

## DBMS Mini Project

---

# CORE (MINI)BANKING MANAGEMENT SYSTEM

---

Haneyah Seemein

Section: K

Semester: 5

## Description

The DBMS project designed for managing a bank is a comprehensive and user-friendly system that caters to the diverse needs of customers, employees, and administrators. With a streamlined UI created using Streamlit in Python, the system ensures efficient and secure access for users with different roles.

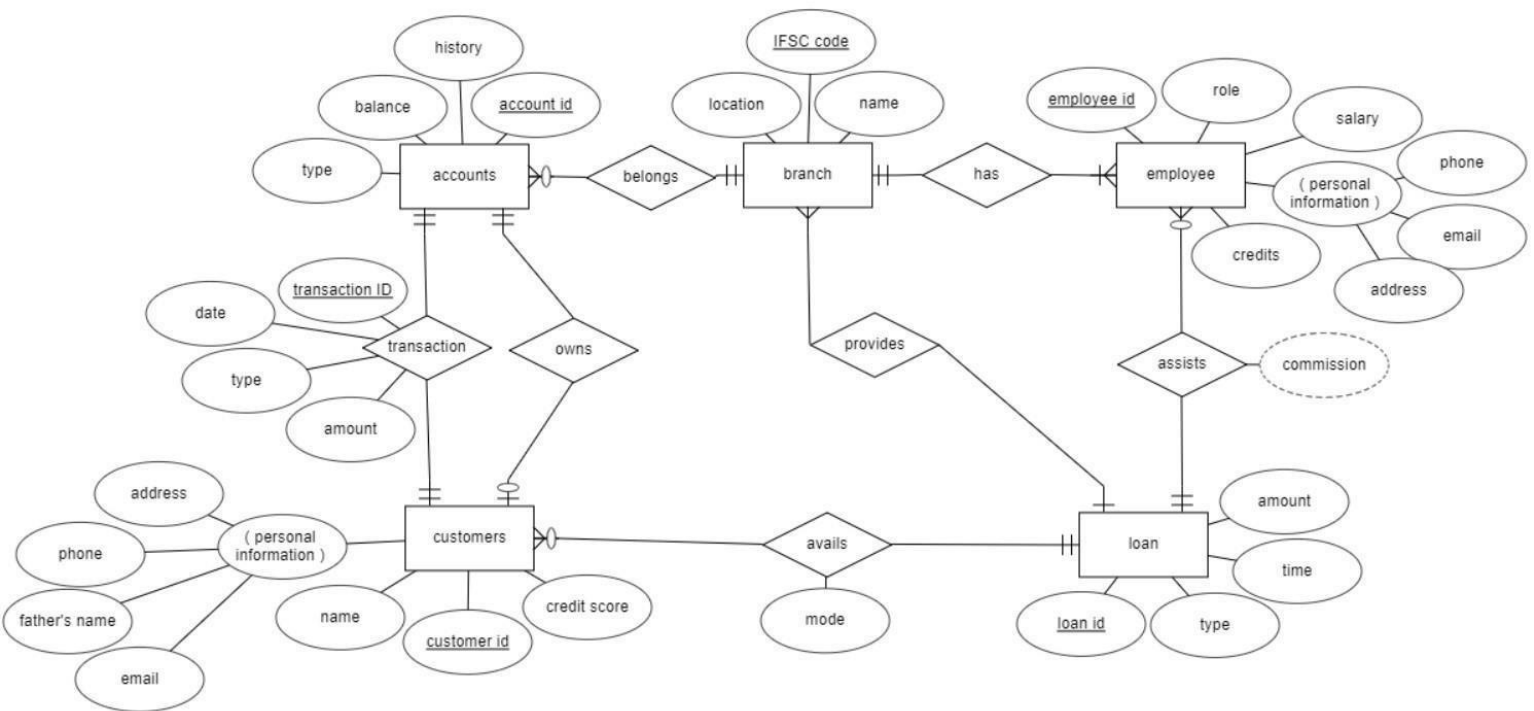
For customers, the platform offers a personalized experience by providing access to their essential banking information. After logging in, they can view their account details, including account ID, balance, account branch, IFSC code, and account type. Additionally, customers can access their transaction history, keeping them informed about their financial activities. For those who have availed loans, the system displays detailed information about their loans, such as Loan ID, loan amount, loan type (e.g., personal, home, education), interest rates, and the employee's name who assisted them. Moreover, customers have access to a helpline number for direct communication with the assisting employee.

Employees have a set of powerful tools at their disposal, allowing them to facilitate financial transactions seamlessly. They can perform credit and debit transactions for customer accounts, apply for loans on behalf of customers, and register new accounts. Furthermore, employees can easily track and access a list of customers they have helped to avail loans, ensuring efficient customer service and relationship management.

For administrators, the system offers control over the bank's operations. They can hire and fire employees, manage customer data by deleting accounts and customers when necessary, and utilize custom commands to meet the bank's specific needs. The interface provides a user-friendly experience for administrators to make essential decisions and tailor the system to the bank's requirements.

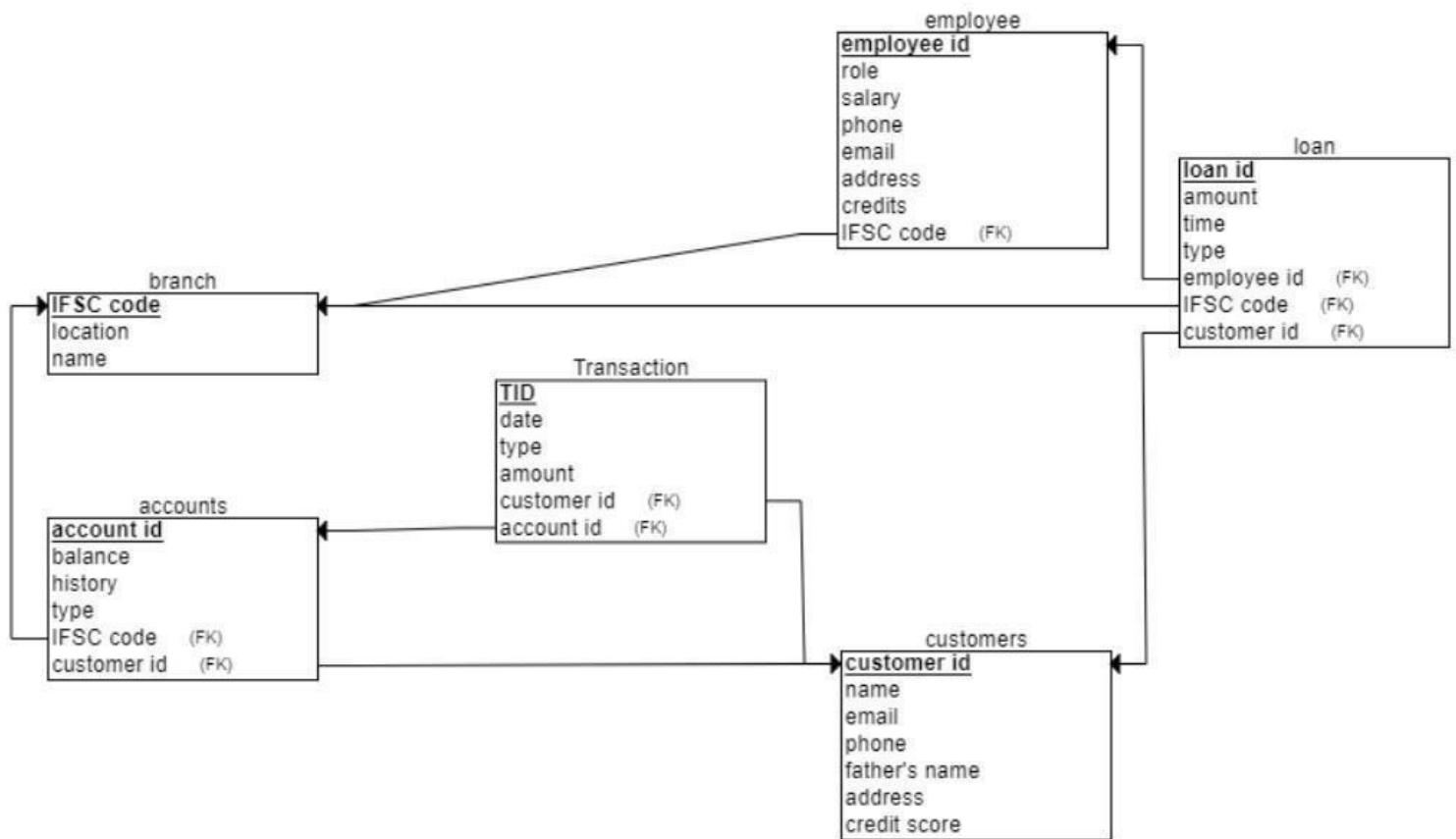
In summary, this DBMS project offers a well-rounded solution for efficient bank management, providing customers with a seamless banking experience, employees with tools to serve customers effectively, and administrators with the control and flexibility they need to manage the bank's operations effectively.

## ER diagram



The given ER Diagram represents the superficial relationship between entities in a Bank. It is a simple outlook of a bank with basic required functionalities.

- **Account:** The account belongs to a customer of the bank and has a unique `account_id` to identify each account. The balance and the transaction history of the same is also tracked by various functionalities.
- **Branch:** Each branch/unit contains the employees, accounts and an IFSC code.
- **Employee:** The employees perform various operations to assist the customer.  
The employees help the customers for availing loans.
- **Customer:** The customer with his profile containing his name, personal information and credit score may or may not have an account and loan.
- **Loan:** created by the employee for the customer, has a unique identifier `loan_id`, time period, amount and the loan type like personal, home, education etc.



## Relational Schema

# Building Database

## Creating branch

```
CREATE TABLE branch
(
  name VARCHAR(255) NOT NULL,   IFSC INT(4) NOT NULL,   location VARCHAR(255) NOT
  NULL,
  PRIMARY KEY (IFSC) );
```

## Creating customer

```
CREATE TABLE customer
(
  customer_id INT(7) NOT NULL,   name VARCHAR(255) NOT NULL,   phone VARCHAR(10) NOT NULL,
  email VARCHAR(255) NOT NULL,   address VARCHAR(255) NOT NULL,   credit_score INT(1) NOT NULL,
  PRIMARY KEY (customer_id) );
```

## Creating employee

```
CREATE TABLE employee
(
  employee_id INT(7) NOT NULL,
  name VARCHAR(255) NOT NULL,
  role VARCHAR(10) NOT NULL,
  salary INT NOT NULL,   phone
  VARCHAR(10) NOT NULL,
  email VARCHAR(255) NOT
  NULL,   address
  VARCHAR(255) NOT NULL,
  credits INT(1) NOT NULL,
  IFSC
  INT(4) NOT NULL,
  PRIMARY KEY (employee_id),
  FOREIGN KEY (IFSC) REFERENCES branch(IFSC)
);
```

## Creating loan

```
CREATE TABLE loan
(   loan_id INT(10) NOT NULL,
    amount INT NOT NULL,   time
    INT NOT NULL,   type
    VARCHAR(10) NOT NULL,
    customer_id INT(7) NOT
    NULL,   employee_id INT(7)
    NOT NULL,
    IFSC INT(4) NOT NULL,
    PRIMARY KEY (loan_id),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (employee_id) REFERENCES
    employee(employee_id),   FOREIGN KEY (IFSC) REFERENCES
    branch(IFSC) );
```

## Creating account

```
CREATE TABLE account
(   account_id INT(10) NOT
    NULL,   balance INT NOT
    NULL,   type VARCHAR(10)
    NOT NULL,   IFSC INT(4) NOT
    NULL,   customer_id INT(7)
    NOT NULL,
    PRIMARY KEY (account_id),
    FOREIGN KEY (IFSC) REFERENCES branch(IFSC),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
    );
```

## Creating transaction

```
CREATE TABLE Transaction
(
    TID INT(10) NOT NULL,
    date INT(8) NOT NULL,
    amount INT NOT NULL,
    account_id INT(10) NOT
    NULL,   customer_id
    INT(7) NOT NULL,
    PRIMARY KEY (TID),
    FOREIGN KEY (customer_id) REFERENCES
    customer(customer_id),   FOREIGN KEY (account_id)
    REFERENCES account(account_id) );
```

## Join Queries

Customer's bank balance and the amount he/she owes to bank for loan.

```
SELECT customer.name, account.balance, loan.amount FROM customer INNER JOIN account
INNER JOIN loan where customer.customer_id=account.customer_id AND
loan.customer_id=customer.customer_id;
```

```
MariaDB [bank]> SELECT customer.name, account.balance, loan.amount FROM customer INNER JOIN account INNER JOIN loan where
customer.customer_id=account.customer_id AND loan.customer_id=customer.customer_id;
+-----+-----+-----+
| name          | balance | amount |
+-----+-----+-----+
| Hilary Welband | 200000  | 380000 |
| Haneyah Seemein | 3100    | 140000 |
+-----+-----+-----+
2 rows in set (0.015 sec)
```

Names of all the customers along with their addresses and account id who belong to the branch with IFSC code 2000.

```
SELECT branch.name, customer.name, account.account_id, branch.location, customer.address FROM
branch INNER JOIN account INNER JOIN customer WHERE branch.IFSC=2000 AND
customer.customer_id=account.customer_id;
```

```
MariaDB [bank]> SELECT branch.name, customer.name, account.account_id, branch.location, customer.address
-> FROM branch INNER JOIN account INNER JOIN customer
-> WHERE branch.IFSC=2000 AND customer.customer_id=account.customer_id;
+-----+-----+-----+-----+-----+
| name      | name          | account_id | location    | address          |
+-----+-----+-----+-----+-----+
| Bangalore | Haneyah Seemein | 2000       | Banashankari | Banashankari     |
| Bangalore | Hilary Welband  | 1111       | Banashankari | 44 Knutson Pass  |
| Bangalore | Juliette Nickerson | 2020       | Banashankari | 9 Prairieview Crossing |
+-----+-----+-----+-----+-----+
3 rows in set (0.001 sec)
```

### 3. All the employees working in the branches located in Bangalore

```
SELECT employee.name, employee.role, branch.location, employee.salary,
employee.credits FROM employee INNER JOIN branch WHERE branch.name='Bangalore' AND
employee.IFSC=branch.IFSC;
```

```
MariaDB [bank]> SELECT employee.name, employee.role, branch.location, employee.salary, employee.credits FROM employee INNER JOIN
branch WHERE branch.name='Bangalore' AND employee.IFSC=branch.IFSC;
```

name	role	location	salary	credits
Cathrin Kenwyn	Branch Hea	inside Mantri Square	70000	10
Bria Coslitt	Advisor	Banashankari	4000	8
Cris Bazley	Advisor	Banashankari	4000	1
Ade Mathonnet	Advisor	Banashankari	4000	1
Tammie Grogan	Advisor	Banashankari	4000	5
Katey Korpola	Manager	Banashankari	160000	2
Job Ingerith	Branch Hea	Banashankari	70000	5

```
7 rows in set (0.004 sec)
```

### 4. Customers who made transactions of more than 1000 rupees

```
SELECT customer.name, transaction.TID, transaction.amount FROM customer JOIN account JOIN
transaction WHERE customer.customer_id=account.customer_id AND
transaction.account_id=account.account_id AND transaction.amount>1000;
```

```
MariaDB [bank]> SELECT customer.name, transaction.TID, transaction.amount FROM customer JOIN account JOIN transaction WHERE
customer.customer_id=account.customer_id AND transaction.account_id=account.account_id AND transaction.amount>1000;
```

name	TID	amount
Haneyah Seemein	44505	1400
Haneyah Seemein	47431	1400

```
2 rows in set (0.001 sec)
```



## Aggregate Functions

### 1. Maximum salary an employee gets in the bank

```
SELECT MAX(salary) FROM employee;
```

```
MariaDB [bank]> SELECT MAX(salary) FROM employee;
+-----+
| MAX(salary) |
+-----+
|      160000 |
+-----+
1 row in set (0.000 sec)
```

### Average credits/ratings of officer.

```
SELECT AVG(credits) FROM employee WHERE role='Officer';
```

```
MariaDB [bank]> SELECT AVG(credits) FROM employee WHERE role='Officer';
+-----+
| AVG(credits) |
+-----+
|         5.2500 |
+-----+
1 row in set (0.001 sec)
```

### Number of employees working for Bombay branch.

```
SELECT count(*) AS BombayEmployees FROM employee where IFSC=3000;
```

```
MariaDB [bank]> SELECT count(*) AS BombayEmployees FROM employee where IFSC=3000;
+-----+
| BombayEmployees |
+-----+
|                8 |
+-----+
1 row in set (0.001 sec)
```

#### 4. Number of Advisors working for the bank

```
SELECT count(*) AS BranchHeads FROM employee where role='Advisor';
```

```
MariaDB [bank]> SELECT count(*) AS BranchHeads FROM employee where role='Advisor';
+-----+
| BranchHeads |
+-----+
|          13 |
+-----+
1 row in set (0.000 sec)
```

## Set Operations

**IDs of customers whose loan amount is greater than 10000 and bank balance more than 2000**

```
SELECT customer_id FROM loan WHERE amount>10000 INTERSECT SELECT customer_id FROM account WHERE balance>2000;
```

```
MariaDB [bank]> SELECT customer_id FROM loan WHERE amount>10000 INTERSECT SELECT customer_id FROM account WHERE balance>2000;
+-----+
| customer_id |
+-----+
|          3141 |
|          2000 |
+-----+
2 rows in set (0.016 sec)
```

**2. Names of the employees working for Bombay Branch along with Managers**

```
SELECT name FROM employee WHERE IFSC=3000 UNION SELECT name FROM employee WHERE role='Manager';
```

```
MariaDB [bank]> SELECT name FROM employee WHERE IFSC=3000 UNION SELECT name FROM employee WHERE role='Manager';
+-----+
| name |
+-----+
| Ninnette Normington |
| Marwin Duffin |
| Latashia Berry |
| Peyter Parsisson |
| Tallia Beacroft |
| Bronnie Hearle |
| Angie Menco |
| Nananne Verney |
| Kerrill Happer |
| Katey Korpola |
| Cecilla Wooland |
+-----+
11 rows in set (0.013 sec)
```

**3. All the employees in Bangalore except the managers**

```
SELECT name FROM employee WHERE IFSC=1000 EXCEPT SELECT name FROM employee WHERE role='Manager';
```

```
MariaDB [bank]> SELECT name FROM employee WHERE IFSC=2000 EXCEPT SELECT name FROM employee WHERE role='Manager';
+-----+
| name |
+-----+
| Bria Coslitt |
| Cris Bazley |
| Ade Mathonnet |
| Tammie Grogan |
| Job Ingerith |
+-----+
5 rows in set (0.001 sec)
```

#### 4. Names of all good employees and customers for bonus/gifts

```
SELECT name FROM employee WHERE credits>8 UNION SELECT name FROM customer WHERE credit_score>8;
```

```
MariaDB [bank]> SELECT name FROM employee WHERE credits>8 UNION SELECT name FROM customer WHERE credit_score>8;
+-----+
| name |
+-----+
| Marwin Duffin |
| Corrie Samworth |
| Agretha Embra |
| Cathrin Kenwyn |
| DUMMY |
| Haneyah Seemein |
| Janka Kulic |
| Adella Pegden |
| Monroe Lightoller |
| Flore Brazer |
| Amandie Fairhead |
| Frances Sirey |
| Koressa Parlor |
| Mallorie Parvin |
| Merrily Pursehouse |
| Bald Coopey |
| Quentin Stoter |
| Manny Fullun |
| Heindrick Helversen |
| Darcee Radbone |
+-----+
20 rows in set (0.001 sec)
```

## Function

```

DELIMITER $$
CREATE FUNCTION totalamount(amount INT, type VARCHAR(10))
RETURNS INT
BEGIN
DECLARE interestpercent INT;
IF type = 'Home' THEN
SET interestpercent = 5;
ELSEIF type = 'Personal' THEN
SET interestpercent = 10;
ELSEIF type = 'Education' THEN
SET interestpercent = 2;
ELSEIF type = 'Fund' THEN
SET interestpercent = 0;
END IF;
RETURN amount * interestpercent + amount;
END $$
DELIMITER;

```

```

MariaDB [bank]> SELECT totalamount(20000,"Personal");
+-----+
| totalamount(20000,"Personal") |
+-----+
| 220000 |
+-----+
1 row in set (0.026 sec)

```

## View

```
CREATE VIEW [leads] AS SELECT
loan.loan_id,customer.name,loan.type,customer.phone,loan.amount FROM loan INNER
JOIN customer INNER JOIN employee
ON loan.customer_id = customer.customer_id AND loan.employee_id
=employee.employee_id
```

```
MariaDB [bank]> SELECT * FROM leads;
+-----+-----+-----+-----+-----+
| loan_id | name           | type   | phone      | amount |
+-----+-----+-----+-----+-----+
| 22112512 | Sophronia Postins | Home   | 1164260229 | 380000 |
| 211251209 | Hilary Welband    | Home   | 3977766598 | 380000 |
| 211281154 | Shawna Corbin     | Home   | 1199864941 | 200000 |
| 211281205 | Elizabeth Farey   | Personal | 9368802991 | 300000 |
| 2147483647 | Haneyah Seemein   | Fund   | 9108630164 | 140000 |
+-----+-----+-----+-----+-----+
5 rows in set (0.008 sec)
```

# Trigger

```
CREATE TRIGGER update_bal
AFTER INSERT
ON Transaction for each row BEGIN update account set balance=balance+new.amount where
account.account_id=new.account_id; END $$
```

Before:

```
MariaDB [bank]> SELECT * FROM account where account_id=2020;
+-----+-----+-----+-----+-----+
| account_id | balance | type    | IFSC | customer_id |
+-----+-----+-----+-----+-----+
|          2020 |    20000 | Savings | 2000 |          9382 |
+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

After:

```
MariaDB [bank]> INSERT INTO transaction values (23232, 20221201, 400, 2020, 2642);
Query OK, 1 row affected (0.028 sec)

MariaDB [bank]> SELECT * FROM account where account_id=2020;
+-----+-----+-----+-----+-----+
| account_id | balance | type    | IFSC | customer_id |
+-----+-----+-----+-----+-----+
|          2020 |    20400 | Savings | 2000 |          9382 |
+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

## FRONT-END

The USER-INTERFACE is a python-library (streamlit) based application, that mainly consists of 3 roles ie. Customer, Employee and Admin.

Customers, Employees and the Admin can login into the accounts with their uniqueIDs. The demonstration with screenshots is followed in the next slides.

P.T.O



## Login Page

← → ↻ 🏠 🌐 localhost:8501

🔍 ☆ ⚙️ 🗖️ 👤 ⋮

☰

# Login

ID

0 - +

Role

☒ None

☐ Customer

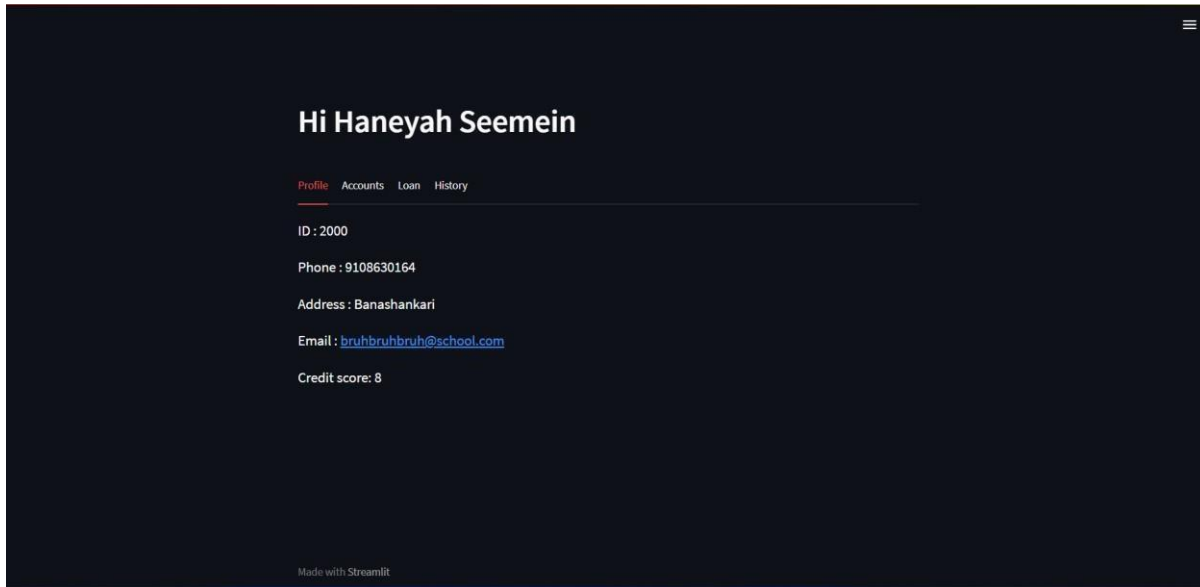
☐ Employee

☐ Admin

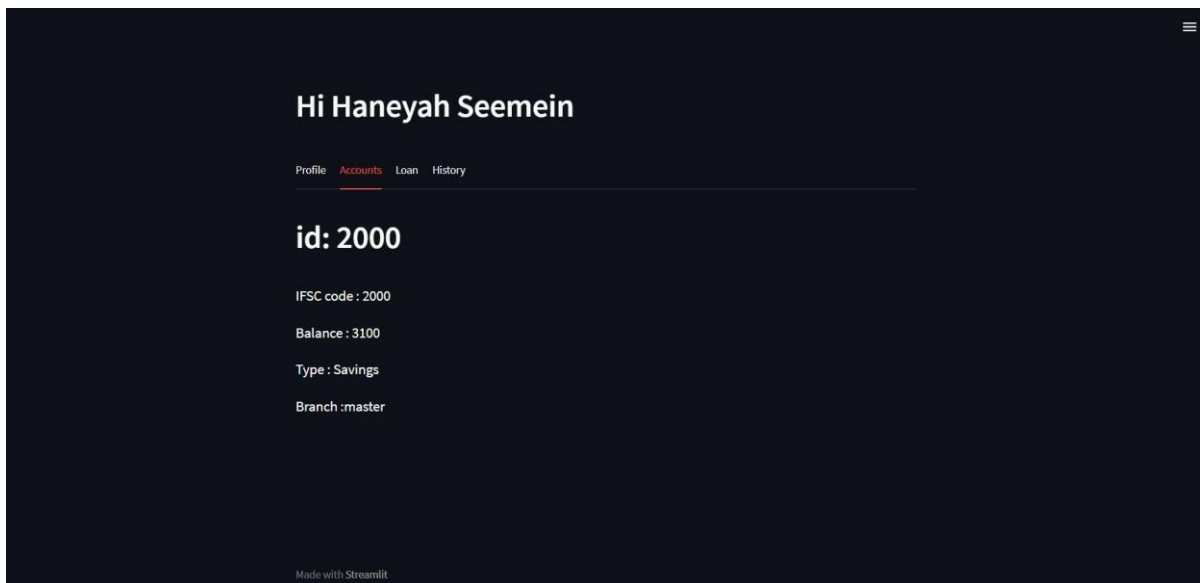
Made with Streamlit

## Customer's interface


### Profile



### Account



## Loan



## Hi Haneyah Seemein

Profile Accounts **Loan** History

---

Loan ID : 2147483647

Amount : 140000

Time : 3

Type : Fund

Total Amount : 140000


Assisted employee name : Hazel Jills

Helpline : 7526358691

Made with Streamlit

Press using your microphone  
Your assistant is ready to respond

## History



## Hi Haneyah Seemein

Profile Accounts Loan **History**

---

### Recieved

**23990**

Amount: ₹300---Date : 20221125

**44505**

Amount: ₹1400---Date : 20221128

**47431**

Amount: ₹1400---Date : 20221125

**81361**



Amount: ₹800---Date : 20221125

### Sent

**23990**

Amount: 300

Date : 20221125

**81361**

Amount: 500

Date : 20221125

**87747**

Amount: -200000

Date : 20221125

## Employee's interface

### Transaction

# Employee

HELLO Hazel Jills

Transaction Loan Account Customers

☒ Credit  
☐ Debit

Customer's account ID  
0

Customer's ID  
10000

Amount  
1000

transact

### Loan

Transaction Loan Account Customers

Amount  
200000

time(in years)  
2

Type  
Home

customer's ID  
0

branch's IFSC code  
1000

Apply

## Account

Transaction

Loan

Account

Customers

Account ID

0

-

+

Amount

200000

-

+

Type

Savings

▼

branch's IFSC code

1000

-

+

customer's ID

0

-

+

Create

## Customers

Employee

HELLO Hazel Jills

Transaction

Loan

Account

Customers

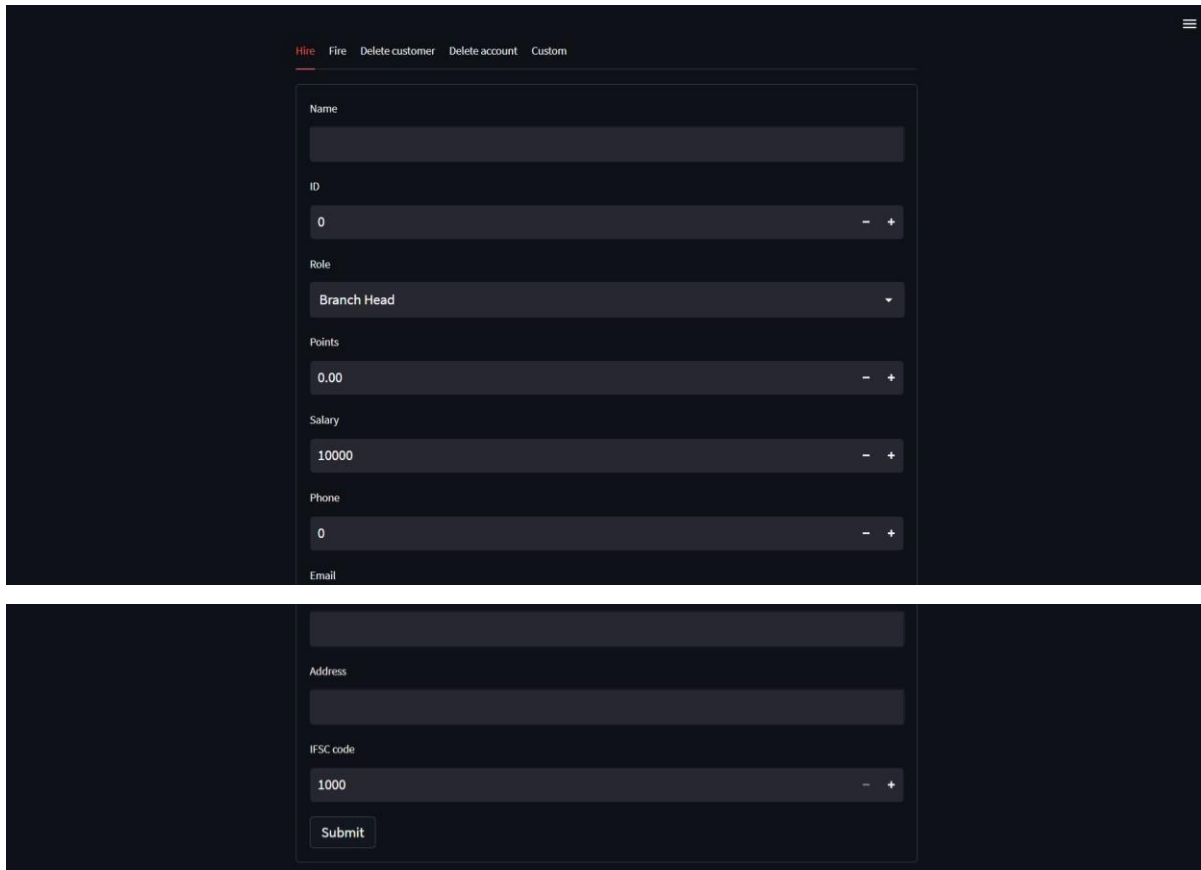
loans

	0	1	2	3	4	5
0	22112512	Sophronia Postlins	Home	1164260229	380000	11
1	211251209	Hilary Welband	Home	3977766598	380000	11
2	211281154	Shawna Corbin	Home	1199864941	200000	11
3	2147483647	Haneyah Seemeln	Fund	9108630164	140000	11

Made with Streamlit

## Admin's Interface

### Hire



The screenshot displays the 'Hire' form within an admin interface. At the top, a navigation bar includes links for 'Hire', 'Fire', 'Delete customer', 'Delete account', and 'Custom'. The form itself is a light gray box with a white background, containing several input fields and a dropdown menu. The fields are labeled 'Name', 'ID', 'Role', 'Points', 'Salary', 'Phone', and 'Email'. The 'ID', 'Points', 'Salary', and 'Phone' fields have numeric input with minus and plus buttons. The 'Role' field is a dropdown menu currently showing 'Branch Head'. Below these fields are two more text input fields for 'Address' and 'IFSC code'. The 'IFSC code' field also has numeric input with minus and plus buttons. A 'Submit' button is located at the bottom left of the form.

[Hire](#) [Fire](#) [Delete customer](#) [Delete account](#) [Custom](#)

Name

ID

0 - +

Role

Branch Head

Points

0.00 - +

Salary

10000 - +

Phone

0 - +

Email

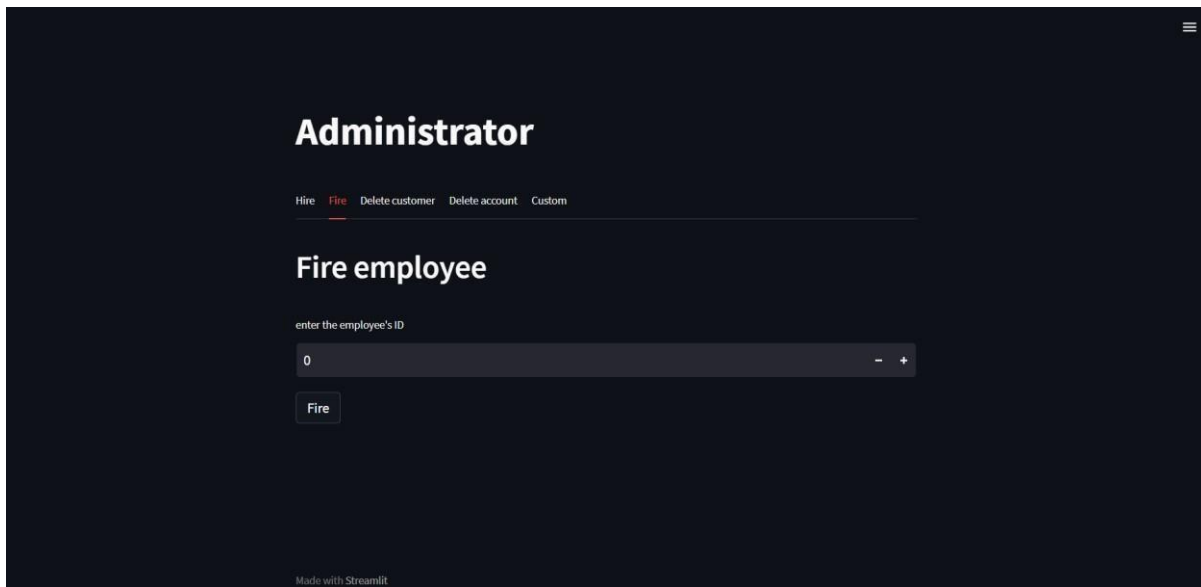
Address

IFSC code

1000 - +

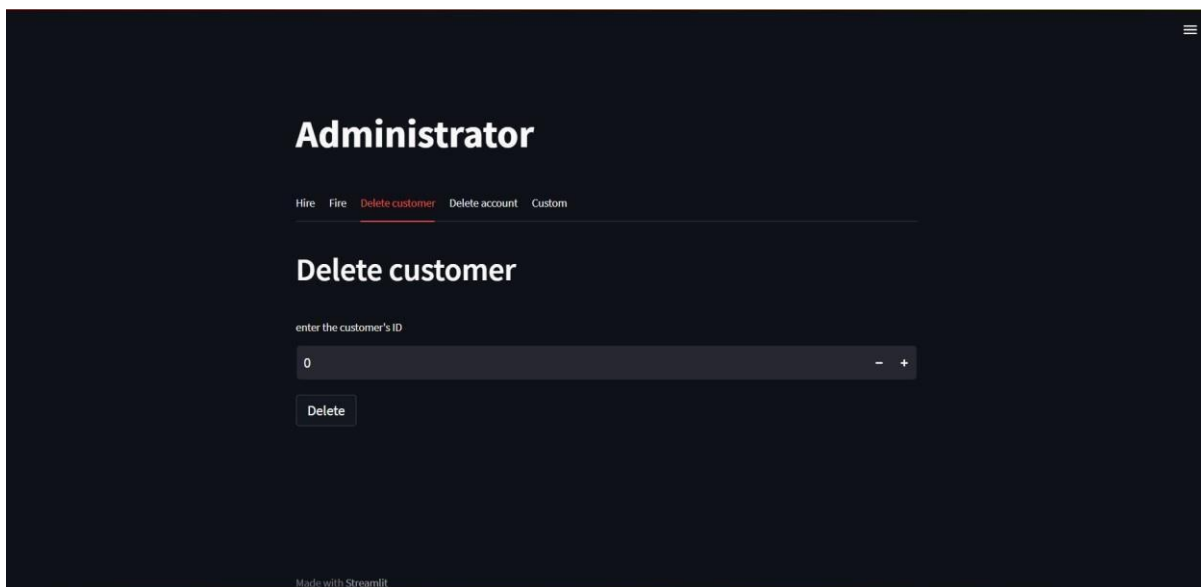
Submit

## Fire



The screenshot shows the 'Administrator' interface with a dark theme. At the top, there is a navigation bar with links: 'Hire', 'Fire' (highlighted in red), 'Delete customer', 'Delete account', and 'Custom'. Below the navigation bar, the main heading is 'Fire employee'. Underneath, there is a text input field labeled 'enter the employee's ID' containing the number '0'. To the right of the input field are minus and plus icons. Below the input field is a button labeled 'Fire'. At the bottom left, there is a small text 'Made with Streamlit'.

## Delete customer



The screenshot shows the 'Administrator' interface with a dark theme. At the top, there is a navigation bar with links: 'Hire', 'Fire', 'Delete customer' (highlighted in red), 'Delete account', and 'Custom'. Below the navigation bar, the main heading is 'Delete customer'. Underneath, there is a text input field labeled 'enter the customer's ID' containing the number '0'. To the right of the input field are minus and plus icons. Below the input field is a button labeled 'Delete'. At the bottom left, there is a small text 'Made with Streamlit'.

## Delete account

The screenshot shows the 'Administrator' dashboard with a navigation bar containing 'Hire', 'Fire', 'Delete customer', 'Delete account' (highlighted in red), and 'Custom'. The main heading is 'Delete account'. Below it, a text input field is labeled 'enter the account's ID' and contains the value '0'. To the right of the input field are minus and plus icons. A 'Delete' button is positioned below the input field. At the bottom left, it says 'Made with Streamlit'.

Administrator

Hire Fire Delete customer **Delete account** Custom

## Delete account

enter the account's ID

0 - +

Delete

Made with Streamlit

## Custom

The screenshot shows the 'Administrator' dashboard with a navigation bar containing 'Hire', 'Fire', 'Delete customer', 'Delete account', and 'Custom' (highlighted in red). The main heading is 'All operations'. Below it, a text input field is labeled 'enter your query' and contains the SQL query 'select \* from branch'. A 'Run' button is positioned below the input field. Below the button, a table displays the results of the query.

Administrator

Hire Fire Delete customer Delete account **Custom**

## All operations

enter your query

select \* from branch

Run

	0	1	2
0	Bangalore	1000	inside Mantri Square
1	Bangalore	2000	Banashankari
2	Bombay	3000	Juhu Beach
3	Delhi	4000	Lajpath Nagar
4	Kolkatta	5000	RT Port