required field.
- **Phone_Number**: `UNIQUE`. Each phone number should be unique to prevent multiple accounts using the same contact information.
- **Email**: `UNIQUE`. The customer's email should be unique.
- **Address**: `NOT NULL`. The customer's address is a required field.
- **Date_of_Birth**: A `CHECK` constraint can be used to ensure the customer is of a certain age, such as 18 or older, to open an account.

**`ACCOUNT` table**

- **Account_number**: `PRIMARY KEY`. Every account must have a unique identifier.
- **Customer_id**: `FOREIGN KEY` referencing `CUSTOMER(Customer_id)`. This links an account to a specific customer.
- **Branch_id**: `FOREIGN KEY` referencing `BRANCH(Branch_id)`. This links an account to the branch where it was opened.
- **Account_Type**: `NOT NULL` and a `CHECK` constraint to enforce allowed types (e.g., 'Savings', 'Checking').
- **Balance**: `NOT NULL` and a `DEFAULT` constraint to set the initial balance to 0 upon account creation.
- **Balance**: A `CHECK` constraint to ensure the balance cannot be negative.

**`TRANSACTION` table**

- **TransactionID**: `PRIMARY KEY`. Each transaction must have a unique identifier.
- **Account_number**: `FOREIGN KEY` referencing `ACCOUNT(Account_number)`. This links a transaction to a specific account.
- **Date_of_Transaction**: `NOT NULL` and a `DEFAULT` constraint to set the value to the current date and time.
- **Transaction_Type**: `NOT NULL` and a `CHECK` constraint to restrict values to allowed transaction types (e.g., 'Deposit', 'Withdrawal', 'Transfer').
- **Amount**: `NOT NULL` and a `CHECK` constraint to ensure the amount is a positive number.

**`BRANCH` table**

- **Branch_id**: `PRIMARY KEY`. Unique identifier for each bank branch.
- **Name**: `UNIQUE` and `NOT NULL`. The branch name must be unique within the bank.
- **Address**: `NOT NULL`. Every branch needs an address.

## CHAPTER 5 - DATABASE IMPLEMENTATION

**6.1** DATABASE CONNECTIVITY CODE WITH JAVA:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

   private static final String URL = "jdbc:mysql://localhost:3306/bank"; // Replace 'bank' with
your database name

   private static final String USER = "root"; // Replace with your database username

   private static final String PASSWORD = "Harish@7200 "; // Replace with your database
password

   public static Connection getConnection() throws SQLException {

      try {

         Class.forName("com.mysql.cj.jdbc.Driver"); // Load the JDBC driver

         return DriverManager.getConnection(URL, USER, PASSWORD);

      } catch (ClassNotFoundException e) {

         System.err.println("JDBC Driver not found: " + e.getMessage());

         throw new SQLException("JDBC Driver not found.", e);

      }

   }
```

```
    public static void closeConnection(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                System.err.println("Error closing connection: " + e.getMessage());
            }
        }
    }
}
```

Code for inserting a customer:

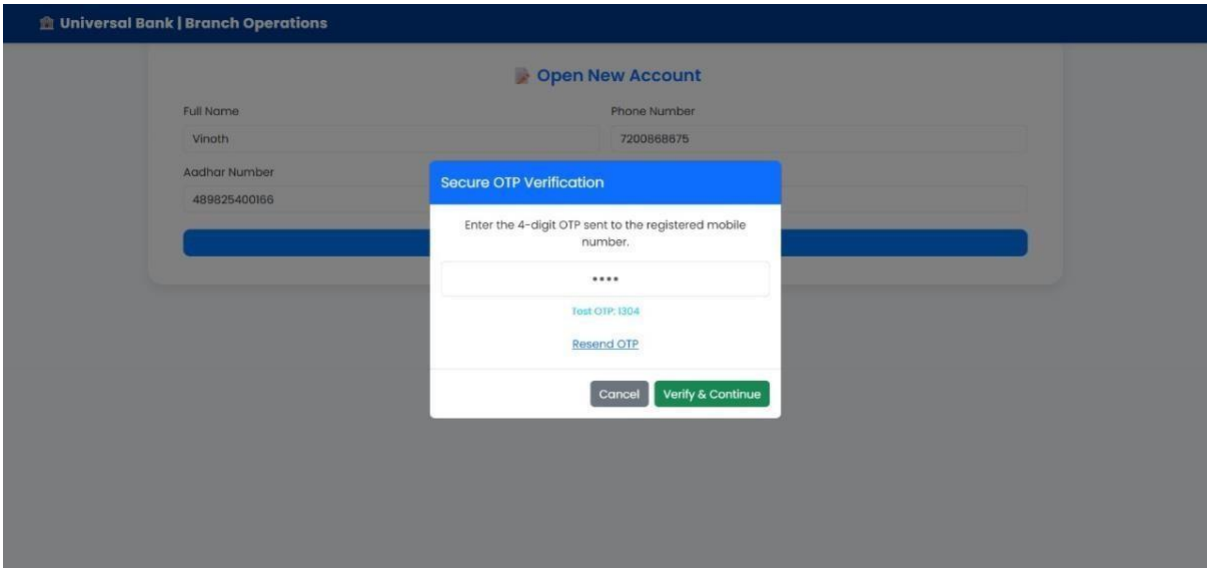
```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class CustomerDAO {
```

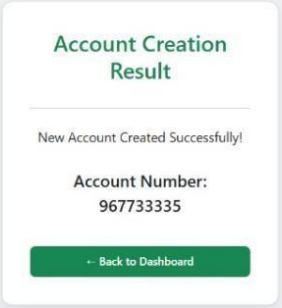
```
public void addCustomer(String name, String address, String phone) {
    String sql = "INSERT INTO customers (name, address, phone) VALUES (?, ?, ?)";
    try (Connection connection = DatabaseConnection.getConnection();
         PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setString(1, name);
        statement.setString(2, address);
        statement.setString(3, phone);
        statement.executeUpdate();
        System.out.println("Customer added successfully.");
    } catch (SQLException e) {
        System.err.println("Error adding customer: " + e.getMessage());
    }
}

public static void main(String[] args) {
    CustomerDAO customerDAO = new CustomerDAO();
    customerDAO.addCustomer("John Doe", "123 Main St", "555-1234");
}
```

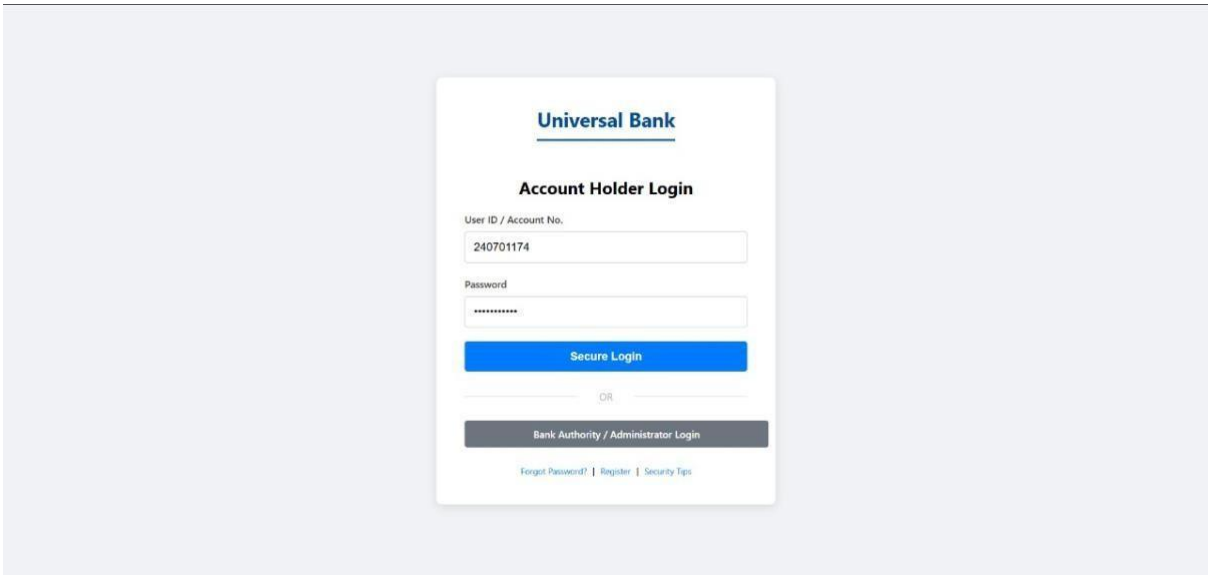
```
}

// connection.java

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;



public class ConnectionProvider {

   private static Connection con = null;



   public static Connection getCon() {

      try {

         // Load the MySQL JDBC driver

         Class.forName("com.mysql.cj.jdbc.Driver");



         // Establish the connection

         // Replace "username" and "password" with your MySQL credentials

         con = DriverManager.getConnection("jdbc:mysql://localhost:3306/BANK", "username",
"password");
```

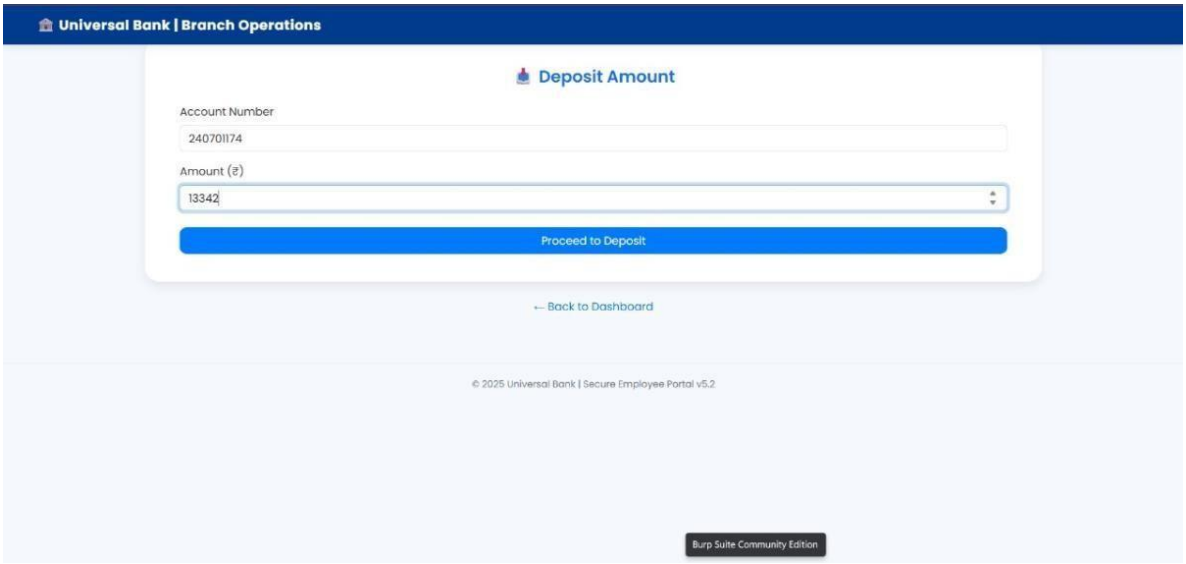
```
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return con;
    }
}
```

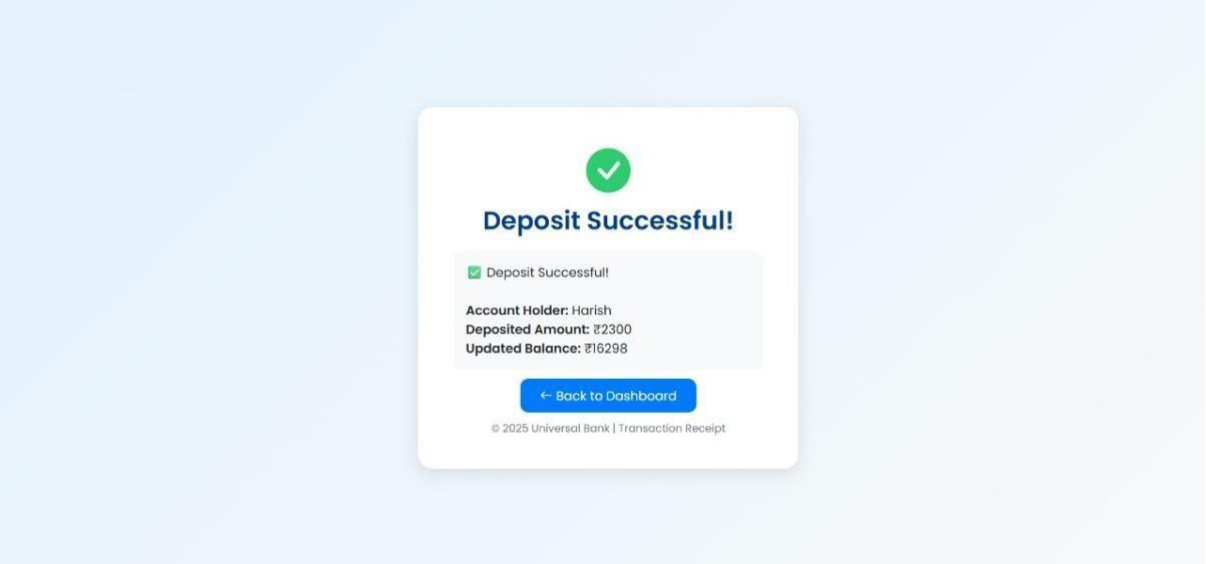
# CHAPTER 6 - SCREENSHOTS

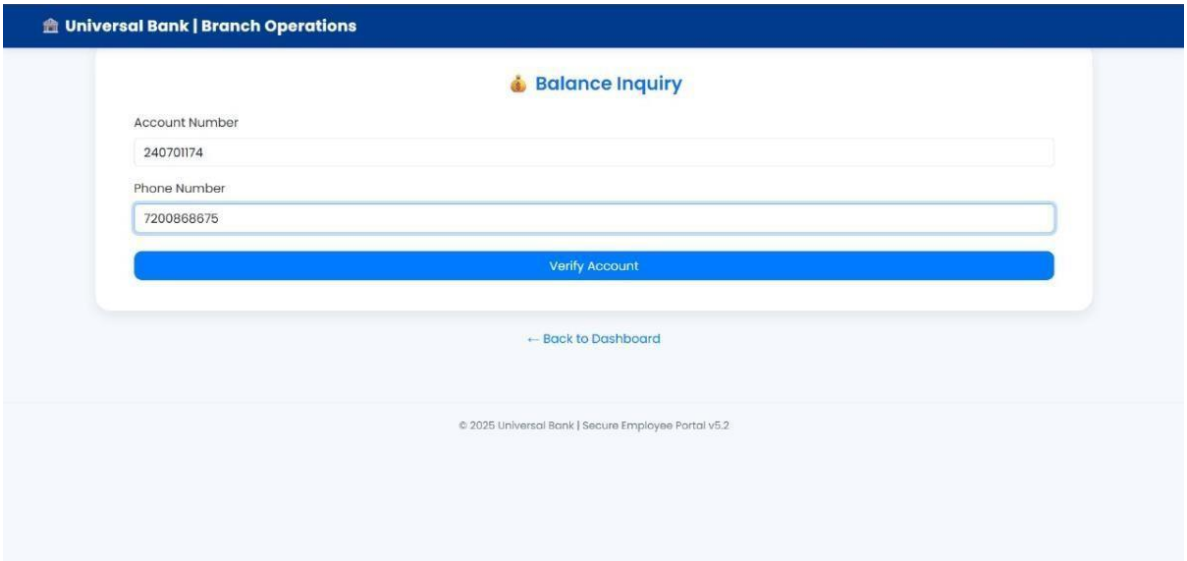


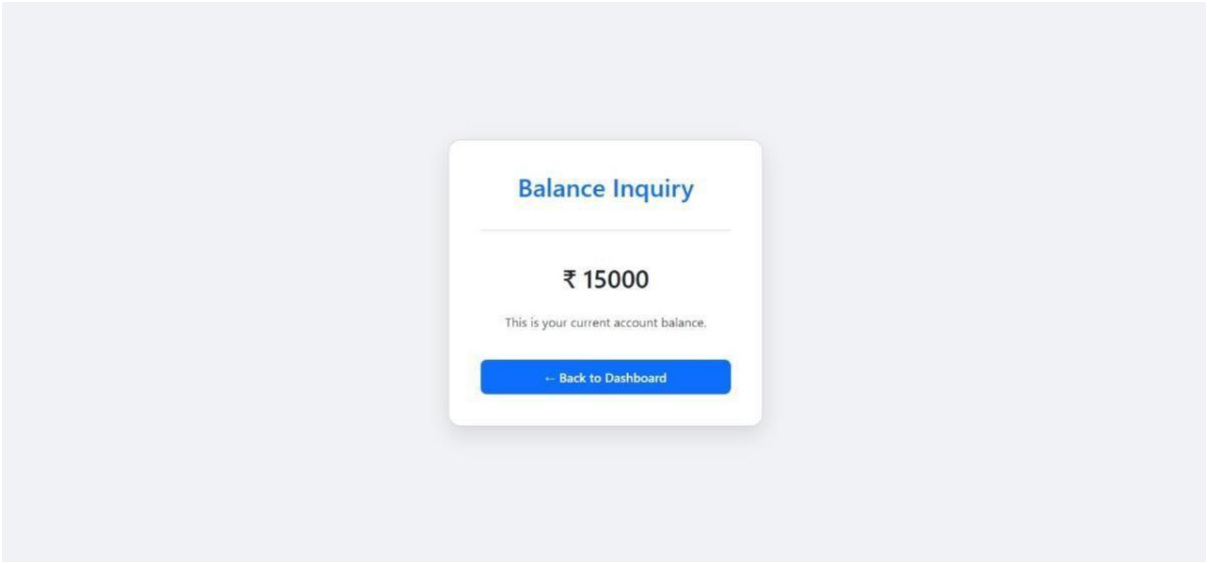



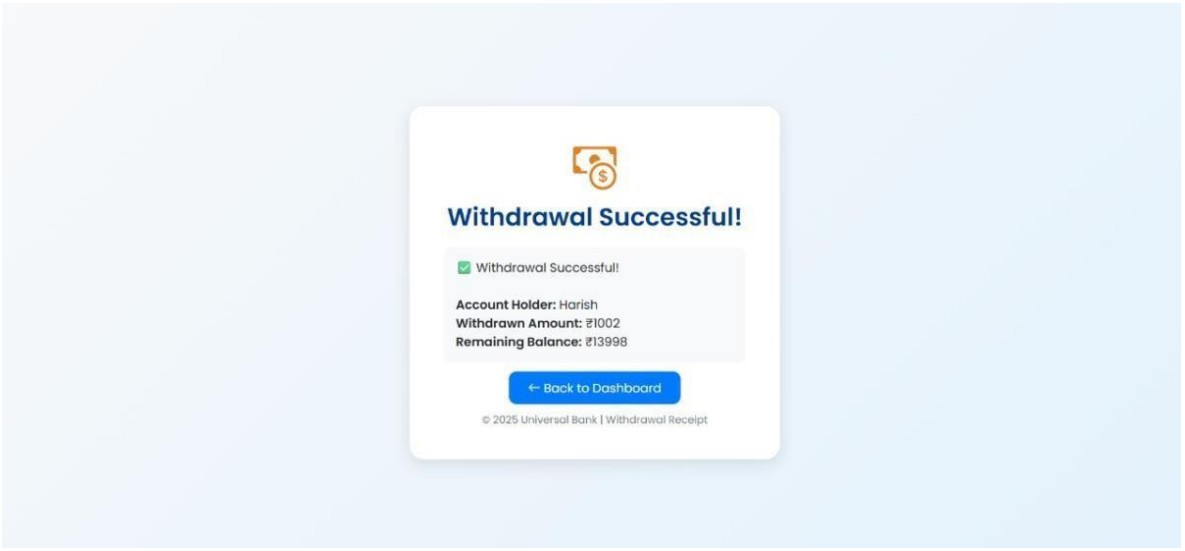

Universal Bank
Account Holder Login
User ID / Account No.
240701174
Password
Secure Login
OR
Bank Authority / Administrator Login
Forgot Password? | Register | Security Tips

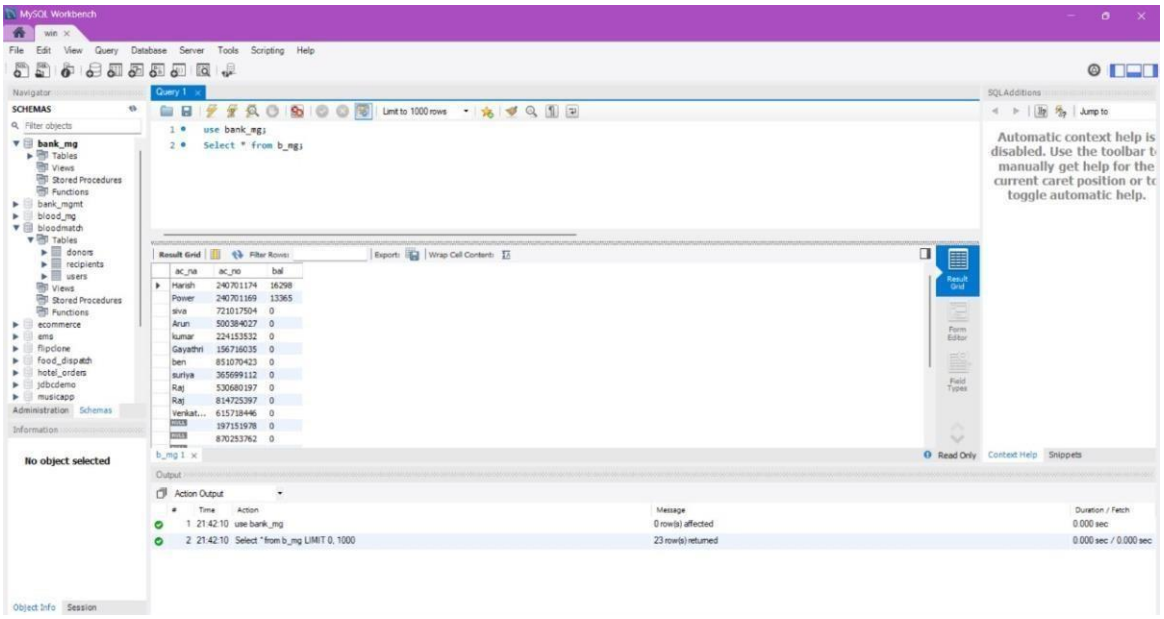

Universal Bank | Branch Operations
Deposit Amount
Account Number
240701174
Amount (₹)
13342
Proceed to Deposit
← Back to Dashboard
© 2025 Universal Bank | Secure Employee Portal v5.2
Burp Suite Community Edition


Deposit Successful!
Deposit Successful!
Account Holder: Harish
Deposited Amount: ₹2300
Updated Balance: ₹16298
← Back to Dashboard
© 2025 Universal Bank | Transaction Receipt


Universal Bank | Branch Operations
Balance Inquiry
Account Number
240701174
Phone Number
7200868675
Verify Account
← Back to Dashboard
© 2025 Universal Bank | Secure Employee Portal v5.2


Balance Inquiry
₹ 15000
This is your current account balance.
← Back to Dashboard


Withdrawal Successful!
Withdrawal Successful!
Account Holder: Harish
Withdrawn Amount: ₹1002
Remaining Balance: ₹13998
← Back to Dashboard
© 2025 Universal Bank | Withdrawal Receipt


MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
bank_mg
Tables
Views
Stored Procedures
Functions
bank_mgmt
blood_mg
bloodmatch
Tables
donors
recipients
users
Views
Stored Procedures
Functions
ecommerce
ems
flipclone
food_dispatch
hotel_orders
jdbcdemo
musicapp
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
1 use bank_mg;
2 Select * from b_mg;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
ac_na ac_no bal
Harish 240701174 16298
Power 240701169 13365
siva 721017504 0
Arun 500384027 0
kumar 224153532 0
Gayathri 156716035 0
ben 851070423 0
suriya 365699112 0
Raj 530680197 0
Raj 814725297 0
Venkat... 615718446 0
197151978 0
870253762 0
b_mg 1
Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
# Time Action Message Duration / Fetch
1 21:42:10 use bank_mg 0 row(s) affected 0.000 sec
2 21:42:10 Select * from b_mg LIMIT 0, 1000 23 row(s) returned 0.000 sec / 0.000 sec

# Conclusion:

- Bank management system is a virtualization of transactions in banking system.
- The banking system uses manual working but when we use a bank management system it is totally virtual and this process avoid manual work and converts it in automatic process.
- Bank management system is saving the time with accuracy than traditional bank system.
- It also reduces the human errors causeD

# FUTURE ENHANCEMANT:

- This project aspires to be a simulation of the Banking system for banks and also reduces the human errors caused by employees or the customer itself. • If coupled with appropriate hardware this system can be turned into an ATM software.d by employees or the customer itself.

# REFERENCES

**Websites**


- https://www.javatpoint.com/online-banking-project
- https://projectsgeek.com/2016/02/complete-banking-system-java-project.html
- https://www.apachefriends.org/index.html


**Software / Tools Used**

- MySQL Workbench (Database Design & Query Execution)
- NetBeans / Eclipse (Java Programming & GUI)
- XAMPP (Local Server Environment)
- Draw.io / Lucidchart (ER Diagram Design)