BANKING SYSTEM

A PROJECT REPORT

Submitted by

MD. ASIF KHAN

MAHBOOB ALAM

Under the Supervision of

Dr. JEELANI

in partial fulfillment for the award of the degree

of

Master of Computer Application

DEPARTMENT OF COMPUTER APPLICATION

INTEGRAL UNIVERSITY, LUCKNOW

JUNE 2023

INTEGRAL UNIVERSITY, LUCKNOW

CERTIFICATE

Certified that this project report **“BANKING SYSTEM”** is the bonafide work of “**MD. ASIF KHAN** and **MAHBOOB ALAM”** who carried out the project work under my supervision.

Dr. Jeelani

Project Guide

Deptt. of Computer Application

Integral University, Lucknow

INTEGRAL UNIVERSITY, LUCKNOW

CERTIFICATE

Certified that this project report **"BANKING SYSTEM"** is the bonafide work of "**MD. ASIF KHAN** and **MAHBOOB ALAM"** who have successfully carried out the project.

Dr. Md. Faizan Farooqui	**Prof. Mohammad Faisal**
Project Coordinator	**Head**
Deptt. of Computer Application	Deptt. of Computer Application
Integral University, Lucknow	Integral University, Lucknow

DECLARATION

I/We hereby declare that this submission is my/our own work and that, to the best of my/our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date:

(MD. ASIF KHAN)
Enrolment No: 1800102795

(MAHBOOB ALAM)
Enrolment No: 1800103957

ACKNOWLEDGEMENT

This is a great opportunity to acknowledge and thank all those persons without whose support and help this project would have been impossible. Wewould like to add a few heartfelt words for the people who were part of this project in numerous ways.

I would like to thank my project guide **Dr. JEELANI,** for his indefatigable guidance, valuable suggestions, moral support, constant encouragement and contribution of time to the successful completion of projectwork. I am very grateful to him, for providing all the facilities needed during the project development. At the outset, I sincerely thank all faculty members of my institution for their extra effort to make our session online inspire all ideas.

I thank my Counselor for his indispensable support and encouragement throughout the project. I must also thank to respect **Dr. MD. FAIZAN FAROOQUI** for his valuable suggestion while working on the project. I would like to thank all those who helped me directly or indirectly.

Last but not least, I would like to acknowledge the ongoing support ofmy parents and my family members, whose patience and encouragement during these long days and nights have been paramount in making this project a reality.

Thank you

ABSTRACT

This project is aimed at developing an Online Banking for customer. The system is an online application that can be accessed throughout the organization and outside as well with proper login provided.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of Oracle 10g and all the user interfaces have been designed using the JAVA. The database connectivity is planned using the "Database" methodology. The standards of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology, in mind. The specification has been normalized up to 3NF to eliminate all the anomalies that may arise due to the database transaction that are executed by the general users and the organizational administration. The internal database has been selected as Oracle 10g.The basic constructs of table spaces, clusters and indexes have been exploited to provide higher consistency and reliability for the data storage. The Oracle 10g was a choice as it provides the constructs of high-level reliability and security. The total front end was dominated using the HTML 5. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations. The database connectivity was planned using the latest "Database connection" technology provided by Oracle. The authentication and authorization were crosschecked at all the relevant stages. The user level accessibility has been restricted into two zones namely.

TABLE OF CONTENTS

LIST OF SYMBOLS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Introduction

A Banking Management System is a software application that enables banks to manage and streamline their day-to-day operations efficiently. It helps in automating tasks such as customer account management, deposit management, loan management, and transaction management. A Banking Management System can be a standalone application or a web-based application that can be accessed by bank employees and customers through the internet.

A Banking Management System major project involves designing and developing a comprehensive software solution that caters to the various needs of a bank. The project usually involves a team of developers, business analysts, and project managers who work together to create a solution that meets the bank's requirements.

The main objectives of a Banking Management System major project are to provide a secure, reliable, and scalable solution that can handle the demands of a modern banking system. The system should be designed to improve operational efficiency, reduce costs, and enhance customer satisfaction. Some of the key features of a Banking Management System major project include:

1. Account Management: The system should allow bank employees to create, manage and maintain customer accounts efficiently.
2. Deposit and Loan Management: The system should enable banks to manage customer deposits and loans effectively.
3. Transaction Management: The system should support various types of transactions such as online payments, fund transfers, and check processing.
4. Customer Relationship Management: The system should enable banks to maintain a good relationship with their customers by providing personalized and efficient service.
5. Reporting and Analytics: The system should generate reports and provide insights into the bank's operations and performance.

1.2. Objective of the project

A Computer based management system is designed to handle all the primary information required to calculate monthly statements of customer account which include monthly statement of any month. Separate database is maintained to handle all the details required for the correct statement calculation and generation.

This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updation can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

The main objective of our project is providing the different typed of customers facility, the main objective of this system is to find out the actual customer service. Etc.

- It should fulfill almost all the process requirements of any Bank.
- It should increase the productivity of bank by utilizing the working hours more and more, with minimum manpower.

This project includes the entire upgraded feature required for the computerization banking system. This system is very easy to use, so that any user can use without getting pre-knowledge about this. Its very much user friendly and meet almost all daily working process requirements. This system is completely GUI based and can be use by mouse and as well as keyboard. This system is melded in such a way that has got all features to upgrade without making much change in existing components.

1.3. Hardware Requirements and Software Requirements

There are two types of requirements as follows:

1.3.1. Hardware Requirements

Hardware is a set of physical components, which performs the functions of applying appropriate, predefined instructions. In other words, one can say that electronic and mechanical parts of computer constitute hardware.

- **Processor:** Intel corei5 or higher.
- **RAM:** 8GB or higher.
- **Hard Disk Space:** 500GB or higher.
- **Display:** 1366 x 768 resolution or higher.
- **Internet Connectivity:** Broadband or high-speed internet.

1.3.2. Software Requirements

The software is a set of procedures of coded information or a program which when fed into the computer hardware, enables the computer to perform the various tasks. Software is like a current inside the wire, which cannot be seen but its effect can be felt.

- **Operating System:** Windows 10 or higher.
- **Java Development Kit (JDK):** 8 or higher.
- **Integrated Development Environment (IDE):** Eclipse, NetBeans or IntelliJ IDEA.
- **Web Server:** Apache Tomcat 8 or higher.
- **Database Server:** MYSQL or Oracle.
- **JavaScript Libraries:** jQuery, Bootstrap and AngularJS.

2. PROBLEM IDENTIFICATION &FEASIBILITY STUDY

2.1. Problem Definition

Before starting a major banking system project, it is essential to identify the problems faced by the bank and its customers that the project aims to solve. Some of the common problems faced by banks include:

1. Inefficient manual processes: Many banks still rely on manual processes, which are time-consuming and error-prone, leading to delays and mistakes.
2. Lack of automation: Some banks do not have a centralized system for managing customer data, account information, and transactions, which makes it difficult to track and analyze data effectively.
3. Security concerns: As the banking industry moves towards digitalization, security threats such as cyber-attacks and data breaches have become more prevalent, leading to a loss of customer trust and financial losses.
4. Poor customer experience: Banks may struggle to provide a seamless and efficient customer experience due to a lack of modern technology and outdated systems.

2.2. Feasibility Study

Understanding Feasibility: Feasibility study means the analysis of problem to determine if It can be solved effectively. In other words, it is the study of the possibilities of the proposed system it studies the work ability, impact on the organization ability to meet user's need and efficient use of resources. Three aspects in which the system has to be feasible are: -

Economic Feasibility: The economic analysis checks for the high investment incurred on the system. It evaluates development & implementing charges for the proposed "Banking Project". The S/W used for the development is easily available at minimal cost & the database applied is freely available hence it results in low-cost implementation.

Technical Feasibility: This aspect concentrates on the concept of using Computer Meaning, "Mechanization" of human works. Thus, the automated solution leads to the need for a technical feasibility study. The focus on the platform used database management &users for that S/W. The proposed system doesn't require an in-depth technical knowledge as the system development is simple and easy to understand. The S/W (VB.NET) used makes the system user friendly (GUI). The result obtain should be true in the real time conditions.

Behavioral Feasibility: Behavioral feasibility deals with the runtime performance of the S/W the proposed system must score higher than the present in the behavioral study. The S/W should have end user in mind when the system is designed while designing s/w the programmer should be aware of the conditions user's Knowledge input, output, calculations etc. The s/w contains only a minimum no. of bugs. Care should be also taken to avoid non-working means &t buttons.

3. RESOURCE REQUIREMENT

- **IDE:** NetBeans.
- **Front End:** JSP, SERVLET, JDBC, JavaScript.
- **Programming Language:** JAVA.
- **Back End:** MySQL.

3.1. Project Description

A banking system major project is a comprehensive software system that allows banks to efficiently manage customer accounts and transactions. The system typically includes several components, including customer interface, account management, transaction processing, and security features.

The customer interface component provides a user-friendly platform for customers to access their account information and perform various transactions such as opening savings or current accounts, making deposits, withdrawals, transferring funds, paying bills, applying for loans, and checking transaction history. The account management component manages account data and updates account balances based on customer transactions. It also includes features for updating account information, changing passwords, and setting up automatic payments.

Transaction processing is an important component of the banking system major project as it ensures that customer transactions are accurately recorded and updated in the system. This component includes features for verifying account information, checking account balances, and processing transactions such as deposits, withdrawals, transfers, and bill payments.

Security features are also an essential component of the banking system major project as they protect customer data and prevent unauthorized access to the system. Security features may include user authentication, encryption of sensitive data, and monitoring of user activity.

Overall, a banking system major project requires a thorough understanding of the project requirements, careful planning and design, effective implementation, and rigorous testing. Successful implementation of the project can provide numerous benefits to both the bank and its customers, including increased efficiency, improved customer satisfaction, and reduced costs.

3.2. Module Description

3.2.1 ***Create Account Module:*** This module allows bank employees to create new customer accounts in the system. It typically includes features for capturing customer details such as name, address, contact information, and identification documents. The module also generates unique account numbers and handles the verification and approval of new accounts.

3.2.2. Transfer Funds Module: This module allows customers to transfer money between accounts. It includes features for selecting the source and destination accounts, entering the transfer amount, and verifying the transaction details before submitting it for processing. The module also handles the processing and verification of the transfer transaction, ensuring that the transfer is completed successfully.

3.2.3. ***Deposit Amount Module*****:** This module allows customers to deposit money into their accounts. It typically includes features for selecting the account to deposit the money into, entering the deposit amount, and verifying the transaction details before submitting it for processing. The module also handles the processing and verification of the deposit transaction, ensuring that the deposit is completed successfully.

3.2.4. ***Withdraw Amount Module:*** This module allows customers to withdraw money from their accounts. It includes features for selecting the account to withdraw money from, entering the withdrawal amount, and verifying the transaction details before submitting it for processing. The module also handles the processing and verification of the withdrawal transaction, ensuring that the withdrawal is completed successfully.

3.2.5. ***Account Balance Module:*** This module allows customers to view their account balance and transaction history. It includes features for selecting the account to view the balance, displaying the current balance, and providing access to transaction history, including deposits, withdrawals, and transfers.

Each of these modules is critical to the overall functionality of the banking management system, and they must be designed and implemented with the highest level of accuracy, security, and reliability to ensure the smooth operation of the bank's financial transactions.

3.3. SYSTEM DESIGN

3.3.1. Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.4. Normalization

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies. Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

- **Insertion anomaly**: Inability to add data to the database due to absence of other data.
- **Deletion anomaly**: Unintended loss of data due to deletion of other data.
- **Update anomaly**: Data inconsistency resulting from data redundancy and partial update
- **Normal Forms**: These are the rules for structuring relations that eliminate anomalies.

3.4.1. First Normal Form

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

3.4.2. Second Normal Form:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1) Primary key is a not a composite primary key
2) No non key attributes are present
3) Every non key attribute is fully functionally dependent on full set of primary key.

3.4.3. Third Normal Form

A relation is said to be in third normal form if their exits no transitive dependencies. **Transitive Dependency**: If two non-key attributes depend on each other as well as on the primary key then they are said to be transitively dependent. The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

3.5. E-R Diagrams

ER-modeling is a data modeling technique used in software engineering to produce a conceptual data model of a information system. Diagrams created using this ER-modeling technique are called Entity-Relationship Diagrams, or ER diagrams or ERDs. So you can say that Entity Relationship Diagrams illustrate the logical structure of databases.
Dr. Peter Chen is the originator of the Entity-Relationship Model. His original paper about ER-modeling is one of the most cited papers in the computer software field. Currently the ER model serves as the foundation of many system analysis and design methodologies, computer-aided software engineering (CASE) tools, and repository systems.

The original notation for ER-Diagrams uses rectangles to represent entities, and diamonds to represent relationships.

There are three basic elements in ER-Diagrams:

- Entities are the "things" for which we want to store information. An entity is a person, place, thing or event.
- Attributes are the data we want to collect for an entity.
- Relationships describe the relations between the entities.

ERDs show entities in a database and relationships between tables within that database. It is essential to have ER-Diagrams if you want to create a good database design. The diagrams help focus on how the database actually works.

Entity (Instance): An instance of a physical object in the real world.

Entity Class: Group of objects of the same type

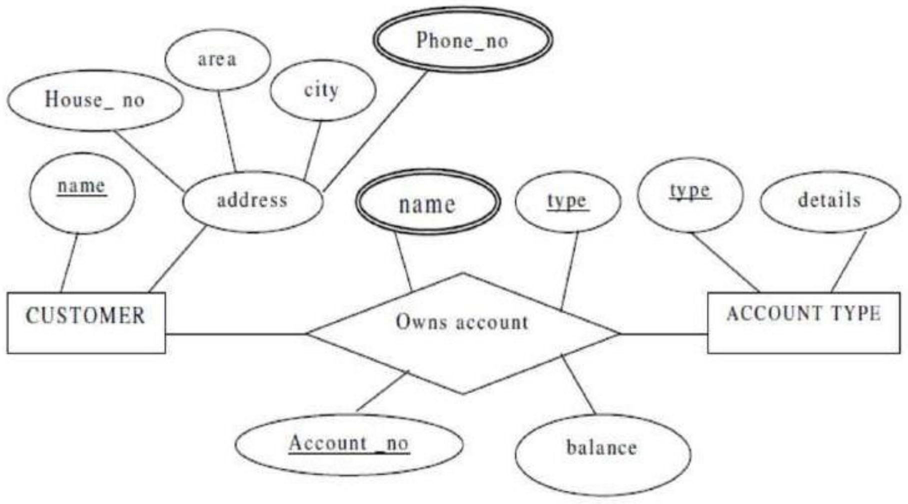
E.g., Entity Class “Student”, Entities “John”, “Trish” etc

Attributes: Properties of Entities that describe their characteristics.

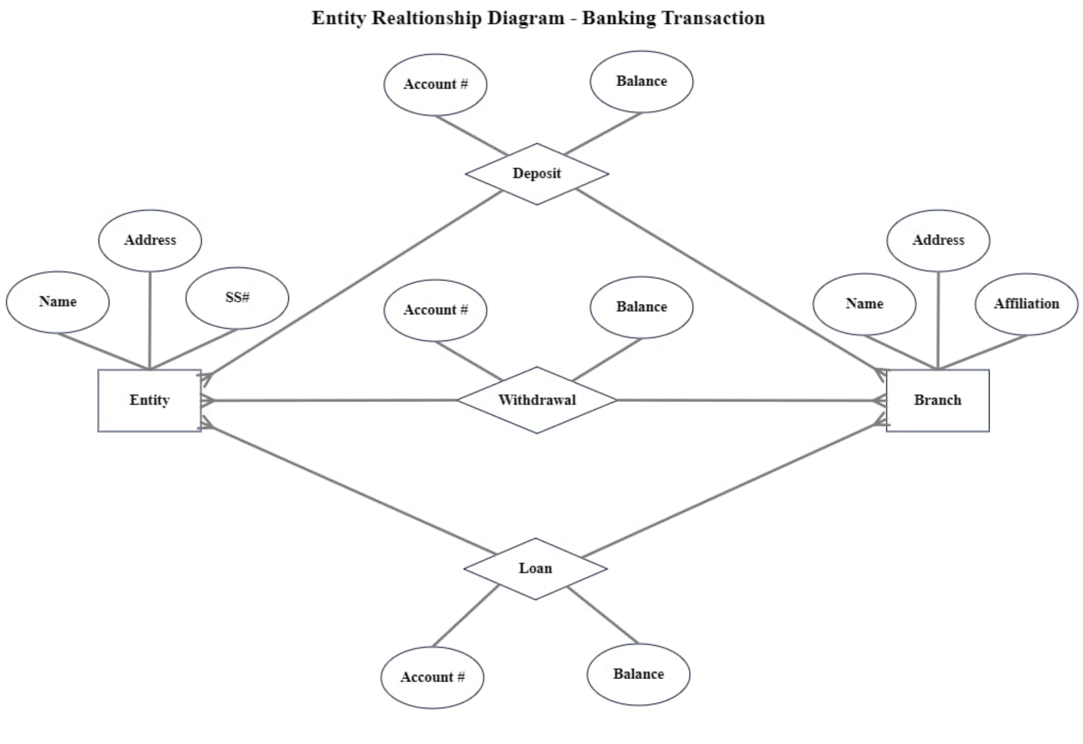
Types:

1. **Simple:** Attribute that is not divisible, e.g., age.
2. **Composite**: Attribute composed of several simple attributes, **e.g.,** address (house number, street, district)
3. **Multiple:** Attribute with a set of possible values for the same entity, e.g., Phone (home, mobile etc.) or email
4. **Key:** Uniquely Ids the Entity e.g., PPSN, Chassis No. Each simple attribute associated with a VS that may be assigned to that attribute for each individual entity, e.g., age = integer

E-R DIAGRAM FOR CREATE ACCOUNT



E-R DIAGRAM FOR DEPOSITE AND WITHDRAW



3.6. RULES GOVERNING THE DFD'S

3.6.1. Process

- No process can have only outputs.
- No process can have only inputs. If an object has only inputs than it must be a sink.
- A process has a verb phrase label.

3.6.2. Data Store

- Data cannot move directly from one data store to another data store, a process must move data.
- Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
- A data store has a noun phrase label.

3.6.3. Source Or Sink

The origin and /or destination of data.

- Data cannot move direly from a source to sink it must be moved by a process.
- A source and /or sink has a noun phrase land.

3.6.4. Data Flow

- A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated however by two separate arrows since these happen at different type.
- A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
- A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
- A Data flow to a data store means update (delete or change).
- A data Flow from a data store means retrieve or use.

3.7. Data Flow Diagram

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Game and Sarson notation develops the data flow diagrams. Each component in a DFD is labelled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower-level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

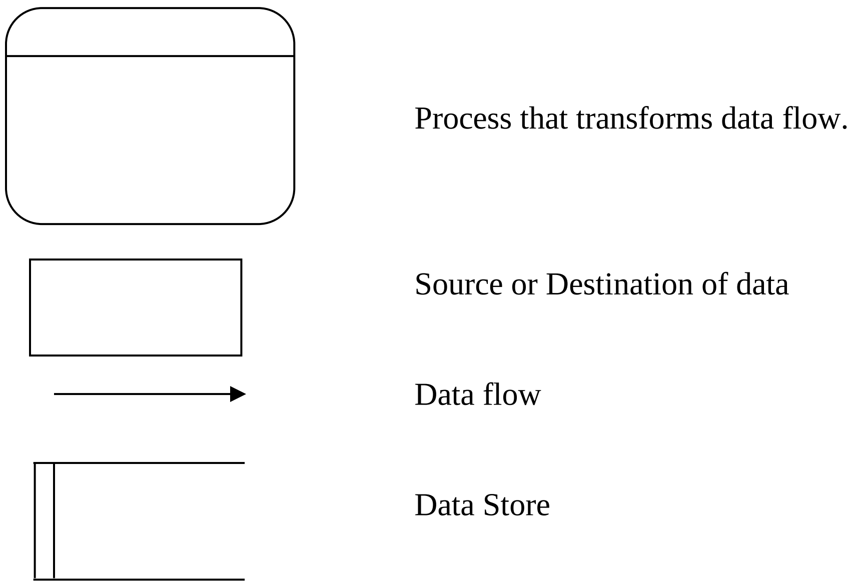
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this led to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

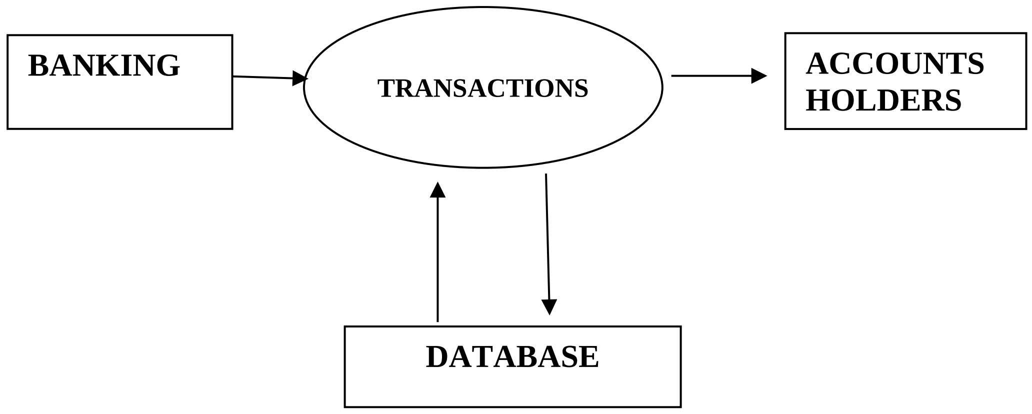
3.7.1. DFD Symbols

In the DFD, there are four symbols

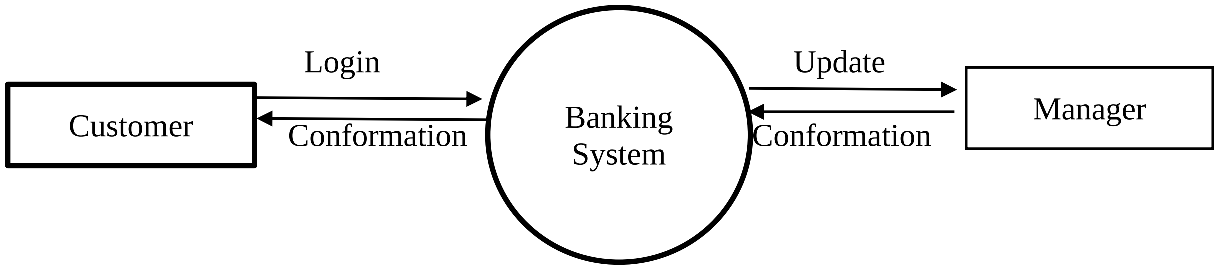
1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data


Process that transforms data flow.
Source or Destination of data
Data flow
Data Store

SYSTEM DATA FLOW DIAGRAM

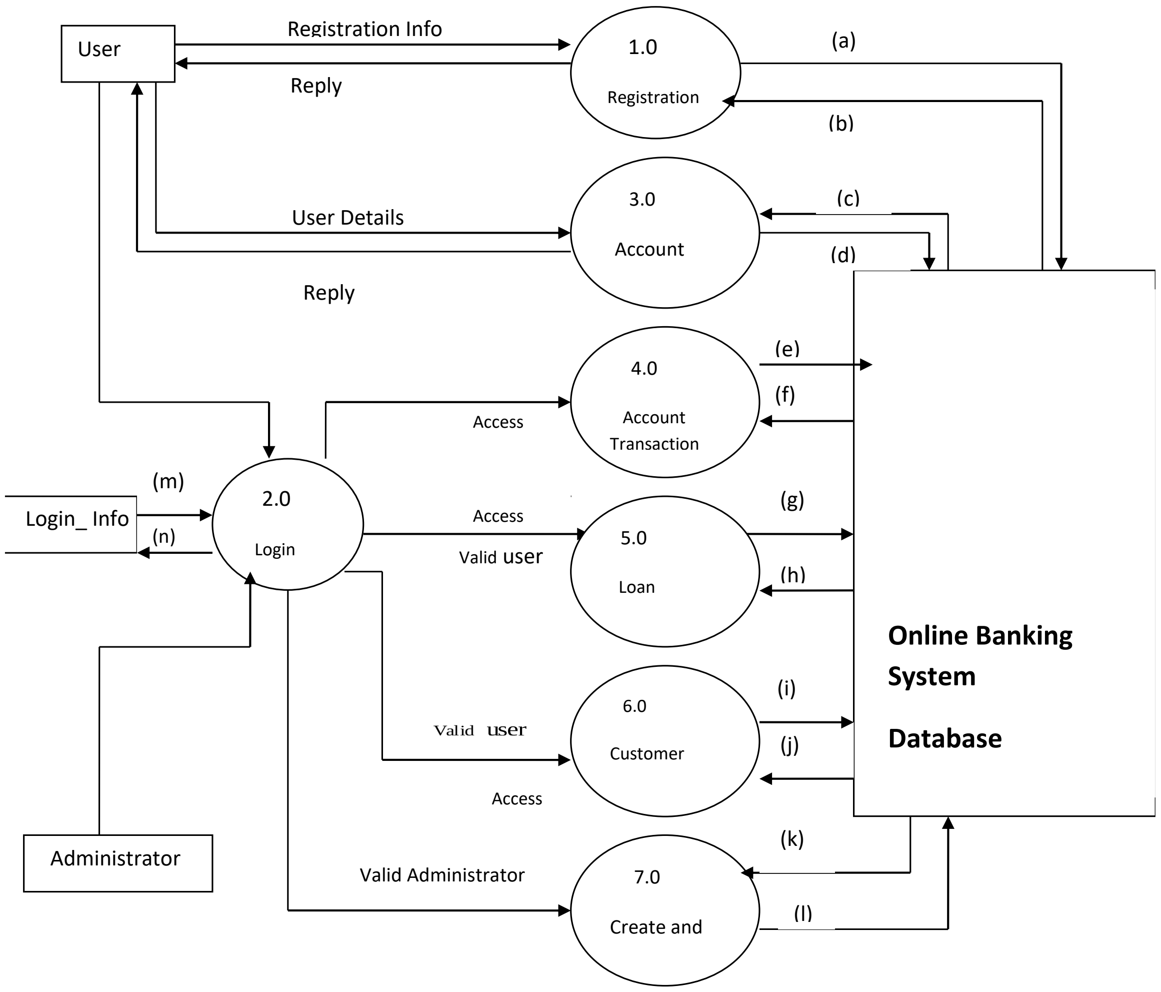


0 Level DFD



Explanation of the diagram:

- The main component is the "Banking System."
- The system interacts with two types of users: "User" and "Admin."
- Users can perform actions such as "Create Account," "Login," and "Perform Actions."
- Admins can perform actions such as "Delete User" and "Login."
- The "Perform Actions" block represents various transactional activities like deposit, withdraw, and fund transfer.
- Both users and admins interact with the "Database" component, which stores the user and transaction information.



(a): User Details

(b): Response

(c): Personal details

(d): Reply

(e): Account Transaction entry

(f): Transaction Details

(g): Loan Application

(h): Response

Level 1 DFD

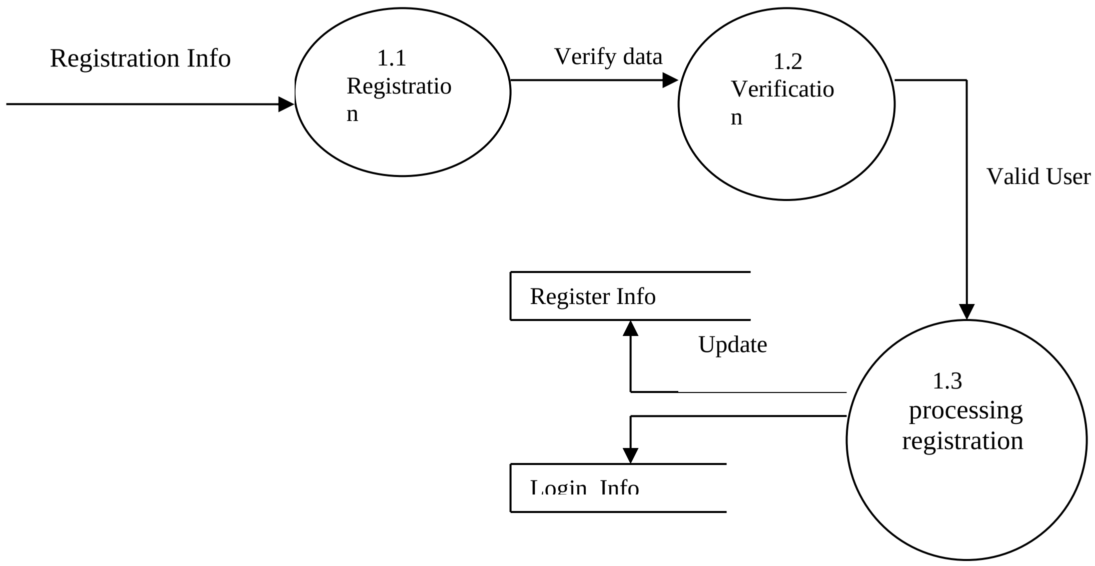


Fig. Level 2 DFD process-1

4. DATABASE

A database is a structured collection of data organized and stored in a computer system. It is designed to efficiently store, manage, retrieve, and update vast amounts of information. A database serves as a central repository for storing data that can be accessed and manipulated by various users or applications.

In a database, data is organized into tables, which consist of rows and columns. Each row represents a record or an instance of data, while each column represents a specific attribute or characteristic of the data. The tables are interrelated through relationships established based on common data elements, known as keys.

The main purpose of a database is to provide a reliable, consistent, and secure means of storing and retrieving data. It offers several advantages over traditional file-based systems, such as data integrity, data independence, efficient data access, and concurrent data sharing.

A Database Management System (DBMS) is software that facilitates the creation, organization, and management of databases. It provides tools and functionalities for defining the structure of the database, specifying data types, enforcing data integrity rules, querying the data using a standardized language (e.g., SQL), and controlling user access to the data.

Databases are widely used in various industries and applications, including business operations, e-commerce, finance, healthcare, telecommunications, and more. They play a crucial role in data-driven decision-making, information storage, and retrieval, as well as supporting complex data processing tasks.

4.1. Introduction to SQL

SQL is a standard computer language for accessing and manipulating databases.

- SQL stands for **S**tructured **Q**uery **L**anguage.
- SQL allows you to access a database.
- SQL is an ANSI standard computer language.

- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can insert new records in a database.
- SQL can delete records from a database.
- SQL can update records in a database.
- SQL is easy to learn.

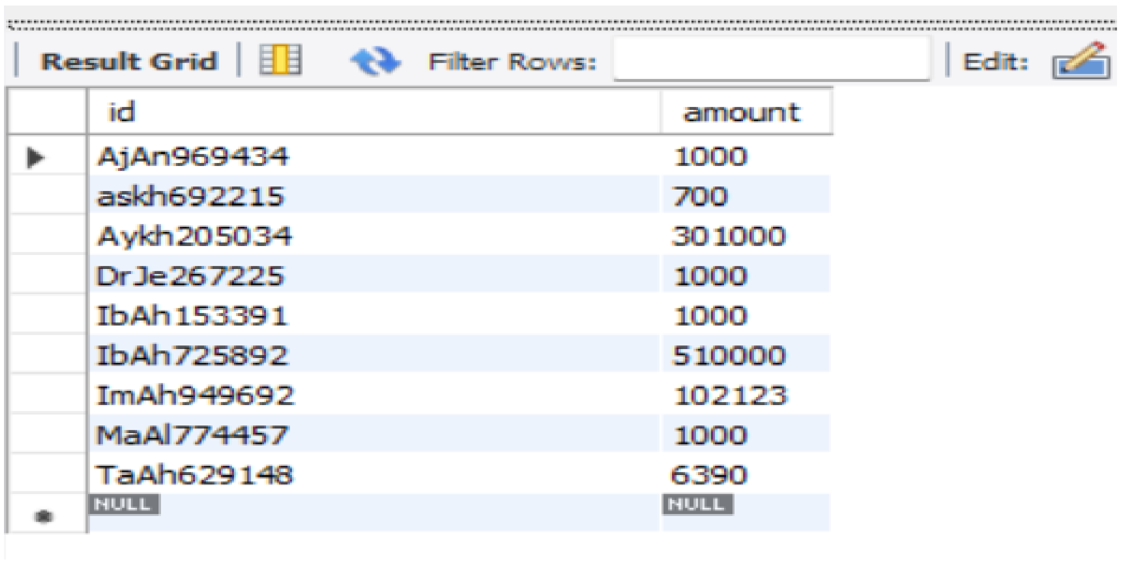
- SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc.
- Unfortunately, there are many different versions of the SQL language, but to be in compliance with the ANSI standard; they must support the same major keywords in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others).

4.2. Database Table:

Table 1: Create account table

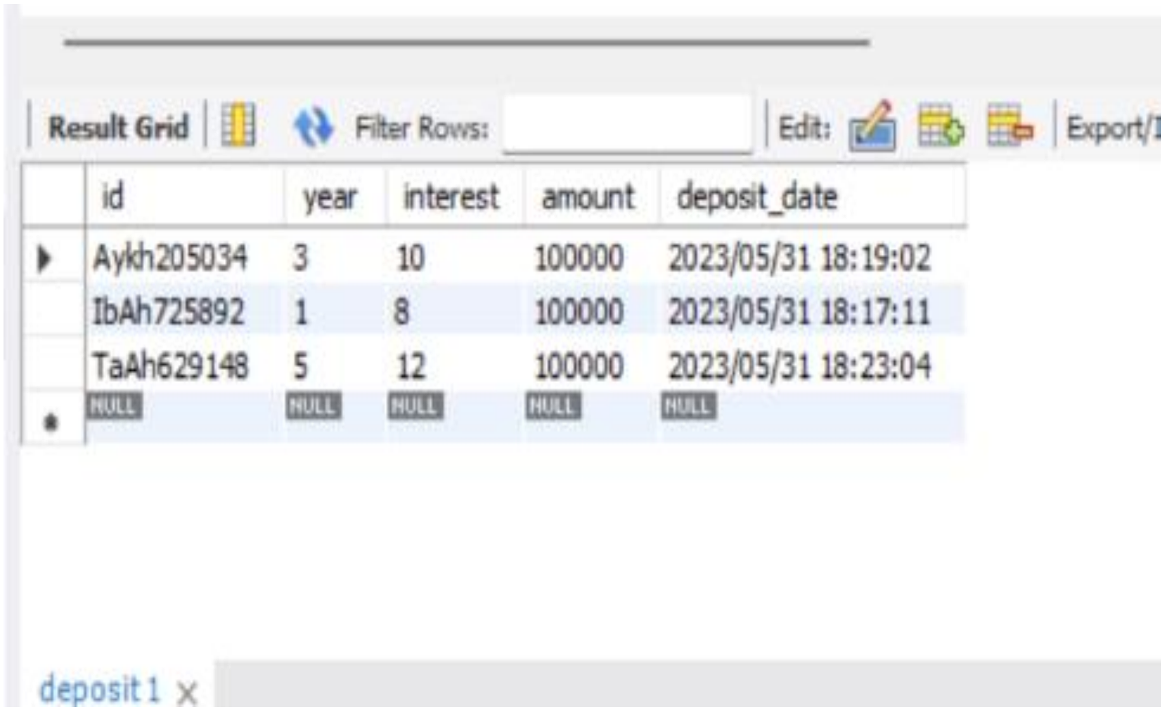
Column name	Data type	Nullable	Primary key
ACCOUNTINFO	Number	No	Yes
USERNAME	Varchar2	Yes	No
PASSWORD	Varchar2	Yes	No
AMOUNT	Varchar2	Yes	No
ADDRESS	Varchar2	Yes	No
PHONE	Varchar2	Yes	No

Table 2: Amount Table



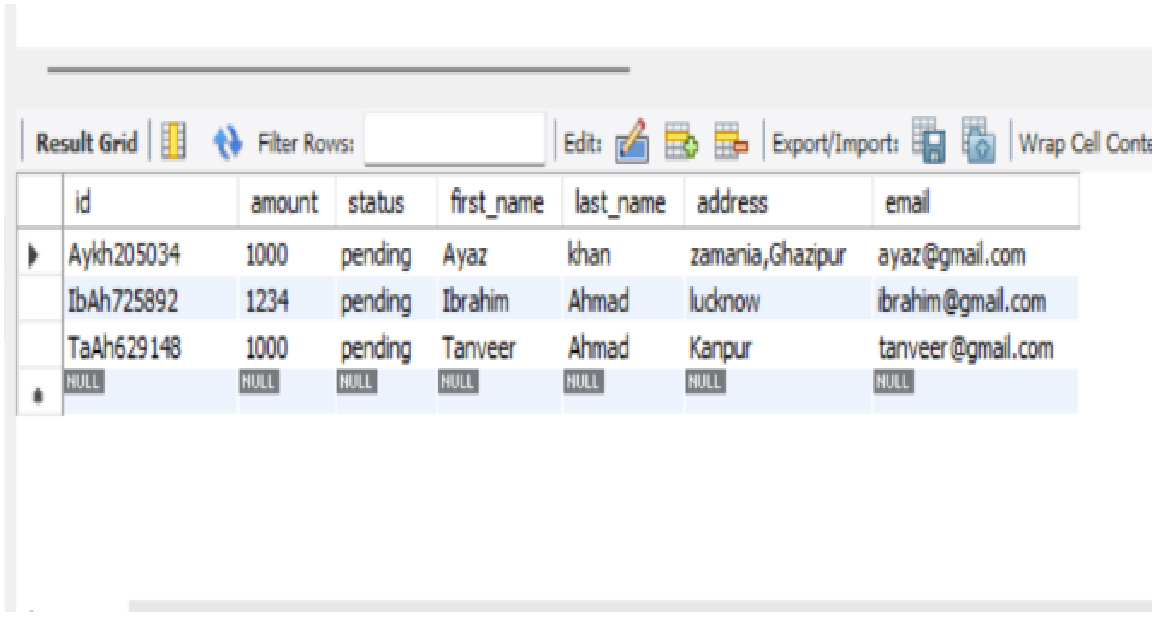
id	amount
AjAn969434	1000
askh692215	700
Aykh205034	301000
DrJe267225	1000
IbAh153391	1000
IbAh725892	510000
ImAh949692	102123
MaAl774457	1000
TaAh629148	6390
NULL	NULL

Table 3: Deposit Table



id	year	interest	amount	deposit_date
Aykh205034	3	10	100000	2023/05/31 18:19:02
IbAh725892	1	8	100000	2023/05/31 18:17:11
TaAh629148	5	12	100000	2023/05/31 18:23:04
NULL	NULL	NULL	NULL	NULL

Table 4: Loan table



id	amount	status	first_name	last_name	address	email
Aykh205034	1000	pending	Ayaz	khan	zamania,Ghazipur	ayaz@gmail.com
IbAh725892	1234	pending	Ibrahim	Ahmad	lucknow	ibrahim@gmail.com
TaAh629148	1000	pending	Tanveer	Ahmad	Kanpur	tanveer@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Chapter V

5. OUTPUTS AND CODING

CreateAccountServlet

```
package  com.online_banking;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;

import com.online_banking.model.AccountModel;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class CreateAccountServlet extends HttpServlet {
        String account_no, first_name, last_name, address, city, branch, zip, username, password,
re_password,
                        phone_number, email, account_type;
        String amount;

    @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                PrintWriter out = response.getWriter();

                first_name = request.getParameter("first_name");
                last_name = request.getParameter("last_name");
                address = request.getParameter("address");
                city = request.getParameter("city");
                branch = request.getParameter("branch");
                zip = request.getParameter("zip");
                username = request.getParameter("username");
                password = request.getParameter("password");
                re_password = request.getParameter("re_password");
                phone_number = request.getParameter("phone");
                email = request.getParameter("email");
                account_type = request.getParameter("account_type");
                amount =request.getParameter("amount");

                // Generating account number
                Random rand = new Random();
                int random_num = 100000 + rand.nextInt(999999);
```

```
			account_no = first_name.substring(0, 2) + last_name.substring(0, 2) +
random_num;
			System.out.println(account_no);

			//Getting Current date
			DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
			String reg_date = df.format(new Date());

			// Setting all variables to model class
		AccountModel am = new AccountModel();
			am.setAccount_no(account_no);
			am.setFirst_name(first_name);
			am.setLast_name(last_name);
			am.setAddress(address);
			am.setCity(city);
			am.setBranch(branch);
			am.setZip(zip);
			am.setUsername(username);
			am.setPassword(password);
			am.setPhone_number(phone_number);
			am.setEmail(email);
			am.setAccount_type(account_type);
			am.setAmount(amount);
			am.setReg_date(reg_date);

			if (password.equals(re_password)) {
				request.setAttribute("Account_details", am);
			    RequestDispatcher rd =
request.getRequestDispatcher("create_account_progress.jsp");
				rd.forward(request, response);

			} else {
				request.setAttribute("not_match", "yes");
				RequestDispatcher rd =
request.getRequestDispatcher("create_account.jsp");
				rd.forward(request, response);
			}
	}

}
```

DepositSchemeServlet

```
package com.online_banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;

import com.online_banking.database.DatabaseOperations;
import com.online_banking.database.JDBC_Connect;
import com.online_banking.model.AccountModel;
import com.online_banking.model.DepositSchemeModel;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;



public class DepositSchemeServlet extends HttpServlet {
		String account_no, deposit_amount, value;
		int year, interest_rate, amount;


	@Override
		protected void doPost(HttpServletRequest request, HttpServletResponse response)
					throws ServletException, IOException {
				account_no = request.getParameter("account_no");
				year = Integer.parseInt(request.getParameter("year"));
				interest_rate = Integer.parseInt(request.getParameter("interest_rate"));
				deposit_amount = request.getParameter("deposit_amount");
				value = request.getParameter("value");

				if (deposit_amount.equals("1,00,000&#2547;")) {
					amount = 100000;
				} else if (deposit_amount.equals("3,00,000&#2547;")) {
					amount = 300000;

				} else if (deposit_amount.equals("5,00,000&#2547;")) {
					amount = 500000;
				}

					DepositSchemeModel dpModel = new DepositSchemeModel();
				dpModel.setAccount_no(account_no);
				dpModel.setYear(year);
				dpModel.setInterest_rate(interest_rate);
				dpModel.setAmount(amount);
				dpModel.setValue(value);

				try {
							JDBC_Connect connect = new JDBC_Connect();
					Connection conn = connect.getConnection();
							DatabaseOperations operations = new DatabaseOperations();

							AccountModel am = operations.getAccountDetails(conn,
account_no);

					if (Integer.parseInt(am.getAmount()) >= amount) {
						int main_amount  = Integer.parseInt(am.getAmount() )- amount;
						PreparedStatement ps = conn.prepareStatement("update amount
set amount=? where id= ?");
```

```
					ps.setInt(1, main_amount);
					ps.setString(2, account_no);
					ps.executeUpdate();

					boolean allRight = operations.insertDepositScheme(dpModel);
					request.setAttribute("DepositScheme", dpModel);
					request.setAttribute("allRight", allRight);
							RequestDispatcher rd =
request.getRequestDispatcher("deposit_scheme_progress.jsp");
					rd.forward(request, response);

				} else {
					request.setAttribute("Not_Enough", "Yes");
					RequestDispatcher rd =
request.getRequestDispatcher("single_deposit_scheme.jsp?value=" + value);
					rd.forward(request, response);
				}

			} catch (Exception e) {
				e.printStackTrace();
			}
		}

}
```

DepositServlet

```
package com.online_banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;



import com.online_banking.database.JDBC_Connect;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DepositServlet extends HttpServlet {
	String account_no, username, password;
	Connection conn;
	Statement stmt;
	boolean pass_wrong = false;
```

```
        int current_amount, deposit_amount;

    @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                account_no = request.getParameter("account_no");
                username = request.getParameter("username");
                password = request.getParameter("password");
                deposit_amount = Integer.parseInt(request.getParameter("amount"));

                try {
                        JDBC_Connect connect = new JDBC_Connect();
                        conn = connect.getConnection();

                        stmt = conn.createStatement();

                        ResultSet rs = stmt.executeQuery("select * from account where id='" +
account_no + "' and username='" + username
                                        + "' and password='" + password + "'");

                        if (!rs.isBeforeFirst()) {
                                request.setAttribute("isPassOK", "No");
                             RequestDispatcher rd =
request.getRequestDispatcher("deposit.jsp");
                                rd.forward(request, response);
                        } else {
                                System.out.println("I am in");
                                ResultSet rs1 = stmt.executeQuery("select * from amount where
id ='" + account_no + "'");

                                while (rs1.next()) {
                                        current_amount = rs1.getInt(2);

                                        System.out.println(current_amount);
                                }

                                current_amount += deposit_amount;

                                PreparedStatement ps = conn.prepareStatement("update amount
set amount=? where id= ?");
                                ps.setInt(1, current_amount);
                                ps.setString(2, account_no);
                                ps.executeUpdate();

                                conn.close();

                                RequestDispatcher rd =
request.getRequestDispatcher("Deposit_process.jsp");
                                rd.forward(request, response);

                        }
                } catch (SQLException e) {

               System.out.println(e.getMessage());
               e.printStackTrace();
                } catch (Exception ex) {
         Logger.getLogger(DepositServlet.class.getName()).log(Level.SEVERE, null, ex);
      }
       }

}
```

JakartaRestConfiguration

```
package com.online_banking;

import jakarta.ws.rs.ApplicationPath;
import jakarta.ws.rs.core.Application;

/**
 * Configures Jakarta RESTful Web Services for the application.
 * @author Juneau
 */
@ApplicationPath("resources")
public class JakartaRestConfiguration extends Application {

}
```

LoanServlet

```
package com.online_banking;

import java.io.IOException;
import com.online_banking.database.DatabaseOperations;
import com.online_banking.model.LoanModel;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class LoanServlet extends HttpServlet {

    int loan_amount;
    String account_no, first_name, last_name, address, email;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        account_no = request.getParameter("account_no");
        loan_amount = Integer.parseInt(request.getParameter("loan_amount"));
        first_name = request.getParameter("first_name");
        last_name = request.getParameter("last_name");
        address = request.getParameter("address");
        email = request.getParameter("email");

        LoanModel lModel = new LoanModel();
        lModel.setAccount_no(account_no);
        lModel.setFirst_name(first_name);
        lModel.setLast_name(last_name);
        lModel.setAddress(address);
        lModel.setEmail(email);
        lModel.setLoan_amount(loan_amount);
        lModel.setStatus("pending");

        try {
            DatabaseOperations operations = new DatabaseOperations();
            boolean check = operations.insertLoanDetails(lModel);

            if (check) {
                RequestDispatcher rd = request.getRequestDispatcher("loan_process.jsp");
                request.setAttribute("loan_details", lModel);
                rd.forward(request, response);
            } else {
                RequestDispatcher rd = request.getRequestDispatcher("loan_process.jsp");
                request.setAttribute("error", "yes");
                rd.forward(request, response);
            }
        } catch (Exception e) {
        }
    }

}
```

LoginServlet

```
package com.online_banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import com.online_banking.database.JDBC_Connect;
import com.online_banking.model.AccountModel;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class LoginServlet extends HttpServlet {

    String UserName, password;
    Connection conn;
    Statement stmt;
    AccountModel am = null;
   boolean pass_wrong = false;

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        UserName = request.getParameter("UserName");
        password = request.getParameter("password");

        System.out.println(UserName);
        System.out.println(password);

        try {
            // Getting database connection
            JDBC_Connect connect = new JDBC_Connect();
            conn = connect.getConnection();
            stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery(
                    "select * from account where username ='" + UserName + "'" + "and password='" +
password + "'");

            if (!rs.isBeforeFirst()) {
                request.setAttribute("isPassOK", "No");
                RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
                rd.forward(request, response);
            } else {
                while (rs.next()) {
                    pass_wrong = true;
                    // Setting all variables to model class
                    am = new AccountModel();
                    am.setAccount_no(rs.getString(1));
                    am.setFirst_name(rs.getString(2));
                    am.setLast_name(rs.getString(3));
                    am.setAddress(rs.getString(4));
                    am.setCity(rs.getString(5));
                    am.setBranch(rs.getString(6));
                    am.setZip(rs.getString(7));
                    am.setUsername(rs.getString(8));
                    am.setPassword(rs.getString(9));
                    am.setPhone_number(rs.getString(10));
                    am.setEmail(rs.getString(11));
                    am.setAccount_type(rs.getString(12));
                    am.setReg_date(rs.getString(13));
                    ResultSet rs1 = stmt.executeQuery("select * from amount where id ='" +
am.getAccount_no() + "'");

                    while (rs1.next()) {
                        am.setAmount(rs1.getString(2));
                    }

                    // Setting Session variable for current User
                    HttpSession session = request.getSession();
                    session.setAttribute("userDetails", am);

                    RequestDispatcher rd = request.getRequestDispatcher("home.jsp");
                    rd.forward(request, response);

                }
            }

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception ex) {
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

TransferServlet

```
package com.online_banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.online_banking.database.JDBC_Connect;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class TransferServlet extends HttpServlet {
    String account_no, username, target_acc_no, password;
    Connection conn;
    PreparedStatement stmt, stmt1, stmt2, stmt3;
    int own_amount, transfer_amount, recipient_amount;
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        account_no = request.getParameter("account_no");
        username = request.getParameter("username");
        target_acc_no = request.getParameter("target_acc_no");
        password = request.getParameter("password");
        transfer_amount = Integer.parseInt(request.getParameter("amount"));

        try {
            JDBC_Connect connect = new JDBC_Connect();
            conn = connect.getConnection();

            stmt = conn.prepareStatement("select * from account where id=? and username=? and
password=?");
            stmt.setString(1, account_no);
            stmt.setString(2, username);
            stmt.setString(3, password);
            ResultSet rsOwn = stmt.executeQuery();

            stmt1 = conn.prepareStatement("select * from account where id=?");
            stmt1.setString(1, target_acc_no);
            ResultSet rstTarget = stmt1.executeQuery();

            if (!rsOwn.isBeforeFirst() && !rstTarget.isBeforeFirst()) {
                request.setAttribute("isPassOK", "No");
                RequestDispatcher rd = request.getRequestDispatcher("transfer.jsp");
                rd.forward(request, response);
            } else {
                System.out.println("I am in");

                try (PreparedStatement rs1 = conn.prepareStatement("select * from amount where id
=?")) {
                    rs1.setString(1, account_no);
                    ResultSet rs1Result = rs1.executeQuery();

                    while (rs1Result.next()) {
                        own_amount = rs1Result.getInt(2);
                    }
                }

                if (own_amount >= transfer_amount) {
                    own_amount -= transfer_amount;

                    try (PreparedStatement rs2 = conn.prepareStatement("select * from amount where id
=?")) {
                        rs2.setString(1, target_acc_no);
                        ResultSet rs2Result = rs2.executeQuery();

                        while (rs2Result.next()) {
                            recipient_amount = rs2Result.getInt(2);
                        }
                    }

                    recipient_amount += transfer_amount;

                    try (PreparedStatement ps = conn.prepareStatement("update amount set amount=?
where id= ?");
                         PreparedStatement ps1 = conn.prepareStatement("update amount set amount=?
where id= ?")) {

                        ps.setInt(1, own_amount);
                        ps.setString(2, account_no);
                        ps.executeUpdate();

                        ps1.setInt(1, recipient_amount);
                        ps1.setString(2, target_acc_no);
                        ps1.executeUpdate();
                    }

                    conn.close();

                    RequestDispatcher rd = request.getRequestDispatcher("transfer_process.jsp");
                    rd.forward(request, response);
                } else {
                    conn.close();
                    request.setAttribute("EnoughMoney", "No");
                    RequestDispatcher rd = request.getRequestDispatcher("transfer.jsp");
                    rd.forward(request, response);
                }

            }

        } catch (SQLException e) {
            System.out.println(e.getMessage());
        } catch (Exception ex) {
            Logger.getLogger(TransferServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

WithdrawServlet

```
package com.online_banking;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import com.online_banking.database.JDBC_Connect;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.util.logging.Level;
import java.util.logging.Logger;


public class WithdrawServlet extends HttpServlet {
        String account_no, username, password;
        Connection conn;
        Statement stmt;
        boolean pass_wrong = false;
        int current_amount, withdraw_amount;

    @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                account_no = request.getParameter("account_no");
                username = request.getParameter("username");
                password = request.getParameter("password");
                withdraw_amount = Integer.parseInt(request.getParameter("amount"));

                try {
                        JDBC_Connect connect = new JDBC_Connect();
                        conn = connect.getConnection();

                        stmt = conn.createStatement();

                        ResultSet rs = stmt.executeQuery("select * from account where id='" +
account_no + "' and username='"
                                        + username + "' and password='" + password + "'");

                        if (!rs.isBeforeFirst()) {
                                request.setAttribute("isPassOK", "No");
                                RequestDispatcher rd =
request.getRequestDispatcher("withdraw.jsp");
                                rd.forward(request, response);
                        } else {
                                System.out.println("I am in");
                                ResultSet rs1 = stmt.executeQuery("select * from amount where
id ='" + account_no + "'");

                                while (rs1.next()) {
                                        current_amount = rs1.getInt(2);

                                        System.out.println(current_amount);
                                }
```

```
                                        if (current_amount >= withdraw_amount) {
                                                  current_amount -= withdraw_amount;

                                                  PreparedStatement ps = conn.prepareStatement("update
amount set amount=? where id= ?");
                                                  ps.setInt(1, current_amount);
                                                  ps.setString(2, account_no);
                                                  ps.executeUpdate();

                                                  conn.close();

                                                  RequestDispatcher rd =
request.getRequestDispatcher("Withdraw_process.jsp");
                                                  rd.forward(request, response);
                                        } else {
                                                  conn.close();
                                                  request.setAttribute("EnoughMoney", "No");
                                                  RequestDispatcher rd =
request.getRequestDispatcher("withdraw.jsp");
                                                  rd.forward(request, response);
                                        }

                              }

                    } catch (SQLException e) {
               System.out.println(e.getMessage());
                              e.printStackTrace();
                    } catch (Exception ex) {
            Logger.getLogger(WithdrawServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
         }

}
```

WEB-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
        version="6.0">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>


   <servlet>
    <servlet-name>CreateAccount</servlet-name>
```

```
  <servlet-class>com.online_banking.CreateAccountServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>CreateAccount</servlet-name>
  <url-pattern>/CreateAccountServlet</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>Login</servlet-name>
  <servlet-class>com.online_banking.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Login</servlet-name>
  <url-pattern>/LoginServlet</url-pattern>
</servlet-mapping>

 <servlet>
  <servlet-name>Deposit</servlet-name>
  <servlet-class>com.online_banking.DepositServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Deposit</servlet-name>
  <url-pattern>/DepositServlet</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>Withdraw</servlet-name>
  <servlet-class>com.online_banking.WithdrawServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Withdraw</servlet-name>
  <url-pattern>/WithdrawServlet</url-pattern>
</servlet-mapping>

 <servlet>
  <servlet-name>Transfer</servlet-name>
  <servlet-class>com.online_banking.TransferServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Transfer</servlet-name>
  <url-pattern>/TransferServlet</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>Loan</servlet-name>
  <servlet-class>com.online_banking.LoanServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Loan</servlet-name>
  <url-pattern>/LoanServlet</url-pattern>
</servlet-mapping>
```

```
 <servlet>
  <servlet-name>DepositScheme</servlet-name>
  <servlet-class>com.online_banking.DepositSchemeServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>DepositScheme</servlet-name>
  <url-pattern>/DepositSchemeServlet</url-pattern>
 </servlet-mapping>


  <welcome-file-list>
    <welcome-file>home.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

JDBC_Connection

```
package com.online_banking.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBC_Connect {

   Connection connection = null;

  public Connection getConnection() throws Exception {
    try {

      Class.forName("com.mysql.cj.jdbc.Driver");
      connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/online_banking", "root", "Asif123");

    } catch ( SQLException e) {

      System.out.println("SQLException: " + e.getMessage());
      System.out.println("SQLState: " + e.getSQLState());
      System.out.println("VendorError: " + e.getErrorCode());

    }

    return connection;

  }

}
```

DatabaseOperations.java

```
package com.online_banking.database;
```

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

import com.online_banking.model.AccountModel;
import com.online_banking.model.DepositSchemeModel;
import com.online_banking.model.LoanModel;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class DatabaseOperations {
        Connection conn;
        int count1, count2;

        public boolean insertAccountDetails(AccountModel model) throws Exception {
                try {
                        JDBC_Connect connect = new JDBC_Connect();
                        conn = connect.getConnection();
                        PreparedStatement ps1 = conn.prepareStatement("insert into 
account(id,first_name,last_name,address,city,"
                                        + 
"branch,zip,username,password,phone,email,account_type,reg_date) values('" + 
model.getAccount_no()
                                        + "','" + model.getFirst_name() + "','" + 
model.getLast_name() + "','" + model.getAddress() + "','"
                                        + model.getCity() + "','" + model.getBranch() + "','" + 
model.getZip() + "','" + model.getUsername()
                                        + "','" + model.getPassword() + "','" + 
model.getPhone_number() + "','" + model.getEmail() + "','"
                                        + model.getAccount_type() + "','" + 
model.getReg_date() + "')");
                        count1 = ps1.executeUpdate();
                        System.out.println("Inserted " + count1 + " row");

                        PreparedStatement ps2 = conn.prepareStatement("insert into 
amount(id,amount) values('"
                                        + model.getAccount_no() + "','" + model.getAmount() + 
"')");
                        count2 = ps2.executeUpdate();

                        System.out.println("Inserted " + count2 + " row");

                        conn.close();

                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return ((count1 > 0) && (count2 > 0));
        }

        public boolean insertLoanDetails(LoanModel model) throws Exception {
                try {
                        JDBC_Connect connect = new JDBC_Connect();
                        conn = connect.getConnection();
                        PreparedStatement ps1 = conn
                                        .prepareStatement("insert into
loan(id,amount,status,first_name,last_name,address,email) values('"
                                                        + model.getAccount_no() + "','" +
model.getLoan_amount() + "','" + model.getStatus() + "','"
                                                        + model.getFirst_name() + "','" +
model.getLast_name() + "','" + model.getAddress() + "','"
                                                        + model.getEmail() + "')");
                        count1 = ps1.executeUpdate();

                        conn.close();

                } catch (SQLException e) {
              e.printStackTrace();
                }

                return (count1 > 0);
        }

        public AccountModel getAccountDetails(Connection conn, String account_no) throws
Exception {
                AccountModel am = new AccountModel();

                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery("select * from account where id ='" +
account_no + "'");
                while (rs.next()) {

                        // Setting all variables to model class
                        am = new AccountModel();
                        am.setAccount_no(rs.getString(1));
                        am.setFirst_name(rs.getString(2));
                        am.setLast_name(rs.getString(3));
                        am.setAddress(rs.getString(4));
                        am.setCity(rs.getString(5));
                        am.setBranch(rs.getString(6));
                        am.setZip(rs.getString(7));
                        am.setUsername(rs.getString(8));
                        am.setPassword(rs.getString(9));
                        am.setPhone_number(rs.getString(10));
                        am.setEmail(rs.getString(11));
                        am.setAccount_type(rs.getString(12));
                        am.setReg_date(rs.getString(13));
```

```
		}
		ResultSet rs1 = stmt.executeQuery("select * from amount where id ='" +
am.getAccount_no() + "'");
		while (rs1.next()) {
			am.setAmount(rs1.getString(2));
		}
		return am;
	}

	public boolean insertDepositScheme(DepositSchemeModel model) throws Exception {
		try {
			JDBC_Connect connect = new JDBC_Connect();
			conn = connect.getConnection();

			// getting current date
				DateFormat dateFormat = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
			Date date = new Date();
			String current_time = dateFormat.format(date);

			PreparedStatement ps1 = conn
					.prepareStatement("insert into
deposit(id,year,interest,amount,deposit_date) values('"
							+ model.getAccount_no() + "','" +
model.getYear() + "','" + model.getInterest_rate() + "','"
							+ model.getAmount() + "','" +
current_time + "')");
			count1 = ps1.executeUpdate();
			System.out.println("Inserted " + count1 + " row");

			conn.close();

		} catch (SQLException e) {
			e.printStackTrace();
		}

		return ((count1 > 0));
	}

	public ArrayList<LoanModel> getLoanList(Connection conn) throws Exception {
		ArrayList<LoanModel> loanList = new ArrayList<>();
		LoanModel loanModel;

		Statement stmt = conn.createStatement();
		ResultSet rs = stmt.executeQuery("select * from loan where status='pending'");
		while (rs.next()) {
			loanModel = new LoanModel();
			loanModel.setAccount_no(rs.getString(1));
			loanModel.setLoan_amount(rs.getInt(2));
			loanModel.setStatus(rs.getString(3));
			loanModel.setFirst_name(rs.getString(4));
```

```
                    loanModel.setLast_name(rs.getString(5));
                    loanModel.setAddress(rs.getString(6));
                    loanModel.setEmail(rs.getString(7));

                    loanList.add(loanModel);

            }

            return loanList;

      }

      public void UpdateAmount(String account_no, int loan_amount) throws SQLException,
Exception {
            int current_amount = 0;
            JDBC_Connect connect = new JDBC_Connect();
            conn = connect.getConnection();

            Statement stmt = conn.createStatement();
            ResultSet rs1 = stmt.executeQuery("select * from amount where id ='" +
account_no + "'");

            while (rs1.next()) {
                    current_amount = rs1.getInt(2);

            }

            current_amount += loan_amount;

            // Updating Loan amount
            PreparedStatement ps = conn.prepareStatement("update amount set amount=?
where id= ?");
            ps.setInt(1, current_amount);
            ps.setString(2, account_no);
            ps.executeUpdate();

            PreparedStatement ps1 = conn.prepareStatement("update loan set status=? where
id= ?");
            ps1.setString(1, "success");
            ps1.setString(2, account_no);
            ps1.executeUpdate();

            conn.close();

      }

}
```

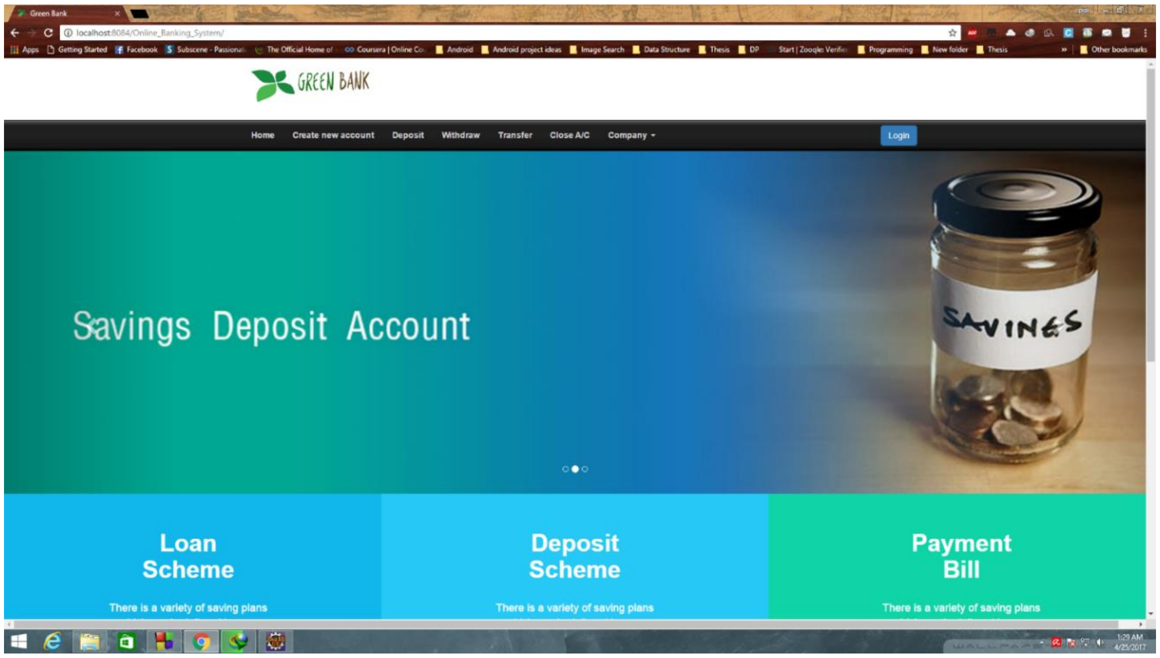
AccountModel.java

```
package com.online_banking.model;
```

```
public class AccountModel {

    private String account_no;
    private String first_name;
    private String last_name;
    private String address;
    private String city;
    private String branch;
    private String zip;
    private String username;
    private String password;
    private String phone_number;
    private String email;
    private String account_type;
    private String reg_date;
    private String amount;

    public String getAccount_no() {
        return account_no;
    }

    public void setAccount_no(String account_no) {
        this.account_no = account_no;
    }

    public String getFirst_name() {
        return first_name;
    }

    public void setFirst_name(String first_name) {
        this.first_name = first_name;
    }

    public String getLast_name() {
        return last_name;
    }

    public void setLast_name(String last_name) {
        this.last_name = last_name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getCity() {
        return city;
```

```
}

public void setCity(String city) {
    this.city = city;
}

public String getBranch() {
    return branch;
}

public void setBranch(String branch) {
    this.branch = branch;
}

public String getZip() {
    return zip;
}

public void setZip(String zip) {
    this.zip = zip;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getPhone_number() {
    return phone_number;
}

public void setPhone_number(String phone_number) {
    this.phone_number = phone_number;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
        this.email = email;
    }

    public String getAccount_type() {
        return account_type;
    }

    public void setAccount_type(String account_type) {
        this.account_type = account_type;
    }

    public String getReg_date() {
        return reg_date;
    }

    public void setReg_date(String reg_date) {
        this.reg_date = reg_date;
    }

    public String getAmount() {
        return amount;
    }

    public void setAmount(String amount) {
        this.amount = amount;
    }

    @Override
    public String toString() {
        return "AccountModel{" + "account_no=" + account_no + ", first_name=" + first_name + ",
last_name=" + last_name + ", address=" + address + ", city=" + city + ", branch=" + branch + ",
zip=" + zip + ", username=" + username + ", password=" + password + ", phone_number=" +
phone_number + ", email=" + email + ", account_type=" + account_type + ", reg_date=" +
reg_date + ", amount=" + amount + '}';
    }


}
```

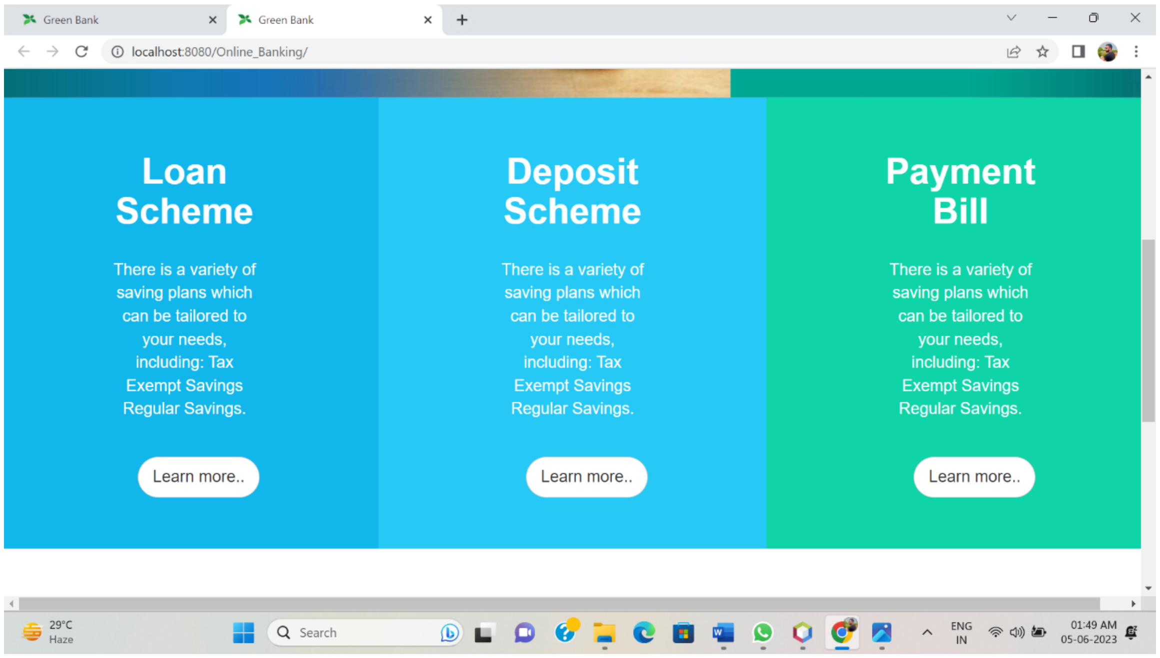
Home Page



There is a variety of saving plans

There is a variety of saving plans

There is a variety of saving plans



Loan Scheme

There is a variety of saving plans which can be tailored to your needs, including: Tax Exempt Savings Regular Savings.

Learn more..

Deposit Scheme

There is a variety of saving plans which can be tailored to your needs, including: Tax Exempt Savings Regular Savings.

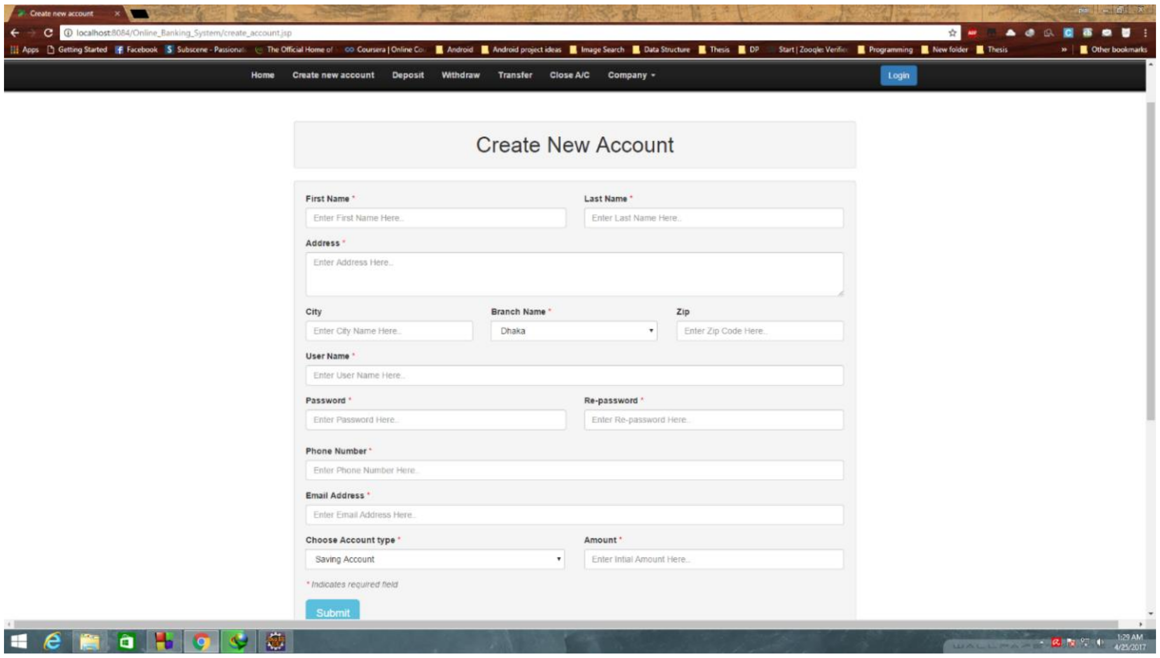
Learn more..

Payment Bill

There is a variety of saving plans which can be tailored to your needs, including: Tax Exempt Savings Regular Savings.

Learn more..

Registration Form



Create New Account

First Name * Enter First Name Here..

Last Name * Enter Last Name Here..

Address * Enter Address Here..

City Enter City Name Here..

Branch Name * Dhaka

Zip Enter Zip Code Here..

User Name * Enter User Name Here..

Password * Enter Password Here..

Re-password * Enter Re-password Here..

Phone Number * Enter Phone Number Here..

Email Address * Enter Email Address Here..

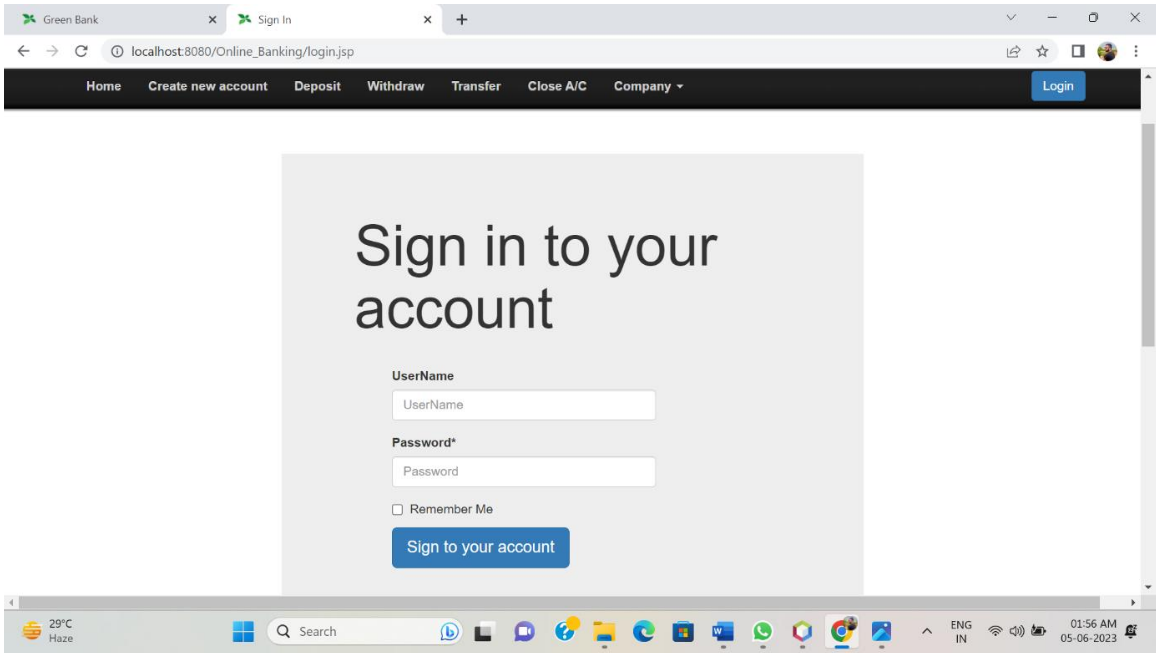
Choose Account type * Saving Account

Amount * Enter Intial Amount Here..

* Indicates required field

Submit

Login



Sign in to your account

UserName

UserName

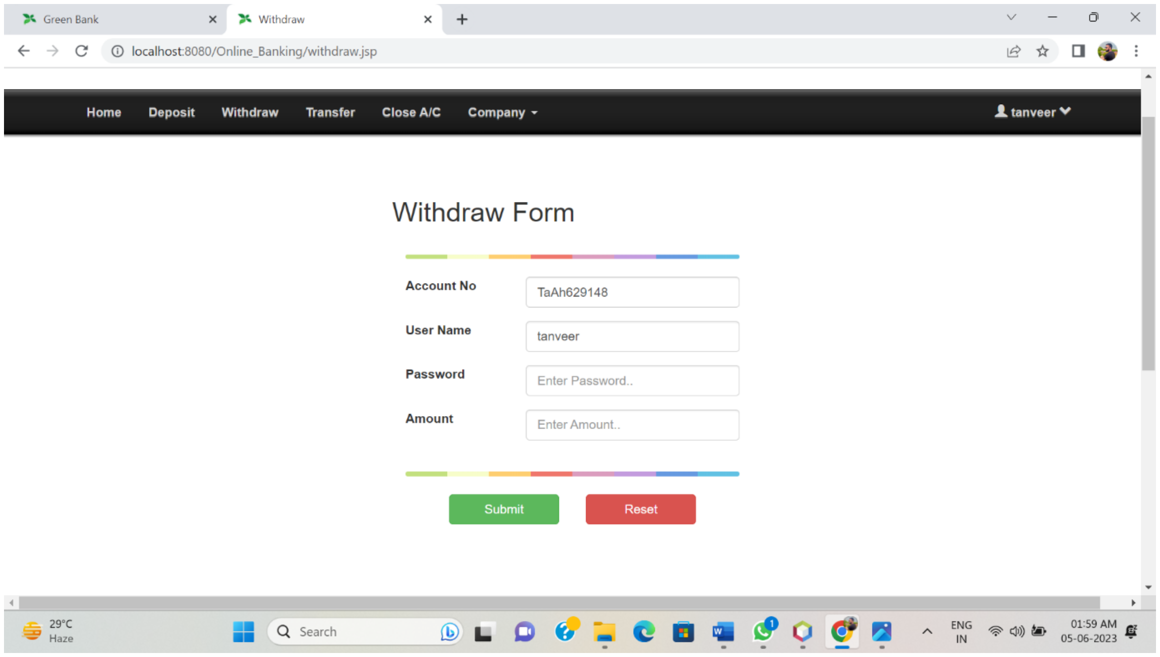
Password*

Password

Remember Me

Sign to your account

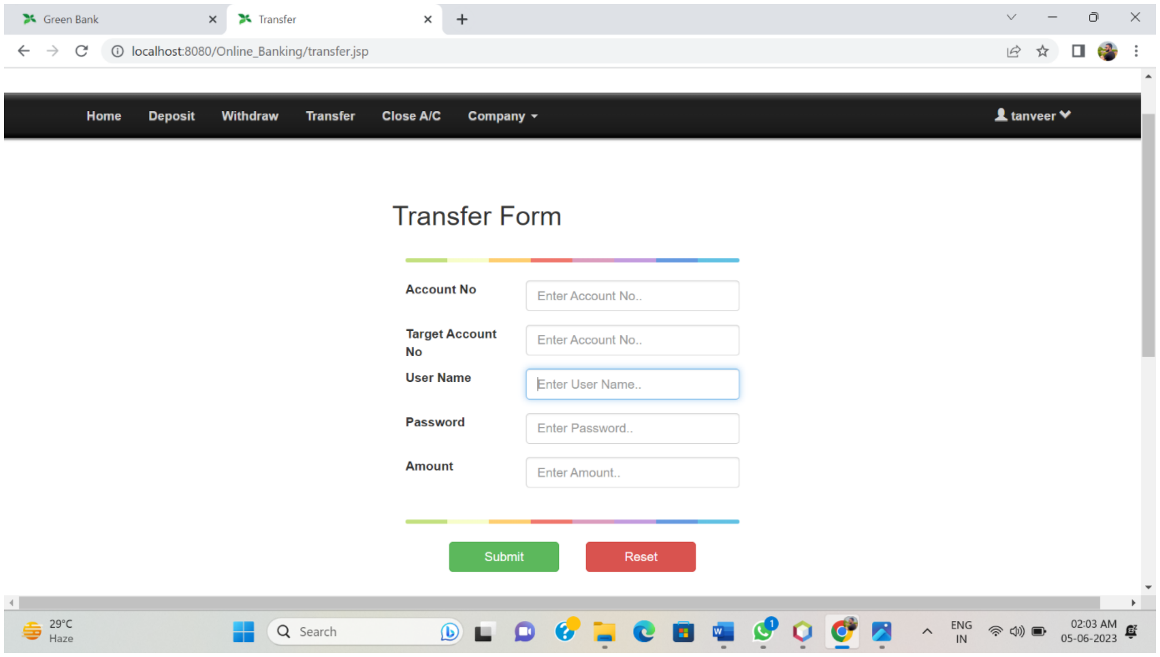
Withdraw Form



Withdraw Form

Account No	TaAh629148
User Name	tanveer
Password	Enter Password..
Amount	Enter Amount..

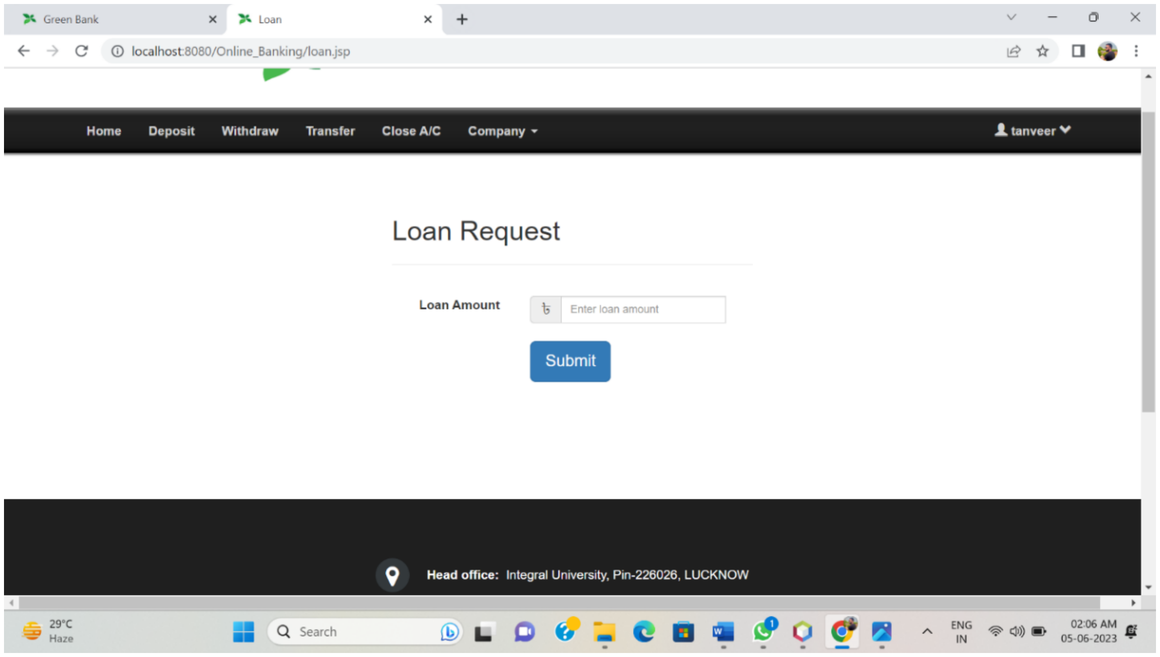
Transfer Form



Transfer Form

Account No	Enter Account No..
Target Account No	Enter Account No..
User Name	Enter User Name..
Password	Enter Password..
Amount	Enter Amount..

Loan Request



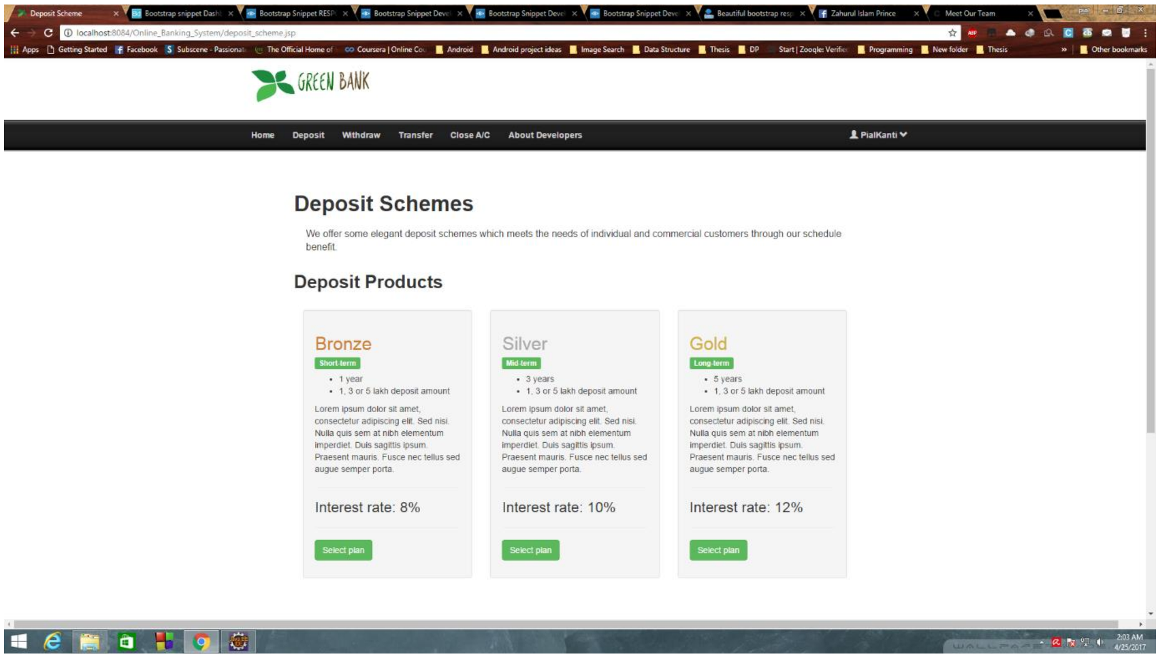
Loan Request

Loan Amount: Enter loan amount

Submit

Head office: Integral University, Pin-226026, LUCKNOW

Deposit Scheme



Deposit Schemes

We offer some elegant deposit schemes which meets the needs of individual and commercial customers through our schedule benefit.

Deposit Products

Bronze
Short-term
- 1 year
- 1, 3 or 5 lakh deposit amount

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.

Interest rate: 8%

Select plan

Silver
Mid-term
- 3 years
- 1, 3 or 5 lakh deposit amount

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.

Interest rate: 10%

Select plan

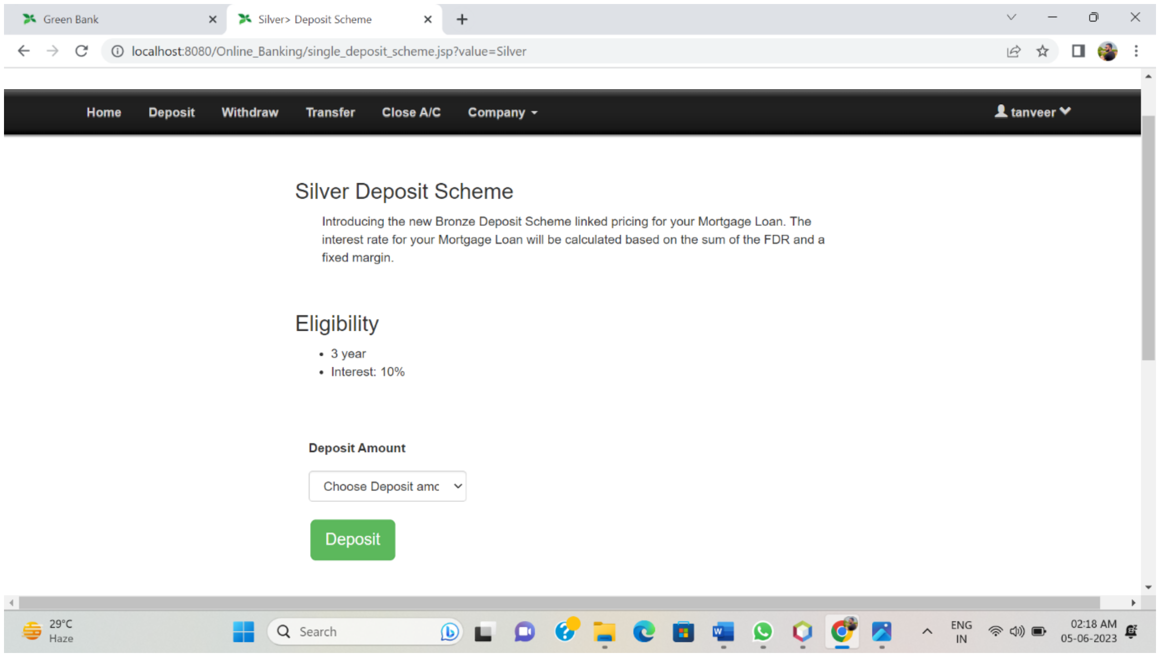
Gold
Long-term
- 5 years
- 1, 3 or 5 lakh deposit amount

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta.

Interest rate: 12%

Select plan

Silver Deposit Scheme


Green Bank
Silver> Deposit Scheme
localhost:8080/Online_Banking/single_deposit_scheme.jsp?value=Silver

Silver Deposit Scheme

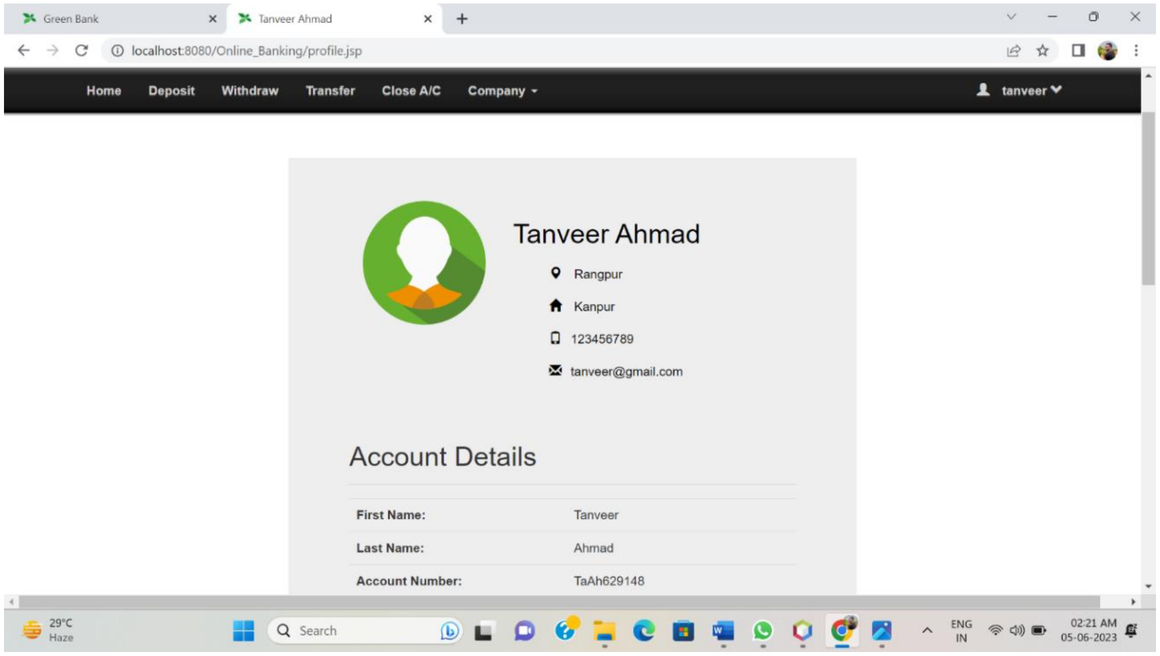
Introducing the new Bronze Deposit Scheme linked pricing for your Mortgage Loan. The interest rate for your Mortgage Loan will be calculated based on the sum of the FDR and a fixed margin.

Eligibility

- 3 year
- Interest: 10%

Deposit Amount

Profile


Green Bank
Tanveer Ahmad
localhost:8080/Online_Banking/profile.jsp

Tanveer Ahmad

Rangpur

Kanpur

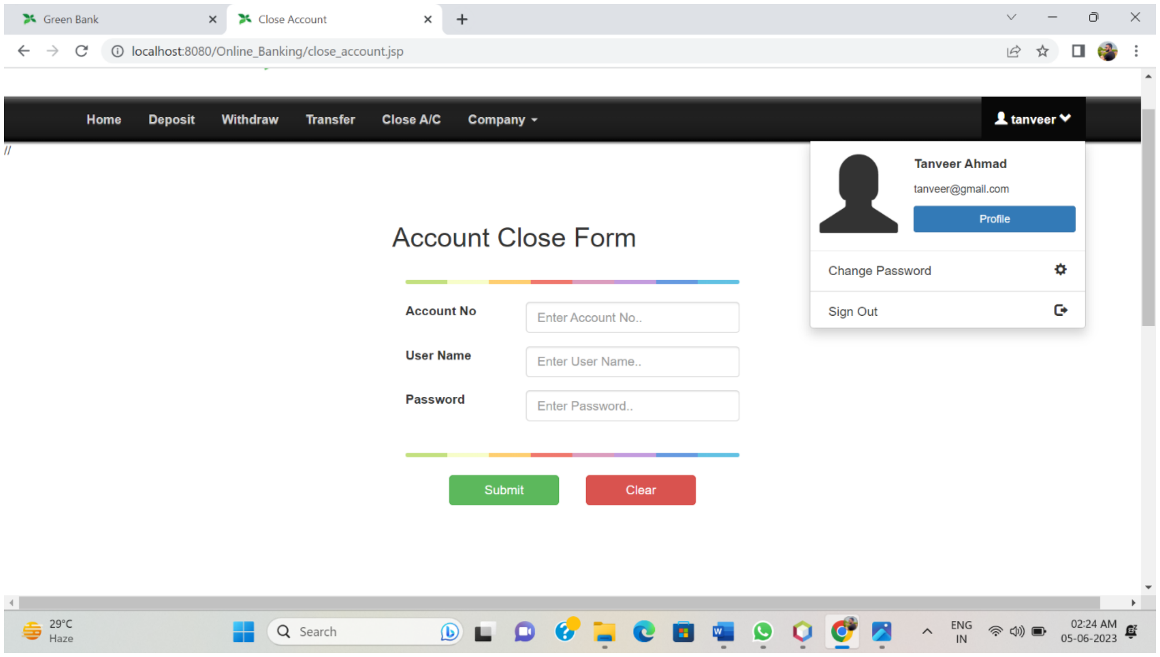
123456789

tanveer@gmail.com

Account Details

First Name:	Tanveer
Last Name:	Ahmad
Account Number:	TaAh629148

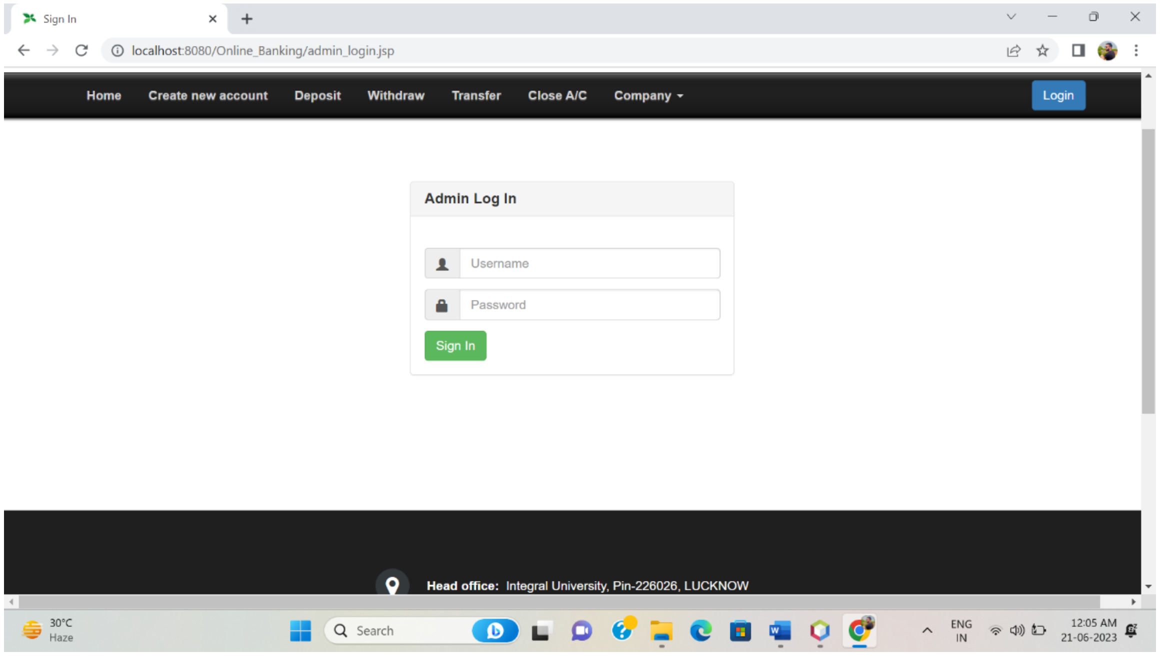
Sign out



Account Close Form

Account No	Enter Account No..
User Name	Enter User Name..
Password	Enter Password..

Admin Login

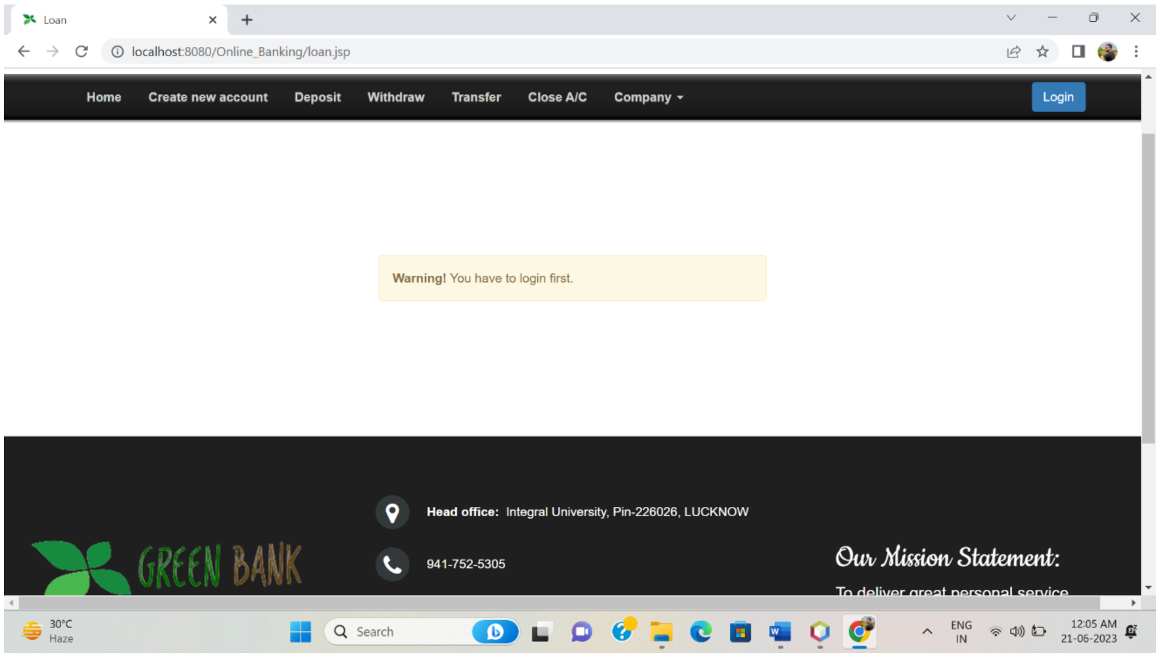


Admin Log In

Username
Password

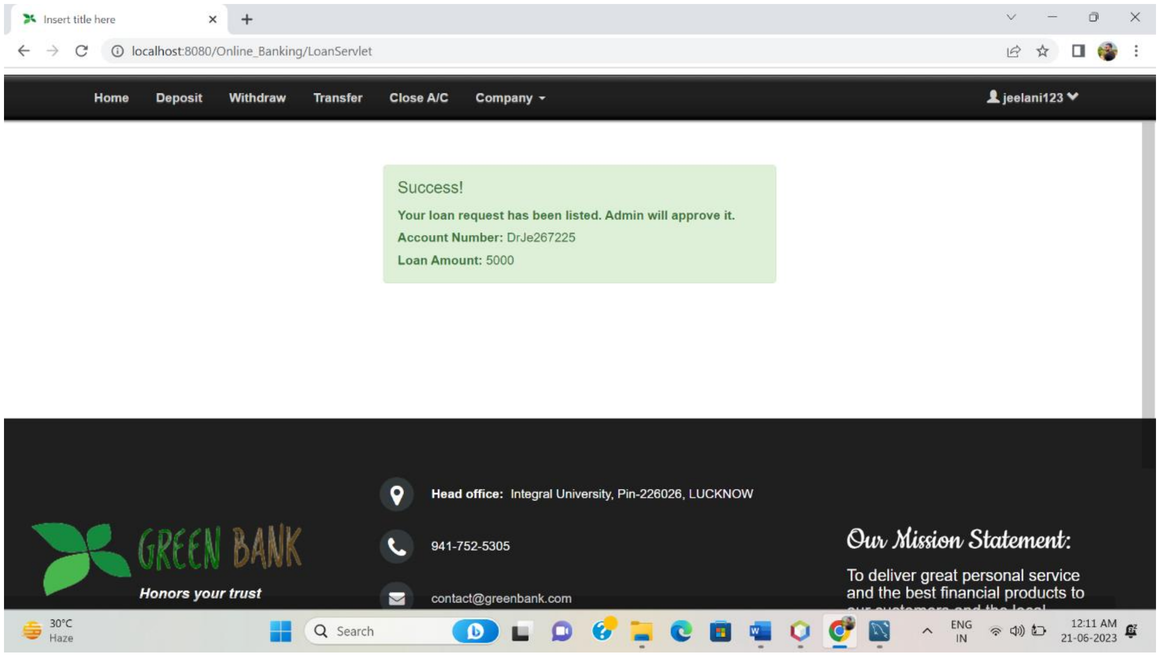
Head office: Integral University, Pin-226026, LUCKNOW

Warning message



Warning! You have to login first.

Loan Request Success


Insert title here
localhost:8080/Online_Banking/LoanServlet
Home
Deposit
Withdraw
Transfer
Close A/C
Company
jeelani123

Success!

Your loan request has been listed. Admin will approve it.

Account Number: DrJe267225

Loan Amount: 5000

Head office: Integral University, Pin-226026, LUCKNOW
941-752-5305
contact@greenbank.com
GREEN BANK
Honors your trust
Our Mission Statement:
To deliver great personal service and the best financial products to

6. SYSTEM TESTING AND IMPLEMENTATION

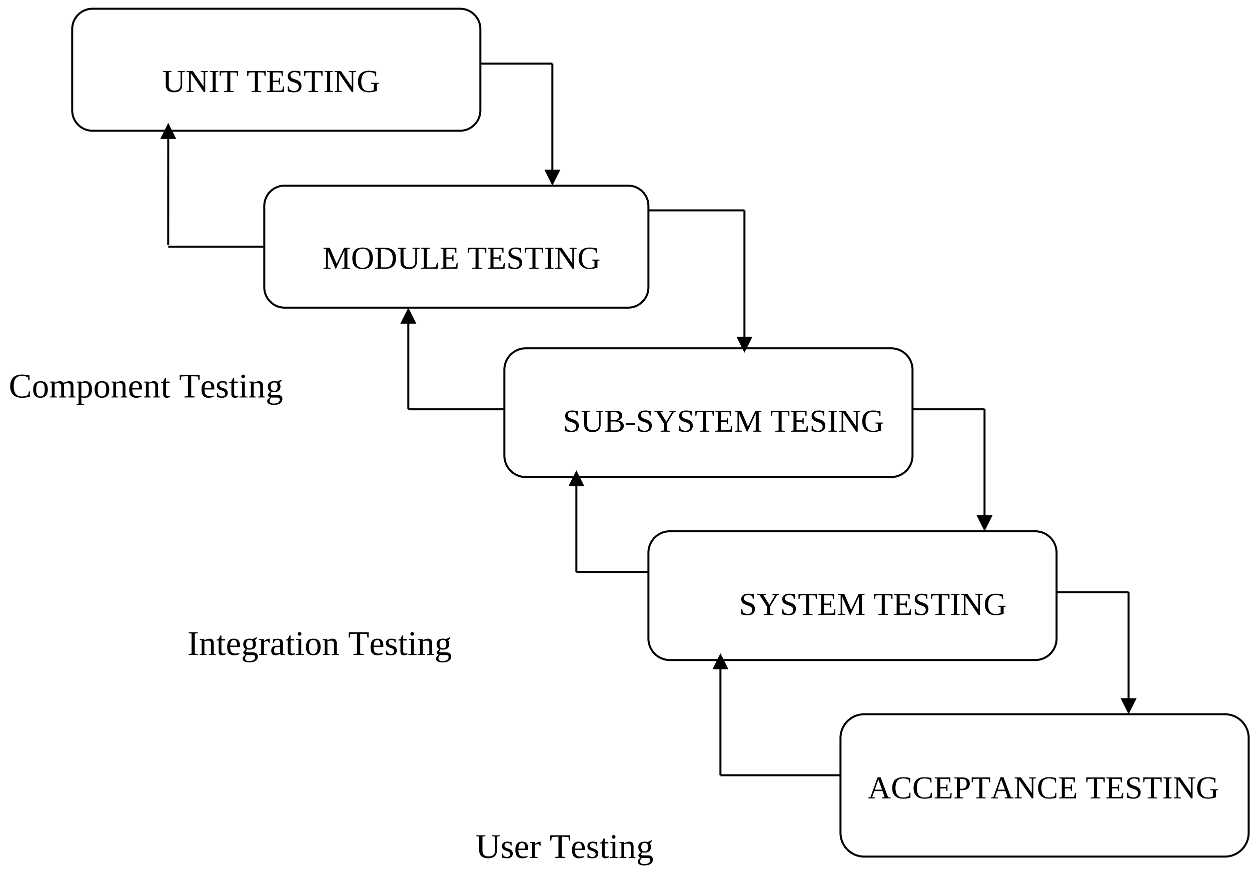
6.1. Introduction

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

6.2. Strategic Approach to Software Testing

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software, we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally, we arrive at system testing, where the software and other system elements are tested as a whole.



6.2.1. Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

6.2.2. White Box Testing

This type of testing ensures that

- All independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity.

6.2.3. Basic Path Testing

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:Use the design of the code and draw correspondent flow graph.

6.2.4. Conditional Testing

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

6.2.5. Data Flow Testing

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

6.2.6. Loop Testing

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

- All the loops were tested at their limits, just above them and just below them.
- All the loops were skipped at least once.
- For nested loops test the inner most loop first and then work outwards.

6.3. Security In Software

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls.

6.3.1. Client-side Validation

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

- VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
- Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
- Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

6.3.2. Serverside Validation

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

- Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
- User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.

7. CONCLUSION AND FUTURE SCOPE

This project developed, incorporated all the activities involved in the browsing centre. It provides all necessary information to the management as well as the customer with the use of this system; the user can simply sit in front of the system and monitor all the activities without any physical movement of the file. Management can service the customer's request best in time. The system provides quickly and valuable information. These modules have been integrated for effective use of the management for future forecasting and for the current need.

7.1 Future Scope

The project was developed to fulfil user requirements however there are lots of scope to improve the performance of the Banking System in the area of user interface, database performance, and query processing time. So, there are many things for future enhancement of this project. The future enhancement that are possible in this project are as follows.

1. **Integration with emerging technologies:** The system can be integrated with emerging technologies such as blockchain and artificial intelligence to enhance security, automate processes, and improve decision-making. This can lead to increased efficiency and cost savings.
2. **Extension to support more languages and currencies:** The system can be extended to support more languages and currencies to cater to the needs of a broader customer base. This can help banks expand their reach and serve a more diverse customer base.
3. **Credit card management:** The system can be expanded to include credit card management, allowing customers to manage their credit card accounts, view transactions, and pay bills online.

REFERENCES

- ***For Java installation***
  - https://www.java.com/en/download/
- ***For Oracle Data Base Installation***
  - http://www.oracle.com/index.html
- ***Reference Websites***
  - www.javatpoint.com
  - www.w3schools.com
  - http://www.tutorialspoint.com/java/index.htm
- ***Reference Books***
  - Thinking in java
  - OCJP Certified Programmer for Java
  - Learn Java in Easy Steps
  - Complete reference Java

RESUME

MD. ASIF KHAN

+91 63941 23775
Email Id: mdasifkhan7317@gmail.com

OBJECTIVE

To be professional towards the organization and to work where my skills will be useful, active to accept all challenges and work hard towards achieving goals of the organization with new techniques.

EDUCATIONAL QUALIFICATION

- Pursuing MCA 2nd year from Integral University Lucknow.
- Passed out BCA from Integral University Lucknow in 2021.
- Passed out 12th from ISC in 2018.
  Marks 74%
- Passed out 10th from ICSE in 2016.
  Marks 68%

TECHNICAL SKILLS

- **Programming Language:** JAVA, JavaScript, JSP, HTML, CSS, Bootstrap.
- **API's:** Servlet.
- **Software:** NetBeans.
- **Databases:** MySQL.

WORK EXPERIENCE

FRESHER

STRENGTH & HOBBIES

- Good strength of working with team, strong ability to work creating and determination.
- Can easily adjust and adopt according to environment.
- Good Communication skill.

PERSONAL DETAILS

Father's Name	: Md. Aslam Khan
D. O. B	: 20-08-2000
Address	: Vill: - Khizirpur, post: - Zamania, dist.: - Ghazipur.
Marital Status	: Single

Date ………………………

RESUME

Mahboob Alam
Bamhour, mubarakpur
Azamgarh
Email **:- saifshaikh9621@gmail.com**
Phone:-8546048553

OBJECTIVES

To seek a dynamic and challenging carrier an organization strives for excellence with my knowledge and team effort while making positive contribution to promote the individual opportunity and professional growth.

EDUCATIONAL QUALIFICATION

Education specification	**institution**	**Passing year**	**Marks obtained in %**
MCA	Integral university	2023	81
BCA	Integral university	2021	72
12th	Kashi Ic Hazipur Bamhour Azamgarh	2018	51.16
10th	Central Public School	2015	58.9

COMPUTER SKILL

- Programming language **:- Core JAVA, HTML ,css,Bootstrap**
- Operating system**:- Window , Linux**
- Web design:- **HTML**

STRENGTH FACTOR

- ➢ Good strength of working with team , strong ability to work creating and determination
- ➢ Can easily adjust and adopt according to environment

HOBBIES

- ➢ Playing cricket
- ➢ Reading story

PERSONAL PROFILE

Language **:- Hindi , English**
D.O.B :- **26/01/1999**
Sports **:- Cricket**
Marital status :- **Single**
Nationality **:- Indian**