BANKING SYSTEM

Prepared By:
Mandeep Kaur (c0788049)
Kamalpreet Kaur (c0793789)
Nency Shobhashana (c0787472)
Sahil Kashyap (c0789962)

Index

Introduction of our banking system

Bank is a place where clients feel a feeling of well-being for their assets. In the bank, customers deposit and withdraw their money. Exchange of cash additionally is where clients take cover in the bank. Presently to keep the conviction and trust of clients, there is the positive requirement for the management of the bank, which can handle all this with comfort and ease. These days, dealing with a bank is a monotonous activity up to certain limits. The present world is a certified computer world and is getting quicker and quicker step by step. The banking application that reduces work is essential. In this way, considering the above necessities, the application for bank executives has become fundamental which would be valuable in dealing with the bank all the more proficiently.

The banking management system is one of the most complex systems because the things it covers under the roof for transparency among the customers. From dealing with the client data, account data to the exchange happening each moment or second. It does not only preserve the details of the transaction and other information but generates the report to further banking functions.

In this banking management system, there are numerous activities which are robotized to facilitate the work for the working of the bank. It reduces the requirement for the manual labor and the automated tasks will be error free as they will only work as they are programmed whereas doing work manually there is always a possibility of human error.

EXISTING SYSTEM OF BANKING MANAGEMENT SYSTEM:

The existing bank system is slow as every task is being performed by the human being and comparing the computer task speed with a computer is not fair. The complexity of this system is increased when their increase in a number of customers and with that there will be number of transactions will be performed now all that requirements to sign in a record for the reference later on which is just not the sort of situation we need right now.

Some other drawbacks of the existing system:

Less security of customer and bank information.
Require more physical work and manpower.
All the manual entry and editing will take more time.
No level of clearance for the different levels of employees.
Safety of paper documents from the disaster.
No backup of the information.

Some improvements by executing the proposed system:

More protected information as it will give a layer of security of authentication and authorization.
Required very less manpower.
Simplify the problem of editing.
More reliable and efficient.
More user-friendly interface.

Description of the system

The banking system is a project for maintaining a customer's account in the bank. The system provides the access to the customer to create an account, issues debit card and credit card, borrow loan if needed, also to view reports of all account present. It is used to keep the records of customers, employee etc in bank to develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks.

As it analyse most of the manual task for the bank but there is a major perturb over the security of the data and resources of the customer so it is very important to keep up with security features and tested each module carefully while deploying.

The main intention of this system is to provide a secure system. Our system is password protected and it only allows official user to access various functions available in the system. This digitization of the bank will help the bank in every condition of the growth. It will not only make the work elementary but significantly improve the speed of work as there are no physical files or data sheets will be there to manage everything will be managed logically with the system and machine.

The information about the customer or if the customer wants to know their data it will be just some clicks away and with an rise in transparency between the customer and fast service it will undoubtedly get the trust of customers.

Banking system entities and their Attributes:

Customer Entity: Attributes of Customer Entity are cust_id, name, address, contact, email, pan_no, dob and gender.
Cust_id is the primary key
Name is a composite attribute which includes f_name and l_name.
Address is a composite attribute which further includes pin,street,state and city

Account Entity: Attributes of Account Entity are acc_no, acc_type and acc_bal.
Acc_no is Primary Key.

Loan Account Entity: Attributes of Loan Entity are loan_id, loan_type, remain_amt, duration, start_date, interest and amount.
Loan_id is Primary Key for Loan Entity.

Employee Entity: Attributes are emp_id, name, dob, gender, contact and address.
Emp_id is the primary key
Name is a composite attribute which includes f_name and l_name.
Address is a composite attribute which further includes pin,street,state and city

Payment Entity: Attributes of payment Entity are pay_id, pay_amt and pay_date.
Pay_id is the primary key.

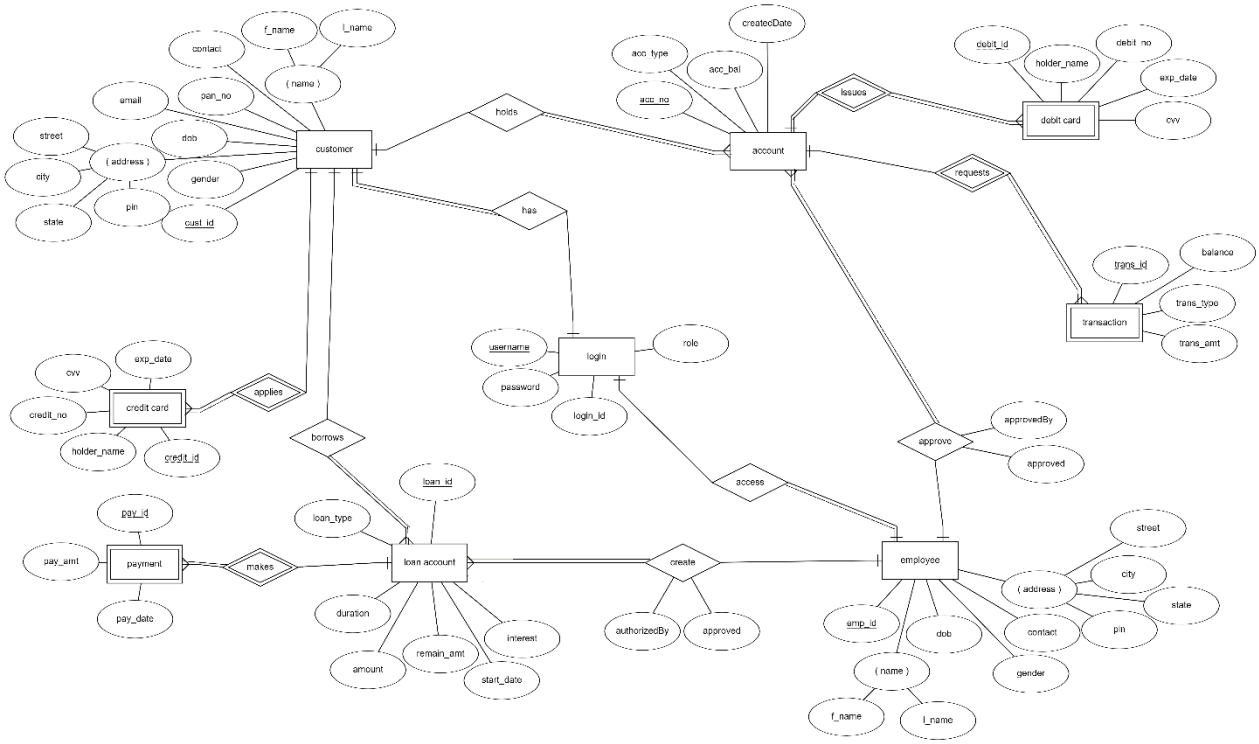
Transaction Entity: Attributes of transaction Entity are trans_id, trans_type, trans_amt and balance.
Trans_id is the primary key.

Debit card Entity: Attributes are debit_id, debit_no, holder_name, cvv, exp_date.
Debit_no is the primary key.

Credit card entity: Attributes are credit_id, credit_no, holder_name, cvv, exp_date.
Credit_no is the primary key.

ER Diagram of Banking System

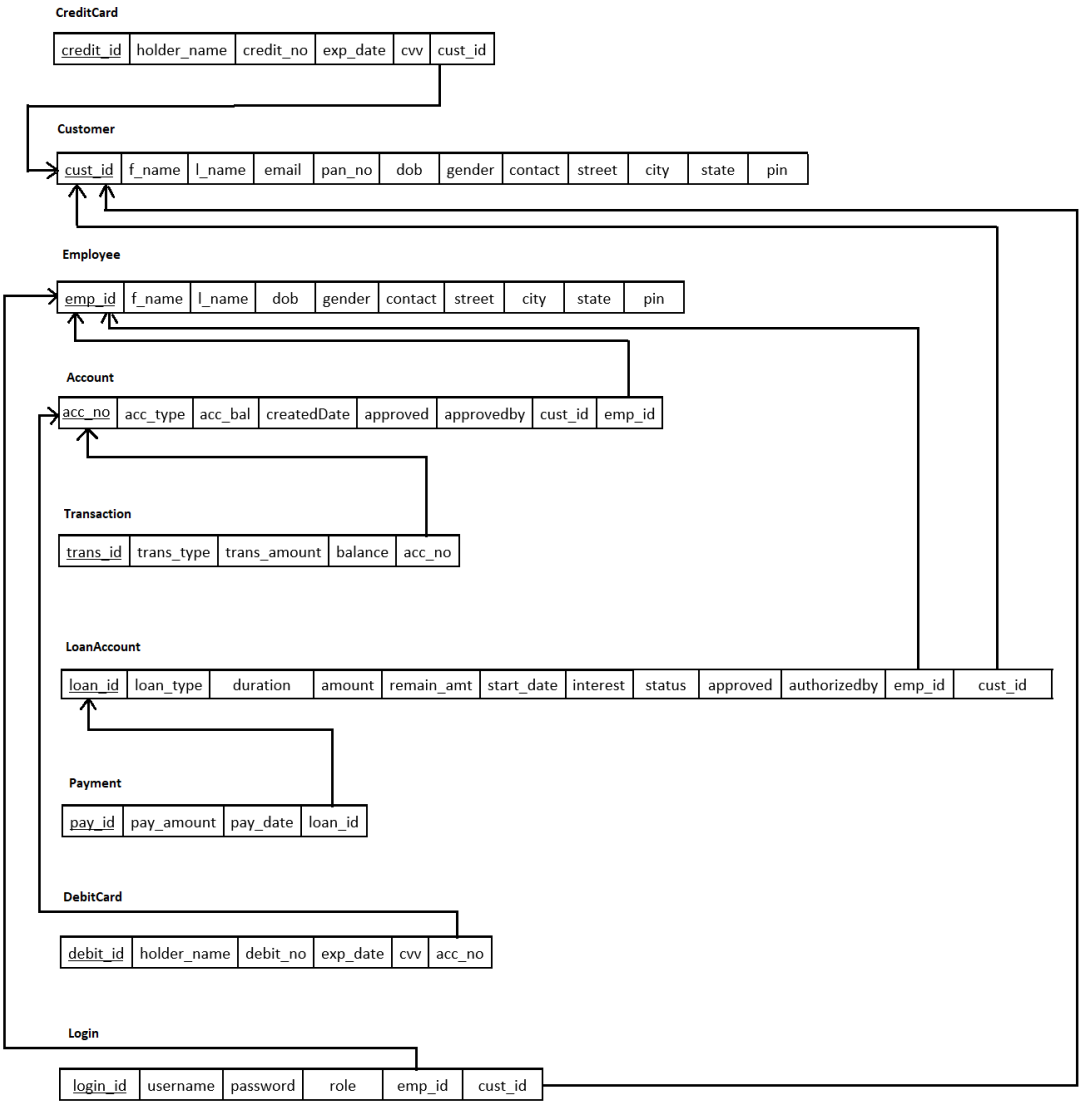
ER diagram is known as Entity-Relationship diagram. It is used to evaluate the structure of the Database. It shows relationships between entities and their attributes.



Mapping of the ER to Relational Schema

Mapping Process

I. Create table for each entity.
II. Entity's attributes should become fields of tables with their respective data types.
III. Declare primary key.



Normalization

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

1NF (First Normal Form) Rules

- ✓ Each table cell contains a single value.
- ✓ Each record needs to be unique.

We re-arrange the relation (table) as below, to convert it to First Normal Form(1NF):

customer

cust_id	f_name	l_name	email	pan_no	dob	gender	contact	street	city	state	pin

employee

emp_id	f_name	l_name	dob	gender	contact	street	city	state	pin

account

acc_no	acc_type	acc_bal	createdDate	approved	approvedby	cust_id	emp_id

transaction

trans_id	trans_type	trans_amount	balance	acc_no

Loan account

loan_id	loan_type	duration	amount	remain_amt	start_date	interest	approved
authorizedby	emp_id	cust_id					

Payment

pay_id	pay_amount	pay_date	loan_id

Debit card

debit_id	holder_name	debit_no	exp_date	cvv	acc_no

Credit card

credit_id	holder_name	credit_no	exp_date	cvv	cust_id

Login

login_id	username	password	role	emp_id	cust_id

2NF (Second Normal Form) Rules

- ✓ Be in 1NF

- ✓ Single Column Primary Key

As above given relation(tables) is passed by 2NF Rules, so it is already in 2NF.

3NF (Third Normal Form) Rules

- ✓ Be in 2NF
- ✓ Has no transitive functional dependencies

As above given relations(tables) is also passed by 3NF rules with no transitive functional dependencies, so it is already in 3NF form.

When we apply add, update and delete anomalies, our system is losing any kind of partial data. That is reason to pass all of three normalization and not need to redo of relational model.

DATA DESCRIPTION

Customer

Column	Datatype	length	PK	FK(references)	Constraints	Mask
cust_id	Integer	5	Yes	No	Not null,Unique	-
f_name	Text	15	No	No	Not null	-
l_name	Text	15	No	No	Not null	-
Email	Text	50	No	No	Not null	-
pan_no	Text	10	No	No	Not null	-
Dob	Date	10	No	No	Check	yyyy-mm-dd
Gender	Text	6	No	No	Not null	-
State	Text	10	No	No	Not null	-
City	Text	10	No	No	Not null	-
Street	Text	20	No	No	Not null	-
Pin	Integer	6	No	No	Not null	-
contact	Integer	10	No	No	Not null	-

Employee

Column	Datatype	length	PK	FK(references)	Constraints	mask
emp_id	Integer	5	Yes	No	Not null, Unique	-
f_name	Text	15	No	No	Not null	-
l_name	Text	15	No	No	Not null	-
contact	Integer	10	No	No	Not null	-
dob	Date	10	No	No	Check	yyyy-mm-dd
gender	Text	6	No	No	Not null	-
state	Text	10	No	No	Not null	-
city	Text	10	No	No	Not null	-
street	Text	20	No	No	Not null	-
pin	Integer	6	No	No	Not null	-

Account

Column	Datatype	length	PK	FK(references)	Constraints	Mask
cust_id	Integer	5	No	Customer.cust_id	Not null	-
acc_no	Integer	10	Yes	No	Not null, Unique	-
acc_type	Text	10	No	No	Not null	-
emp_id	Integer	5	No	Employee.emp_id	Not null	-
acc_bal	Integer	10	No	No	Not null	-
createdDate	Date	10	No	No	Check	Yyyy-mm-dd
Approvedby	Integer	5	No	Employee.emp_id	-	-
Approved	bool	1	No	No	Not null	-

Transaction

Column	Datatype	length	PK	FK(references)	Constraints	Mask
acc_no	Integer	10	No	Account.acc_no	Not null	-
trans_id	Integer	10	Yes	No	Not null, Unique	-
trans_amt	Integer	10	No	No	Not null	-
trans_type	Text	10	No	No	Not null	-
balance	Integer	10	No	No	Not null	-
trans_date	Date	10	No	No	Not null	-

Loan_Account

Column	Datatype	length	PK	FK(references)	Constraints	mask
loan_id	Integer	10	Yes	No	Not null, Unique	-
cust_id	Integer	5	No	Customer.Cust_id	Not null	-
emp_id	Integer	5	No	Emp_manager.Emp_id	-	-
authorizedBy	Integer	5	No	Emp_manager.Emp_id	-	-
approved	Bool	1	No	No	-	-
loan_type	Text	10	No	No	Not null	-
interest	Number	3	No	No	Not null	-
duration	Number	3	No	No	Not null	-
amount	Number	10	No	No	Not null	-
start_date	Date	10	No	No	Check	yyyy-mm-dd
remain_amt	Number	10	No	No	Not null	-

Payment

Column	Datatype	length	PK	FK(references)	Constraints	Mask
loan_id	Integer	10	No	Loan_Account.loan_id	Not null	-
pay_id	Integer	10	Yes	No	Not null, Unique	-
pay_date	Date	10	No	No	Not null	yyyy-mm-dd
pay_amt	Integer	10	No	No	Not null	-

Login

Column	Datatype	Length	PK	FK(references)	Constraints	mask
username	Text	10	No	No	Not null	-
password	Text	10	No	No	Not null	-
role	Text	1	No	No	Not null	-
login_id	Integer	5	Yes	No	Not null, Unique	-
emp_id	Integer	5	No	Employee.emp_id	-	-
Cust_id	Integer	5	No	Customer.cust_id	-	-

Debit_card

Column	Datatype	length	PK	FK(references)	Constraints	mask
acc_no	Integer	10	No	Account.Acc_no	Not null	-
debit_id	Integer	10	Yes	No	Not null, Unique	-
debit_no	Number	16	Yes	No	Not null, Unique	-
exp_date	Date	10	No	No	Check	yyyy-mm-dd
cvv	Number	3	No	No	Not null	-
holder_name	Text	20	No	No	Not null	-

Credit_card

Column	Datatype	length	PK	FK(references)	Constraints	mask
cust_id	Integer	10	No	Customer.cust_id	Not null	-
credit_id	Integer	10	Yes	No	Not null, Unique	-
credit _no	Number	16	Yes	No	Not null, Unique	-
exp_date	Date	10	No	No	Check	yyyy-mm-dd
cvv	Number	3	No	No	Not null	-
holder_name	Text	20	No	No	Not null	-

SQL Statements

Create table login
(

```
  loginId INT(5) not null AUTO_INCREMENT,
  username varchar(10) not null,
  password varchar(10) not null,
  role char(1) not null,
  cust_id INT(5),
  emp_id INT(5),
  PRIMARY KEY (loginId),
  FOREIGN KEY (cust_id) REFERENCES customer(cust_id),
  FOREIGN KEY (emp_id) REFERENCES employee(emp_id)
);

SELECT * from login;

INSERT INTO login(username, password, role, emp _id) values
(concat('emp_', LAST_INSERT_ID()) ,
concat(char(round(rand()*25)+97),
char(round(rand()*25)+97),char(round(rand()*25)+97),char(round(rand()*25)+97),ch
ar(round(rand()*25)+97),char(round(rand()*25)+97)), '1', LAST_INSERT_ID());
```

CREATE TABLE customer

```
(
  cust_id INT(5)not null AUTO_INCREMENT,
  f_name varchar(15) not null,
  l_name varchar(15) not null,
  email varchar(50) not null,
  pan_no varchar(10) not null,
  dob date not null,
  gender varchar(6) not null,
  street varchar(20) not null,
  city varchar(10) not null,
  state varchar(10) not null,
  pin INT(6) not null,
  contact BIGINT(10) not null,
  PRIMARY KEY (cust_id),
  CHECK (dob < ‘2020-09-28’)
);
ALTER TABLE customer AUTO_INCREMENT = 10000;

INSERT INTO customer (f_name, l_name, email, pan_no, dob, gender, street, city,
state, pin, contact) VALUES
(’deep’,’sharma’,’dsharma@gmail.com’,’1234abcd90’,’1999-04-12’,’female’, ‘main
street’, ‘amritsar’, ‘punjab’, 400123, 9876543210);
```

UPDATE customer set city =’patiala’ where cust_id = 24567;

DELETE from customer where pan_no =’1234abc90’;

SELECT *, DATE_FORMAT(dob, '%Y-%m-%d') as dob, CONCAT_WS(', ', street, city, state, pin) AS address from customer;

SELECT *, DATE_FORMAT(dob, '%Y-%m-%d') as dob, CONCAT_WS(', ', street, city, state, pin) AS address from customer WHERE f_name = ’deep’;

CREATE TABLE employee

```
(
  emp_id INT(5)not null AUTO_INCREMENT,
  f_name varchar(15) not null,
  l_name varchar(15) not null,
  dob date not null,
  gender varchar(6) not null,
  street varchar(20) not null,
  city varchar(10) not null,
  state varchar(10) not null,
  pin INT(6) not null,
  contact BIGINT(10) not null,
  PRIMARY KEY (emp_id),
  CHECK (dob < ‘2020-09-28’)
);
ALTER TABLE employee AUTO_INCREMENT = 10000;
```

INSERT INTO employee (f_name, l_name, dob, gender, street, city, state, pin, contact) VALUES (’prince’,’verma’,’2000-01-23’,’male’, ‘oak st’, ‘surat’, ‘gujrat’, 300120, 9870654132);

UPDATE employee set l_name =’virk’ where emp_id = 10000;

DELETE from employee where city =’surat’;

SELECT *, DATE_FORMAT(dob, '%Y-%m-%d') as dob, CONCAT_WS(', ', street, city, state, pin) AS address from employee;

SELECT *, DATE_FORMAT(dob, '%Y-%m-%d') as dob, CONCAT_WS(', ', street, city, state, pin) AS address from employee WHERE f_name = ’prince’;

CREATE TABLE credit_card

```
(
  credit_id BIGINT(10) not null AUTO_INCREMENT,
  holder_name varchar(20) not null,
```

```
  credit_no BIGINT(16) not null,
  exp_date date not null,
  cvv INT(3) not null,
  cust_id INT(5),
  PRIMARY KEY (credit_id),
  FOREIGN KEY (cust_id) REFERENCES customer(cust_id),
);
ALTER TABLE credit_card AUTO_INCREMENT = 1000000000;

INSERT INTO credit_card ( credit_no, exp_date, holder_name, cvv, cust_id) VALUES
( 2341098734560987, ‘2022-03-26’, ‘hema’, 890, 10000);

UPDATE credit_card set cvv = 588 where cust_id = 10000;

DELETE from credit_card where holder_name =’hema’;

SELECT * from credit_card;

SELECT * from credit_card WHERE cust_id=10000;
```

CREATE TABLE account

```
(
  acc_no BIGINT(10)not null AUTO_INCREMENT,
  acc_type varchar(10) not null,
  acc_bal BIGINT(10) not null,
  createdDate date not null,
  cust_id INT(5),
  emp_id INT(5),
  approvedBy INT(5),
  approved BOOL,
  PRIMARY KEY (acc_no),
  FOREIGN KEY (cust_id) REFERENCES customer(cust_id),
  FOREIGN KEY (emp_id) REFERENCES employee(emp_id),
  FOREIGN KEY (approvedBy) REFERENCES employee(emp_id),
  CHECK (createdDate > ‘2020-09-28’)
);
ALTER TABLE account AUTO_INCREMENT = 1000000000;

INSERT INTO account ( acc_type, acc_bal, createdDate, cust_id, emp_id, approved)
VALUES ( ‘saving’, 12000, 2020-09-28, 10000, 10000, 0);

UPDATE account set acc_bal = 25000 where acc_no = 0800122345;

DELETE from account where acc_type = ‘saving’;

SELECT * from account WHERE acc_no = 1000000000 AND approved =1;
```

```
SELECT * from account WHERE cust_id = 10000 AND approved =1;
```

CREATE TABLE debit_card

```
(
  debit_id BIGINT(10) not null AUTO_INCREMENT,
  debit_no BIGINT(16) not null,
  holder_name varchar(20) not null,
  exp_date date not null,
  cvv INT(3) not null,
  acc_no BIGINT(10),
  PRIMARY KEY (debit_id),
  FOREIGN KEY (acc_no) REFERENCES account(acc_no)
);
ALTER TABLE debit_card AUTO_INCREMENT = 1000000000;

INSERT INTO debit_card (debit_no, holder_name, exp_date, cvv, acc_no) VALUES
(8907654678342160, 'rajan', '2024-12-28', 584, 1000000000);

UPDATE debit_card set cvv = 123 where debit_no = 8907654678342160;

DELETE from debit_card where holder_name = 'rajan';

SELECT * from debit_card;

SELECT * from debit_card WHERE acc_no IN (SELECT acc_no from account WHERE
cust_id = 10000);

SELECT * from debit_card WHERE acc_no=1000000000;
```

CREATE TABLE loan_account

```
(
  loan_id BIGINT(10) not null AUTO_INCREMENT,
  loan_type varchar(10) not null,
  interest INT(3) not null,
  duration INT(3) not null,
  amount BIGINT(10) not null,
  start_date date not null,
  remain_amt BIGINT(10) not null,
  cust_id INT(5),
  emp_id INT(5),
  authorizedBy INT(5),
  approved BOOL,
  PRIMARY KEY (loan_id),
  FOREIGN KEY (cust_id) REFERENCES customer(cust_id),
  FOREIGN KEY (emp_id) REFERENCES employee(emp_id),
  FOREIGN KEY (approvedBy) REFERENCES employee(emp_id),
```

```
  CHECK (start_date > '2020-09-28')
);
ALTER TABLE loan_account AUTO_INCREMENT = 1000000000;

 INSERT INTO loan_account ( loan_type, interest, duration, amount, start_date,
remain_amt, cust_id, emp_id, approved) VALUES ( 'car loan', 5, 12, 50000,
'2021-05-02', 20000, 10000, 10000, 0);

 UPDATE loan_account set interest = 5 where loan_id = 1000000000;

 DELETE from loan_account where loan_id = 1000000000;

 SELECT * from loan_account;

  SELECT * from loan_account WHERE cust_id = 1000000000 AND approved = 1
```

CREATE TABLE payment

```
(
  pay_id BIGINT(10) not null AUTO_INCREMENT,
  pay_date date not null,
  pay_amt BIGINT(10) not null,
  loan_id BIGINT(10),
  PRIMARY KEY (pay_id),
  FOREIGN KEY (loan_id) REFERENCES loan_account(loan_id)
);
ALTER TABLE payment AUTO_INCREMENT = 1000000000;

INSERT INTO payment ( pay_date, pay_amt, loan_id) VALUES ('2020-08-21', 5000,
1000000000);
UPDATE loan_account SET remain_amt= remain_amt - 5000 WHERE loan_id =
1000000000;

UPDATE payment set pay_date ='2020-08-21' where loan_id = 1000000000;

DELETE from payment where pay_id = 1000000000;

SELECT * from payment;

SELECT * from payment WHERE loan_id IN(SELECT loan_id from loan_account
WHERE cust_id = 10000);
```

CREATE TABLE transaction

```
(
  trans_id BIGINT(10) not null AUTO_INCREMENT,
  trans_date date not null,
  trans_amt BIGINT(10) not null,
  trans_type varchar(10) not null,
  balance BIGINT(10) not null,
  acc_no BIGINT(10),
  PRIMARY KEY (trans_id),
  FOREIGN KEY (acc_no) REFERENCES account(acc_no)
);
ALTER TABLE transaction AUTO_INCREMENT = 1000000000;


INSERT INTO transaction ( trans_date, trans_amt, trans_type, balance, acc_no)
VALUES (2020-09-21 , 4000, 'credit', (Select acc_bal + 4000 from account where
acc_no=1000000000) ,1000000000);

UPDATE account SET acc_bal=(SELECT balance from transaction where
trans_id=LAST_INSERT_ID()) WHERE acc_no = 1000000000 ;

SELECT * from transaction;

SELECT * from transaction where acc_no=1000000000;

SELECT * from transaction WHERE acc_no IN (SELECT acc_no from account WHERE
cust_id = 10000)
```

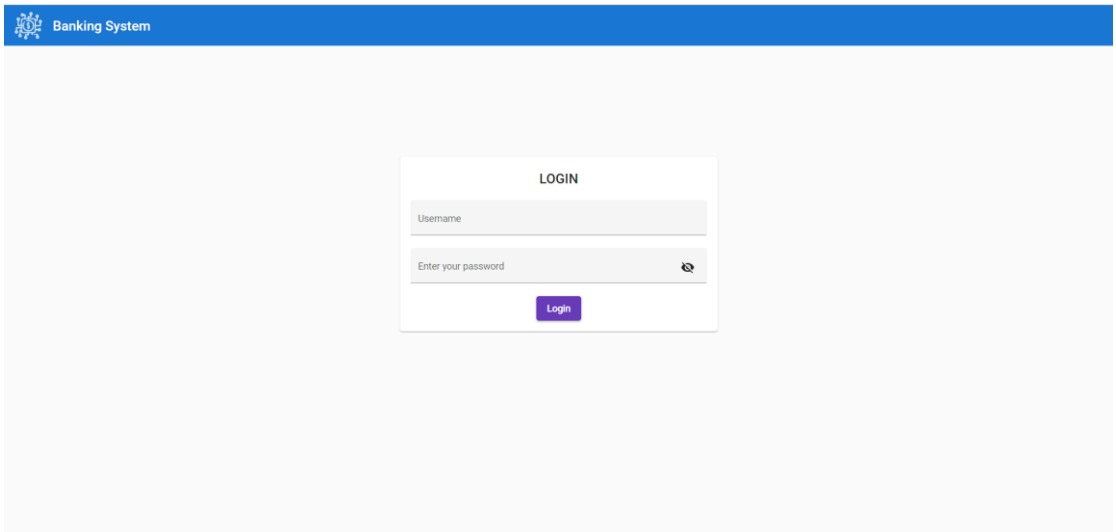
Our Application

We have created a web application using the Angular (version 10). Normal Angular system work as client at client side in browser. Application framework does not give support to communicate with any database directly. So, we have created one server which can work as bridge between application and database. The Server, we have implemented is in node using express library as well as mysql2 library to communicate with our MySQL database server.

Few screenshots with short description of our application is below:

Login Screen:

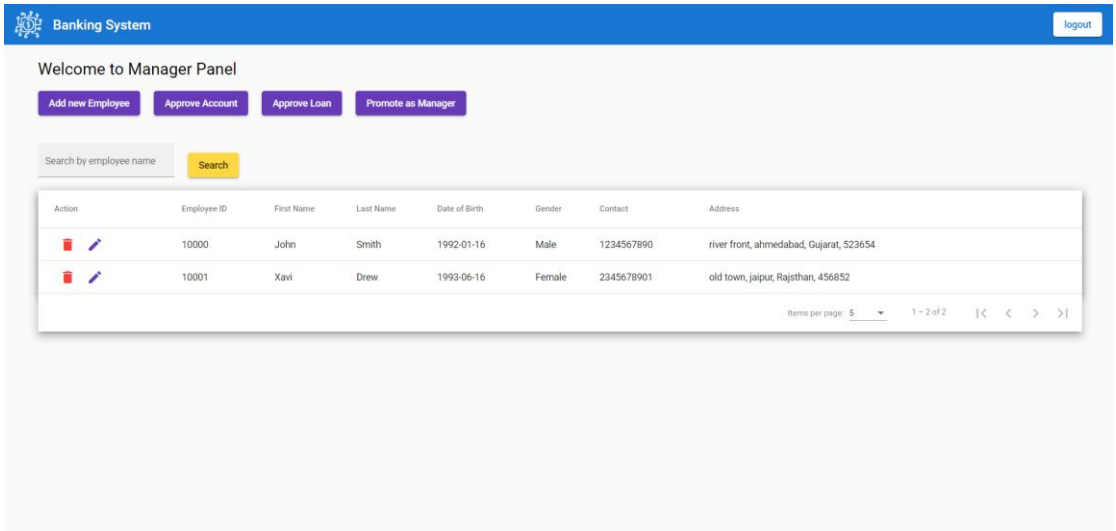
All employee as well as customer can use application via login through this screen. All person can login with username and password which shared with them via paper trail.



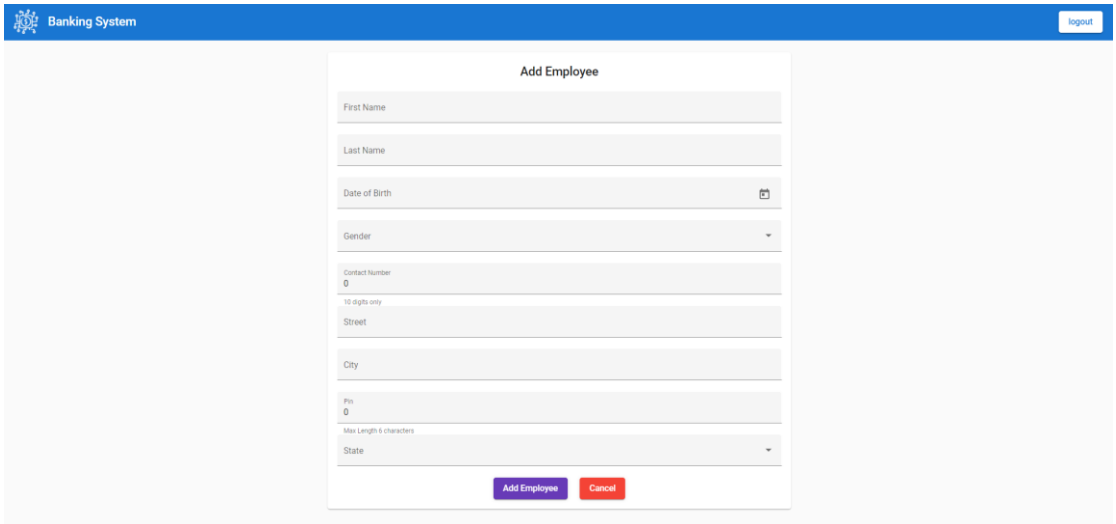
Login Screen

Admin (Manager) Panel:

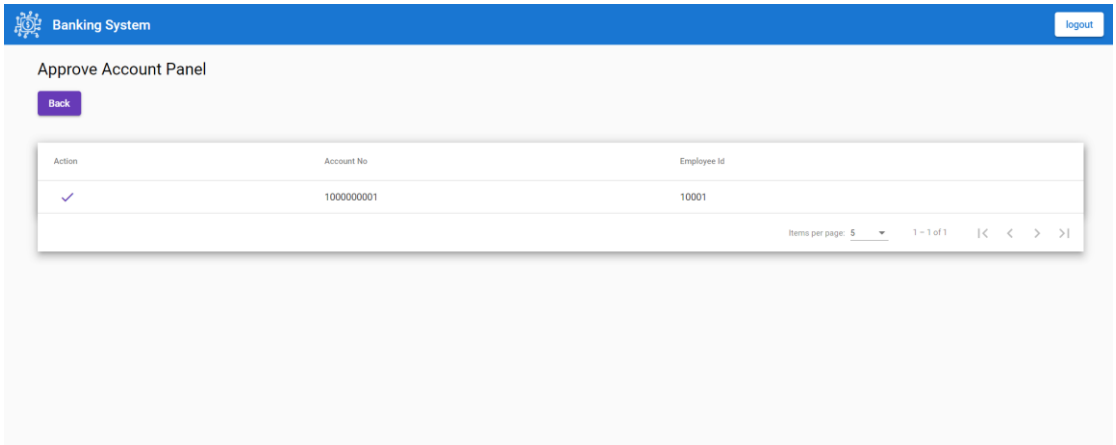
This is a manager panel in which he can add, update, delete employee, approve normal account, authorize loan account as well as he can promote any employee to manager.



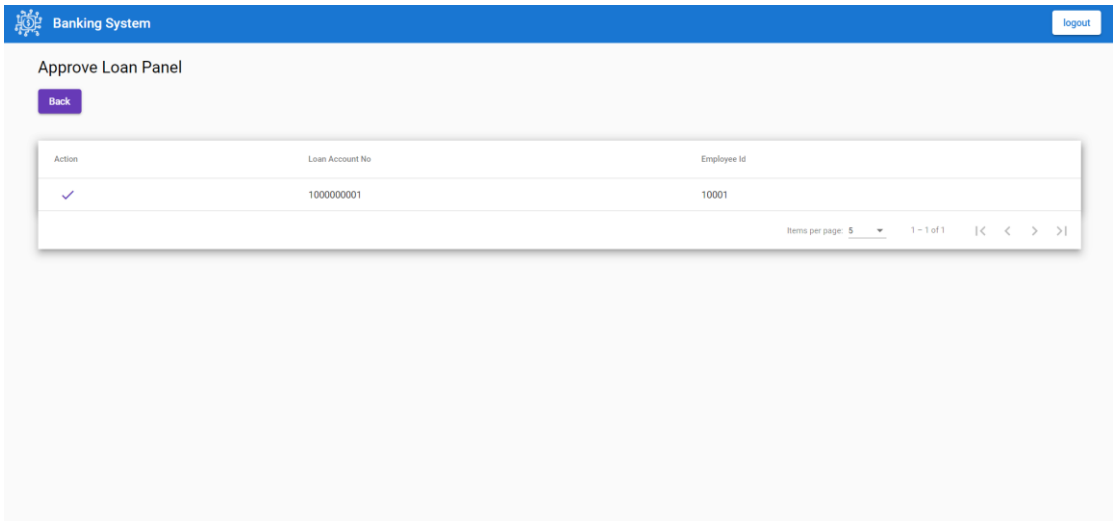
Manager Panel (List of employees and add, edit, delete and search functions for employee)



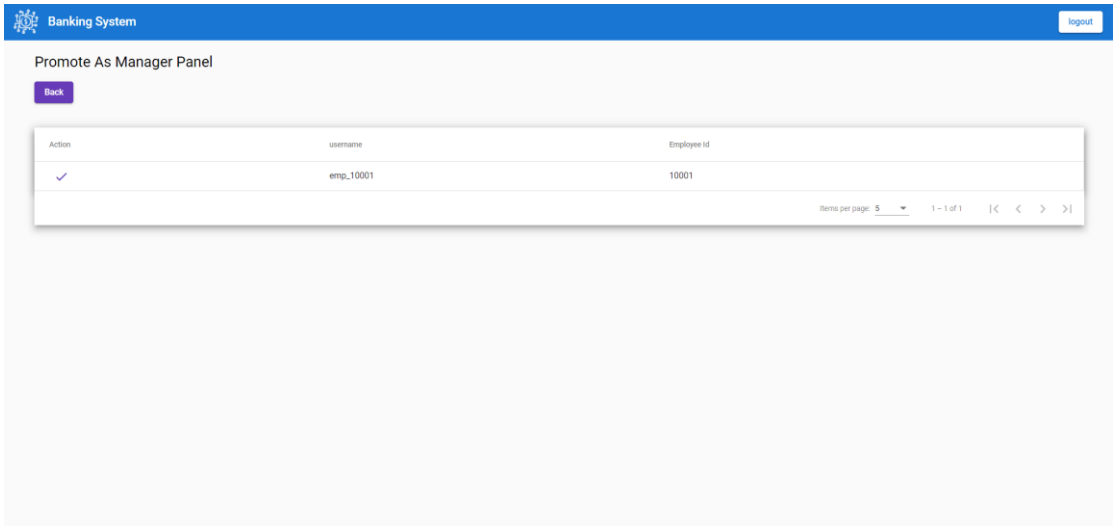
Add New Employee screen when click on **Add New employee** button



Approve Account screen (when click on **Approve Account** button screen show account requests**)**



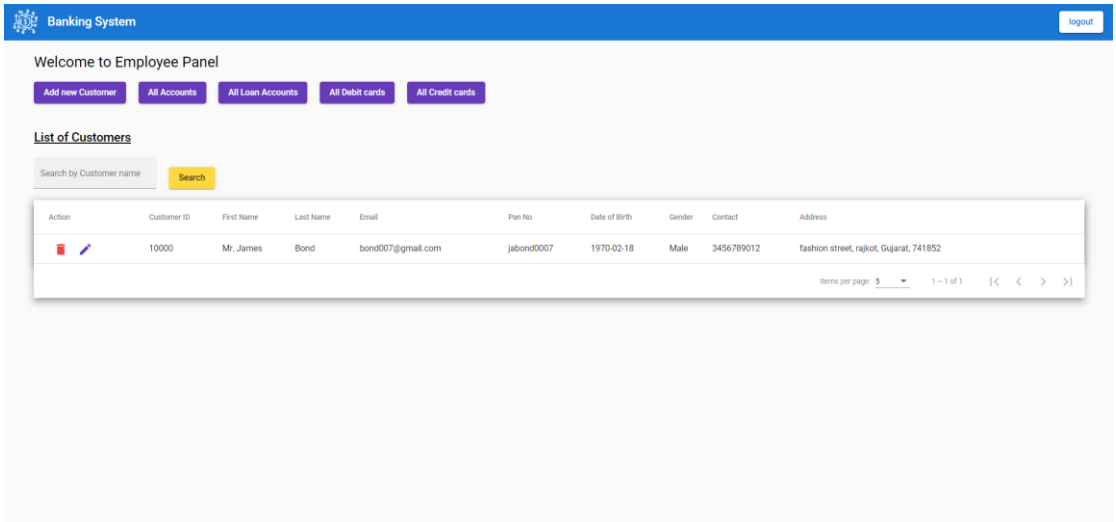
Approve Loan Account screen (when click on **Approve Loan Account** button screen show loan account requests**)**



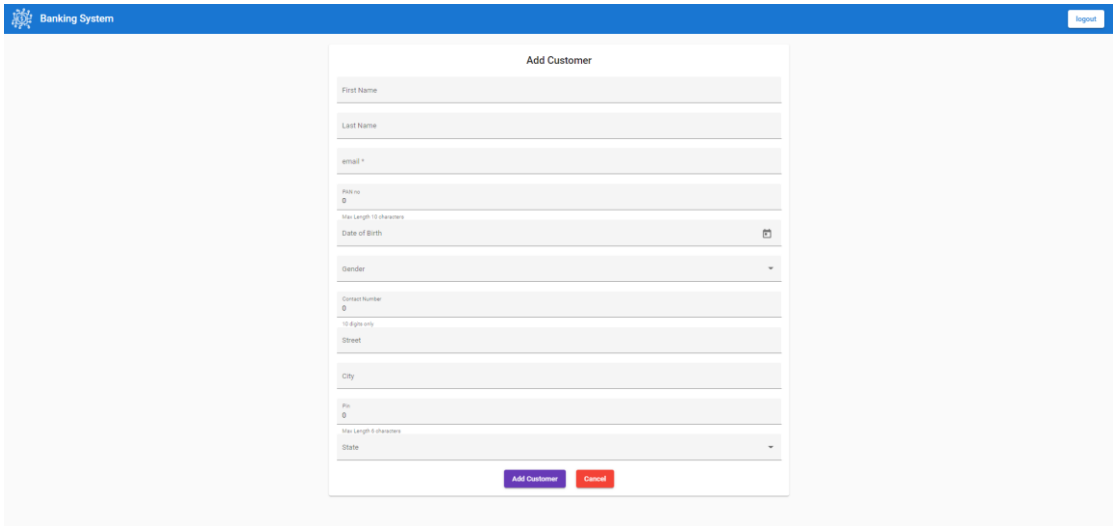
Manager Promoting screen (when click on **Promote as manager button)**

Employee Panel:

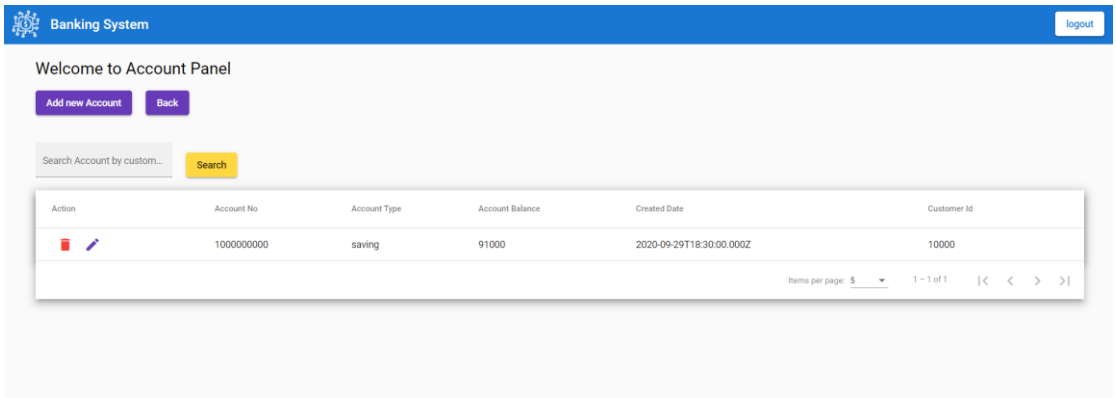
This is an employee panel view from which he can check all customer in system and add, update, and delete customer. He can also create account, loan account, debit card as well as credit card.



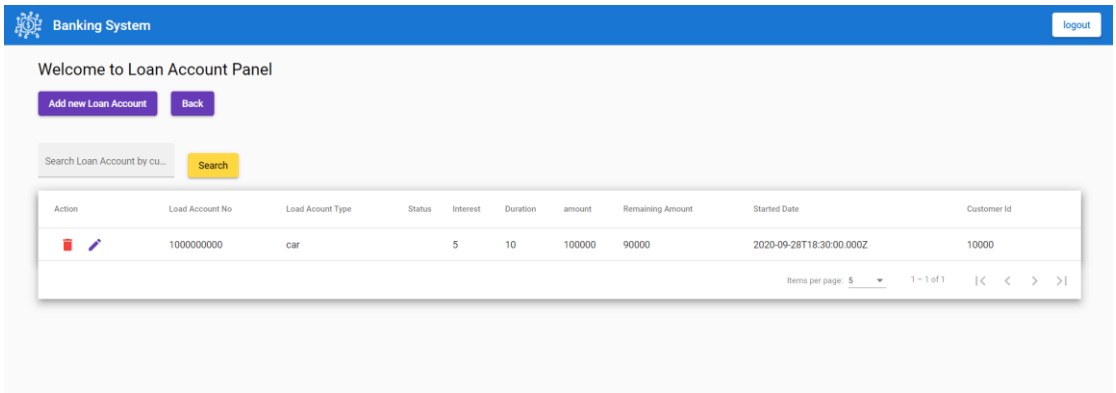
Customer Panel (List of customers and add, edit, delete and search functions for customers)



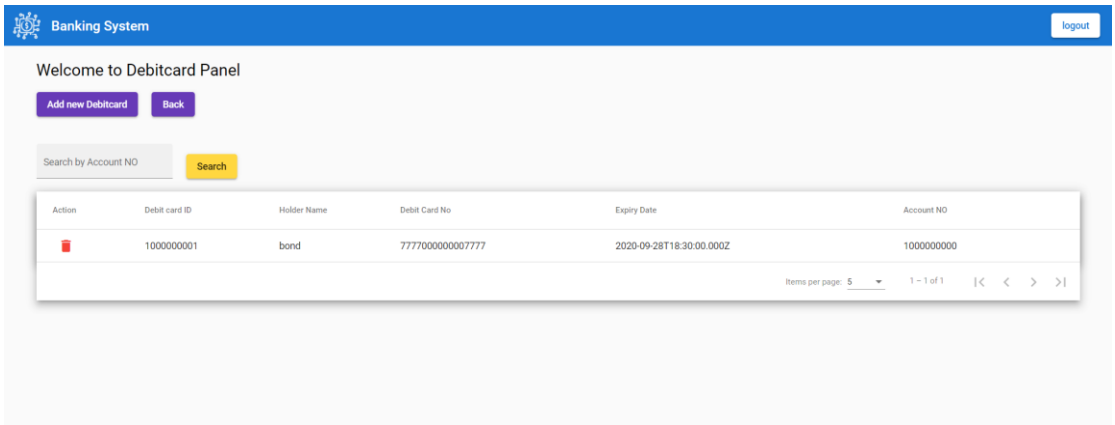
Add New customer screen when click on **Add New customer** button



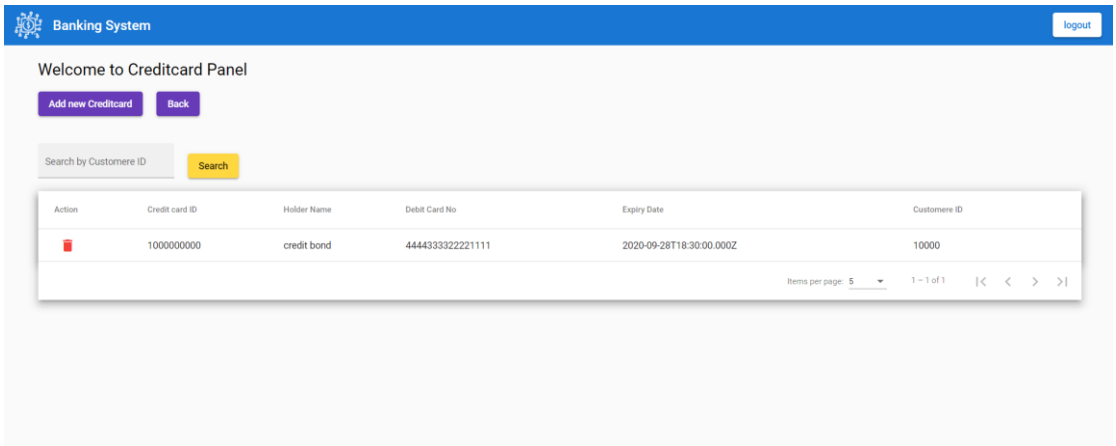
All approved account screen when click on **All Account** button (add, edit, delete and search function on account by employee)



All approved loan account screen when click on **All loan Account** button (add, edit, delete and search function on loan account by employee)



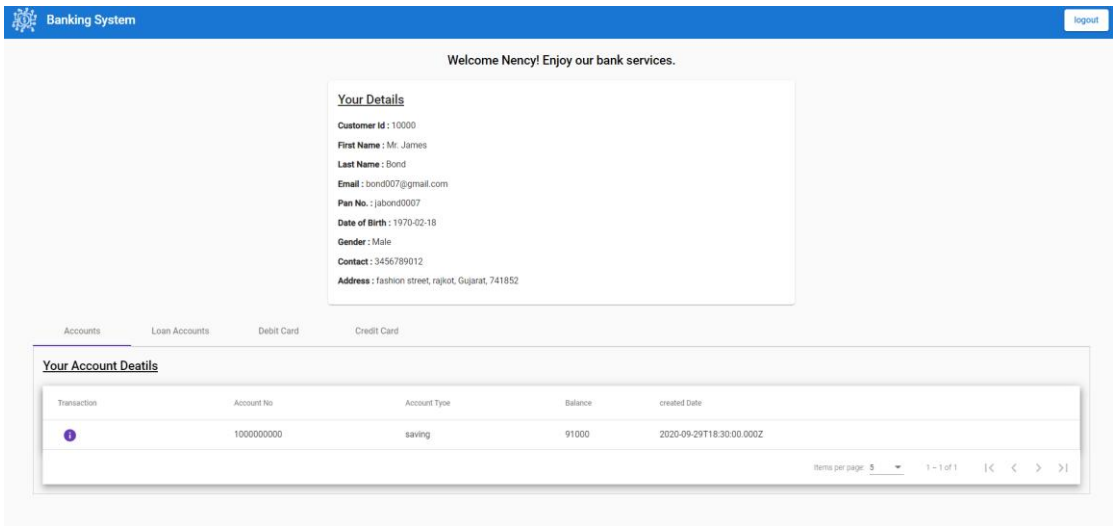
All debit card screen when click on **All Debit card** button (add, delete and search function on account no by employee)



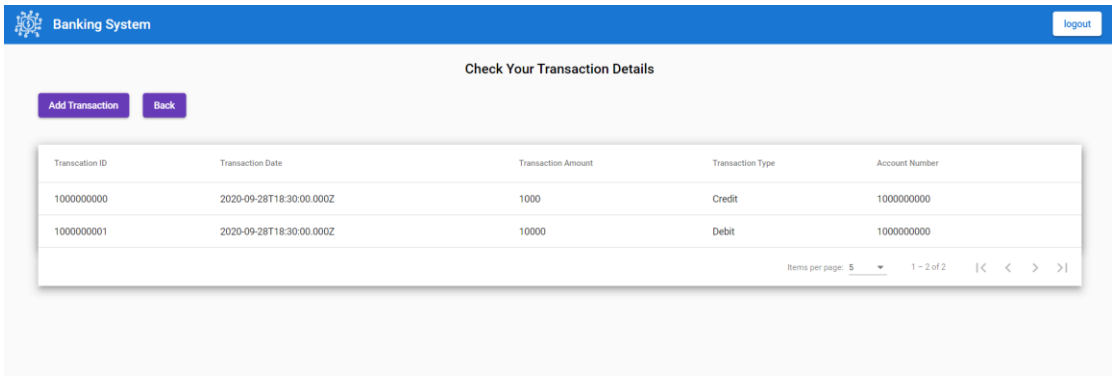
All credit card screen when click on **All Credit Card** button (add, delete and search function on customer Id by employee)

Customer Panel:

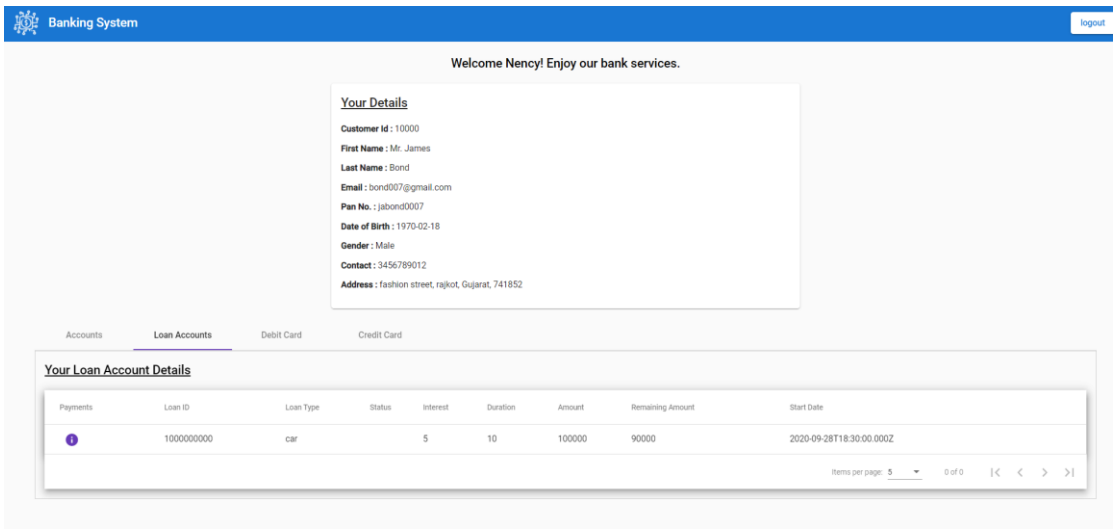
This is customer panel view from which he can see his personal detail as well as all account, loan account, debit card, credit card that is link with his customer id or account id. He can also do transaction as well as payment of his loan through system.



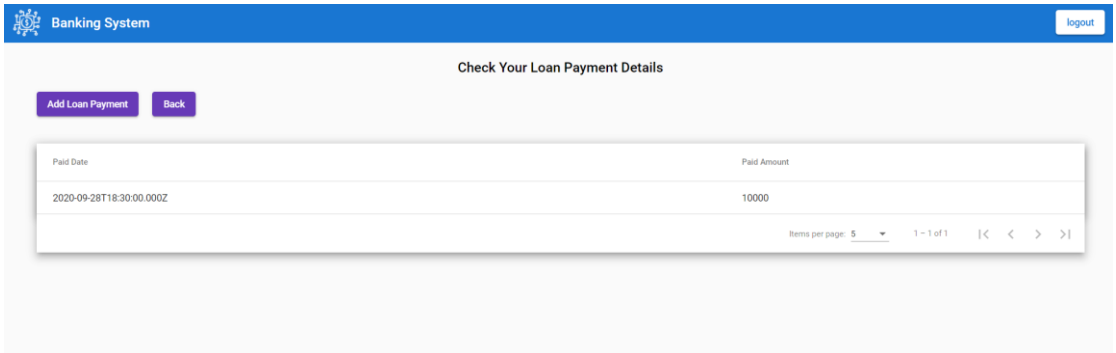
All customer Details and below customer's all accounts, loan account, debit card and credit card listing



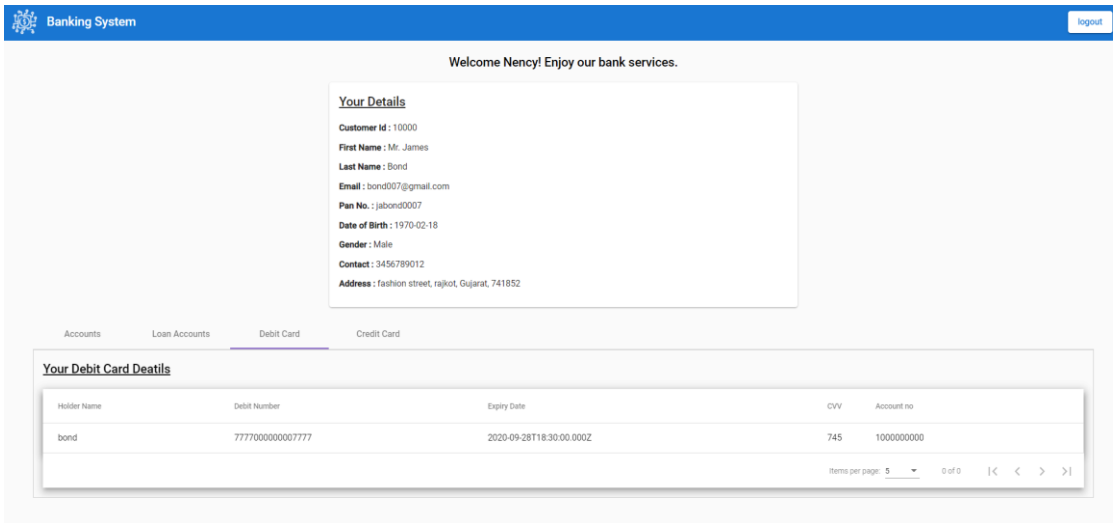
When click on info Icon in accounts listing >> transaction detail page (also add Transaction function)



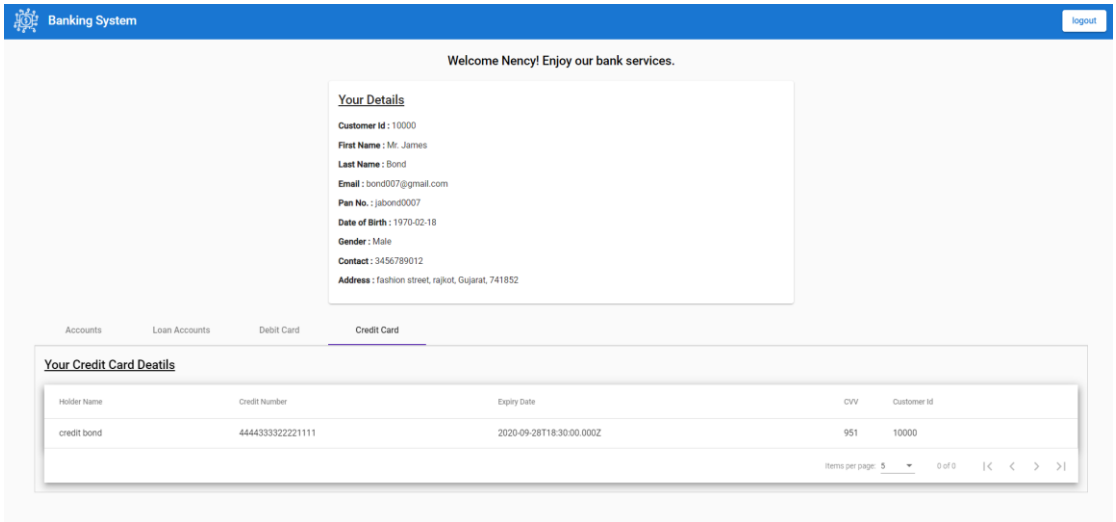
Loan Accounts listing tab is active



When click on info Icon in loan accounts listing >> payment detail page (also add Payment function)



All Debit card listing tab is active



All Credit card listing tab is active

Link source of the application in GitHub

GitHub:

https://github.com/nency-shobhashana/cbd_project

Conclusion

“Banking System" keeps the day by day tally record as a complete banking. Banking system developed user friendly. It can keep the information of Account type, account opening form, Deposit, Withdrawal, and the Transaction report, Individual account opening form, Group Account. The exciting part of this project is that it displays Transaction reports, Statistical.

This project is made to nurture the demands of a customer in a banking state by lodge all the tasks of transaction taking place in a bank. It even reduces the manual work.

References

For ERD :

https://erdplus.com/

For Database:

https://dev.mysql.com/downloads/mysql/

For Angular (development):

https://nodejs.org/en/
https://angular.io/guide/setup-local
https://www.npmjs.com/package/rxjs
https://www.npmjs.com/package/typescript
https://www.npmjs.com/package/cors
https://www.npmjs.com/package/express
https://www.npmjs.com/package/moment
https://www.npmjs.com/package/mysql2