Online Loan Management System

Advanced Database Design CS-603-D

Avalons

Sacred Heart University
School of Computer Science & Engineering
The Jack Welch College of Business & Technology

Submitted To:
Dr. Reza Sadeghi

Late Spring 2022

Project Report of Online Loan Management System

Team Name

Name of the Team	Avalons

Team Members

1. Sambasiva Rao Chennamsetty chennamsettys@mail.sacredheart.edu(Team Head)
2. Jagadishwar Reddy Velma velmaj@mail.sacredheart.edu (TM)
3. Arif Pasha Shaik shaiks11@mail.sacredheart.edu(TM)
4. Teja Sri Ravula ravulat2@mail.sacredheart.edu (TM)
5. Vamsi Kiran Kakkera kakkerav@mail.sacredheart.edu (TM)
6. Siva Rama Krishna Ch chirumamillas2@mail.sacredheart.edu (TM)

Description of Team Members

1. **Sambasiva Rao Chennamsetty**

   I completed my Bachelor's in Information Technology. I had 3+ years of experience as a full-stack developer with Java programming as a backend. I do like to work with a team that has more commitment to work. As long as we all understand the goals and know our priorities, we will work well as a team to complete tasks effectively.
2. **Jagadishwar Reddy Velma**

   I hold 7+ years of experience in SQL Database Administration. I am here to learn and improve better development skills which help me to become an extensive experienced Core Developer.
3. **Arif Pasha Shaik**

   I have completed my Bachelor's in Information Technology, I have done a couple of internships on Visual Basic .net, and I have also done a course on Business Analytics: Data mining and Data warehousing. I have learned about Big data, data analysis, and data management which made me learn more about data. And I love working in a team that has its full dedication towards the work or project.
4. **Teja Sri Ravula**

   I have done my under graduation in computer science and engineering at Sphoorthy Engineering College and started working as a trainee engineer. I worked on Java and PostgreSQL. I do have good knowledge of C, Python, and MySQL. I zeal to learn new trending technologies like artificial intelligence. I would like to work with people who are committed to the work.
5. **Vamsi Kiran Kakkera**

   I have done my Bachelor's degree in the stream of computer science. I'm having work Experience of 2.5 years in the AWS cloud as an Associate Developer. I've chosen this team as they are very coordinative and discuss everything with the team members.

6. **Siva Rama Krishna Chirumamilla**

   I have completed my bachelor's degree in computer science and have 5+ years of work experience as a DevOps engineer and good knowledge of Microsoft technologies. I would describe myself as a cohesive team member and able to do whatever task is necessary to complete the project.

Table of Contents

List of Figures

List of Tables

1 Introduction

Online loan Management System (OLMS) is a project which is taken and being developed by our team which helps people to apply for loans online. Customers need to enter loan applications online. Only Staff or admin has the authority to approve or reject loan applications. Customer can view their Loan account details, Interest rate, repayment schedule details, etc. Customers can make loan payments online as well. After the payment, the system updates with the total paid amount and the balance amount. Administrators can view payment details, loan account details, pending payment details, and do terminate the accounts, etc.

- Admin is the one who verifies the user or the customer who is going to register on the loan management system. There can be only one account of admin and all other accounts can be either of user or customer.
- After verifying the customer's loan request admin can approve the loan and the respected information will be updated in the system with the calculated interest amount. And admin can be able to update customer profiles and add or delete accounts.
- A customer must register him/herself in the application to apply for any type of loan such as a home loan, study loan, car loan, etc. Once customer registration is completed, he can log in with the given credentials in the user module.
- Once the customer has logged in and he/she has made a loan request with the amount, duration, and interest rate. Then Customers loan request goes to the admin module, and if it gets approved, the requested information for that customer will get updated in the system.

2 Business Model

The business model we choose for the project are Happy Money Loan Provider [1], PenFed Credit Union [2], and Light Stream Loans [3]. Which provides loans to the customers online based on their credit card score. With digital transformation assuming a faster pace, loan management software is gaining wider adoption. Faster and more efficient than the legacy lending system, loan software helps automate every stage of the loan lifecycle, from application to closing. We are interested to know how they make the business in the backend and work to grow their organization high and keep being a leading loan provider in the USA.

3 Merits of the project

- This system is designed to easily maintain the data of the loan customers specifically. Customers can apply for loans without visiting the bank.
- Customer can apply for a loan account online. Customer needs to fill their requirements in the loan application.
- This system is made to keep the records of the customers who have taken a loan from a bank.
- This system allows customers to make payments online.
- The admin is the main user of this web application and he can add employee details, Loan types, penalty charges, etc.

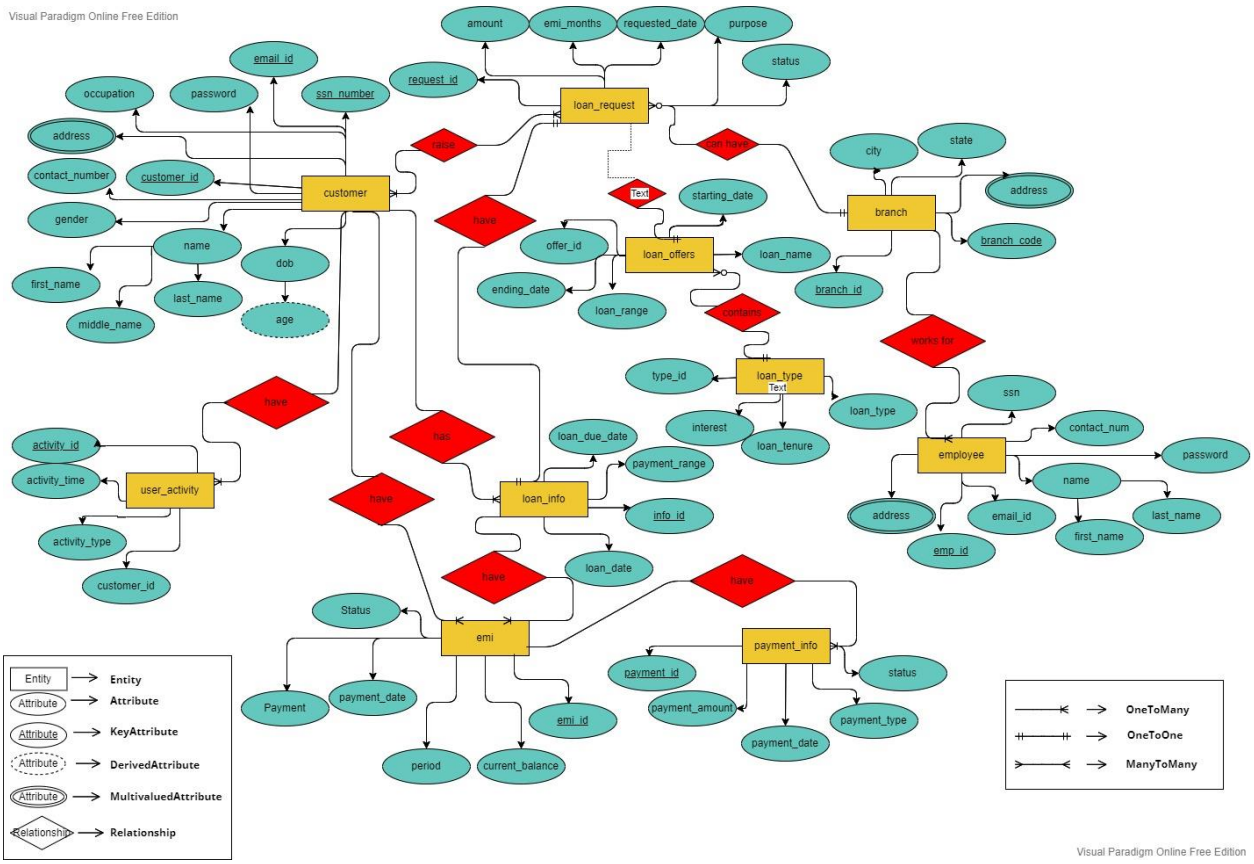
4 Modules of Online Loan Management System

1. **Customer Account module**: This module stores customer account details with login credentials. After the login, the customer can apply for a loan. The customer can update his profile in the profile module and he can change the password in the change password module.
2. **Loan Application Module**: In the loan application module customer can apply for loans by entering loan requirement details. The loan amount will be sanctioned after the admin approves.
3. **Loan Account Module**: This module shows various loan accounts to the customer. Customers can view loan account details with the total loan amount, paid amount, Balance amount, installment details, etc.
4. **Loan Payment Module**: This module allows the customer to make payment for his loan.
5. **Admin Dashboard Module**: This module is for administrators and Employees to manage all web application activities. The administrator is having full authority over the application.
6. **Settings module**: Only the administrator can access this module. The administrator has a unique account with many special access permissions over normal users. In this module administrators or employees can manage the details of Loan types, Employees, processing fees, Delay payment charges, etc.
7. **Report Module**: In the report module Employee or admin can view Loan Payment Report, Loan Account Report, Pending Accounts report, and others.

5 Entity Relationship (ER) Model

In the diagram below we have tables Loan Type, Loan Offers, Loan Request, Customer, Branch, Loan Information, Payment Information, Emi, User Activity. Here we have the cardinality of 1-N between customer and loan requests. One customer can request multiple loans and 1-1 cardinality between loan request and loan information. One loan request contains one loan information. Loan request table has columns request_id where it acts as primary of the table and branch_id, loan_offers_id and customer_id columns as a foreign keys. Branch_id is the primary in branch table whereas it acts as a foreign key in this table to give the information like to which branch customer raised the loan request. Customer table has a column customer_id acts as primary key and foreign key in loan_request table.

Figure 1: ER Diagram of Online Loan Management System

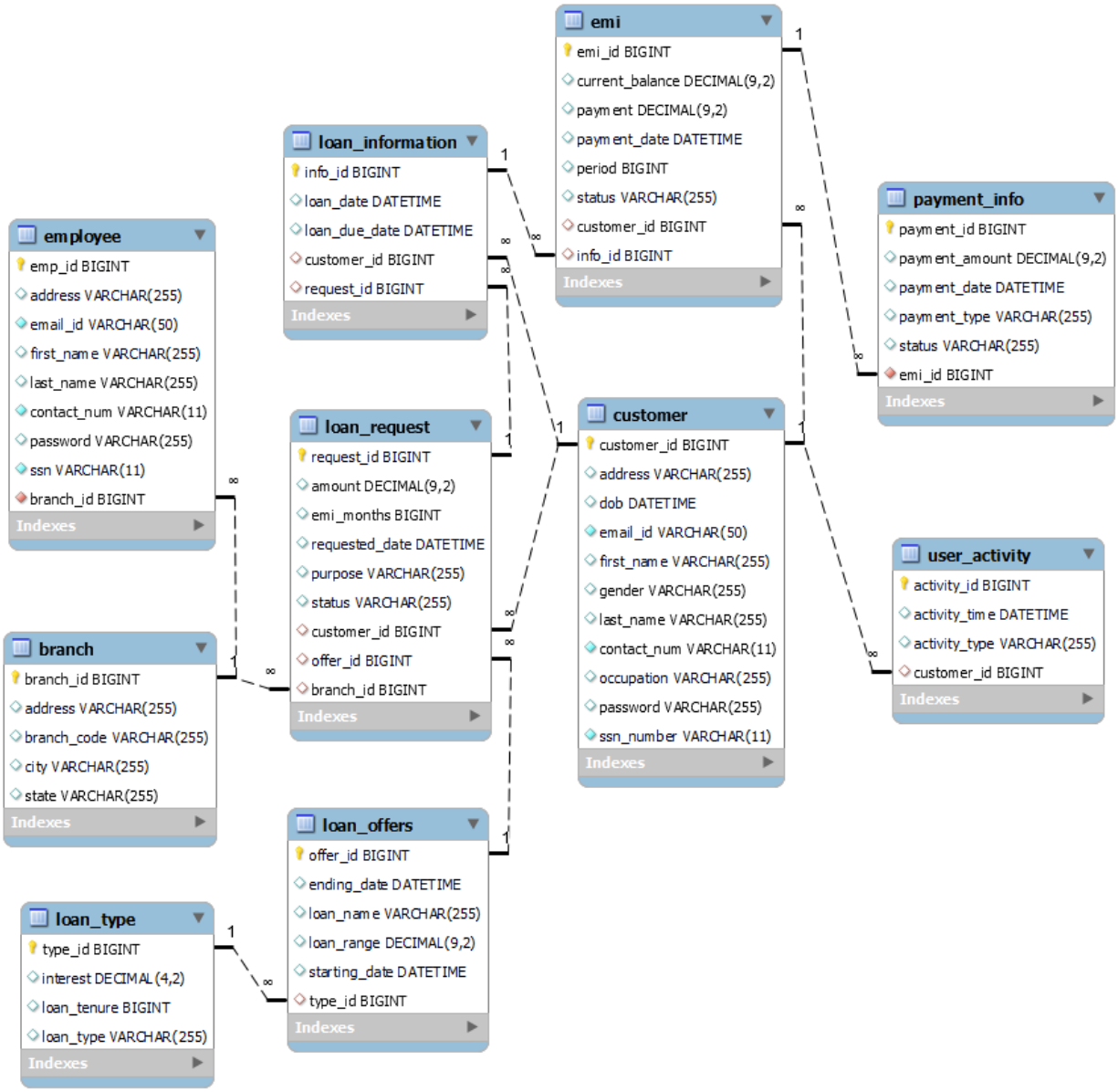


6 Enhanced Entity Relationship (EER) Diagram

The figure below representing Enhanced Entity Relationship of the database we are using for the project of Online loan management system. Figures have multiple tables which are stored in the schema called loan. Each rectangle box denotes as a table of the schema and inside of it describes attributes of the table. The dotted lines describe a table has a relationship with another

table. Each such established relation has a constraint that connects one with another. We have designed this EER diagram for the project by using the MySQL workbench reverse engineering feature.

Figure 2 : EER Diagram of Online Loan Management System



7 Description of the tables

branch: The purpose of the table is used to store branch-related information. In this table, we have taken a total of 5 columns. Admin & user has to select the particular branch that, on which branch admin is going to work on, and on which branch customer is going to apply for the loan. Once select's branch particular branch code(unique) will assign to the

members. This table doesn't have any direct relationship with other tables, but the employee table has a **many-to-one** relationship with this.

Table 1: Branch

Column Name	Datatype	Length	Key type	Cardinality	Description
branch_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
address	varchar	255		-	To store the address of the branch
branch_code	varchar	255		-	Its unique code for each branch
city	varchar	255		-	In which city the branch is located
state	varchar	255		-	In which state the branch is located

employee: Table is used to store employee-related information. The table consists of 9 columns that collect complete information about the employee. The employee has to log in with his credentials which are stored in the table to do their operations from the employee module. The table has a **many-to-one** relationship with the branch table, where multiple employees can work for a branch.

Table 2: Employee

Column Name	Datatype	Length	Key type	Cardinality	Description
emp_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
address	varchar	255		-	To store the address of the employee
email_id	varchar	50		-	Email id of the employee
first_name	varchar	255		-	First name of the employee.

last_name	varchar	255		-	Last name of the employee
contact_num	varchar	11		-	Contact no of the employee.
password	varchar	255		-	The password of the employee.
Ssn	varchar	11		-	Ssn of the employee.
branch_id	bigint		FK	many-to-one	In which branch the employee is working. Branch table pk will store here as a foreign key.

customer: The purpose of the Table is used to store customer information. In this table, we have taken a total of 11 columns. which collects complete information about the customer. The customer must log in with his credentials which are stored in this table. This table has complete customer information. The table has a **one-to-many** relationship with the loan request, user_activity, loan_information, and emi Tables.

Table 3: Customer

Column Name	Datatype	Length	Key type	Cardinality	Description
customer_id	Bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
address	varchar	255		-	To store the address of the Customer
dob	datetime	-		-	Date of birth of the Customer
email_id	varchar	50		-	Email id of the Customer
first_name	varchar	255		-	First name of the Customer
gender	varchar	255		-	Gender of the Customer
last_name	varchar	255		-	Last Name of the Customer.
contact_num	varchar	11		-	Contact No of the Customer.
occupation	varchar	255		-	Occupation details of the Customer.
password	varchar	255		-	The customer password is stored.

ssn_number	varchar	11		-	SSN No of the Customer.

user_activity: The purpose of the Table is used to store user activity. In this table, we have taken a total of 4 columns. User activities are stored in this table like activity time and activity type. customer id which is the primary key in the customer table is taken as a foreign key. This table has **many to one** relationship with the customer table.

Table 4: User Activity

Column Name	Datatype	Length	Key type	Cardinality	Description
activity_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
activity_time	datetime	-		-	To store the activity time.
activity_type	varchar	255		-	To store the activity type.
customer_id	bigint	-	Fk	Many-to-one	In which id of customer is showed. Customer table pk will store here as a foreign key here.

loan_type: The purpose of the table is used to store the type of loan which is applied by the customer. In this table, we have taken total of 4 columns. Each loan type has its unique type_id. This table has **one to many** relationships with loan_offer table.

Table 5: Loan Type

Column Name	Datatype	Length	Key type	Cardinality	Description
type_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
interest	decimal	4,2		-	Interest of the loan is stored.

loan_tenure	bigint	-		-	This stores the time period of loan.
loan_type	varchar	255		-	Stores the type of loan.

loan_offers: The purpose of the Table is used to store loan offers. In this table, we have taken a total of 6 columns. This table stores the loan name and its starting and ending dates and the type of loan. This table has a **many-to-one** relationship with loan_type table where type_id which is the primary key in loan type is taken as a foreign key.

Table 6: Loan Offers

Column Name	Datatype	Length	Key type	Cardinality	Description
offer_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
ending_date	datetime	-		-	to store the ending date of the loan offer.
loan_name	varchar	255		-	Stores the loan name.
loan_range	decimal	9,2		-	Range of the loan is stored.
starting_date	datetime	-			to store the starting date of the loan offer.
type_id	bigint	-	Fk	Many-to-one	Type_id which is the primary key in loan type is taken as foreign key here.

loan_request: The purpose of the Table is used to store loan requests. In this table, we have taken a total of 8 columns. This table stores a unique request_id. The request raised by the customer for a loan and the status of the loan are stored in this table. This table has **many-to-one** relationship with the customer table and **one-to-one** relationship with the loan information and loan offer table.

Table 7: Loan Request

Column Name	Datatype	Length	Key type	Cardinality	Description

request_id	bigint	-	Pk	-	It's the primary key of the table. int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
amount	decimal	9,2		-	Stores the amount requested by the customer.
emi_months	bigint	-		-	No of months to repay the loan.
requested_date	datetime	-		-	Stores the loan requested date.
purpose	varchar	255			The purpose of the loan is stored.
status	varchar	255			Status of loan requested by the customer.
customer_id	bigint	-	Fk	Many-to-one	Customer id which is the primary key in the customer table is used as a foreign key in this table.
offer_id	bigint	-	Fk	One-to-one	Offer id which is the primary key in the loan offer table is used as a foreign key in this table.

loan_information: The purpose of the Table is used to store loan information. In this table, we have taken a total of 5 columns. This table has a customer id which is the primary key in the customer table and is taken as a foreign key in this table. The table has request id which is the primary key in loan offers table is taken as a foreign key in this table. This table has **many-to-one** relationship with the customer table and **one-to-one** relationship with the loan request table.

Table 8: Loan Information

Column Name	Datatype	Length	Key type	Cardinality	Description
info_id	bigint	-	Pk	-	It's the primary key of the table.int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.

loan_date	datetime	-		-	to store the loan date.
loan_due_date	datetime	-		-	Stores the loan due date.
customer_id	bigint	-	**FK**	Many to one	Customer id which is primary key in customer table is used as foreign key in this table.
request_id	bigint	-	**FK**	One to one	request id which is primary key in loan request table is used as foreign key in this table.

payment_info: The purpose of the table is used to store payment related information. In this table, we have taken a total of 6 columns. This table has emi id which is the primary key in the emi table and is taken as a foreign key in this table. This table has **many-to-one** relationship with the emi table.

Table 9: Payment Information

Column Name	Datatype	Length	Key type	Cardinality	Description
payment_id	Bigint	-	Pk	-	It's the primary key of the table.int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
payment_amount	Decimal	9,2		-	To store the payment amount.
payment_date	Datetime	-		-	To store the payment date.
payment_type	Varchar	255		-	To store the payment type.
Status	Varchar	255			The status of the payment is stored.
emi_id	Bigint	-	Fk	Many to one	emi id which is the primary key in emi table is used as a foreign key in this table.

emi: The purpose of the table is used to store emi-related information. In this table, we have taken a total of 8 columns. This table has a customer id which is the primary key in the customer table and is taken as a foreign key in this table. This table has **many-to-one** relationship with the customer table and info id which is the primary key in the loan

information table is taken as a foreign key here. This table has **many-to-one** relationship with the loan information table.

Table 10: EMI

Column Name	Datatype	Length	Key type	Cardinality	Description
emi_id	bigint	-	Pk	-	It's the primary key of the table.int is a 32-bit long while bigint is 64-bit long, therefore it can store much larger numbers like 123456789123456789.
current_balance	decimal	9,2		-	to store the current balance of emi amount.
Payment	decimal	9,2		-	Payment amount of the emi.
payment_date	datetime	-		-	To show payment date.
Period	bigint	-			Month of the emi.
Status	varchar	255			To show the status of the emi.
customer_id	bigint	-	Fk	Many-to-one	In which customer_id is stored. customer table pk will store here as a foreign key here.
info_id	bigint	-	Fk	Many-to-one	In which information id is stored. Loan information pk will store here as a foreign key here.

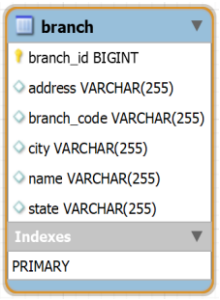
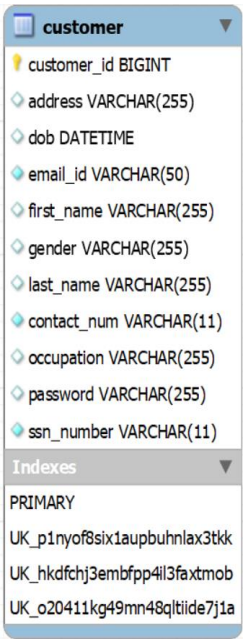
8 Data Definition Language (DDL) of Database

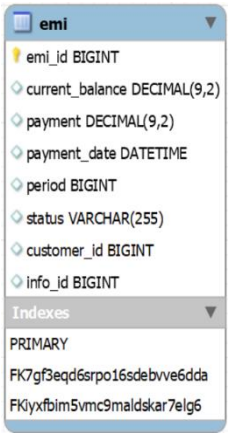
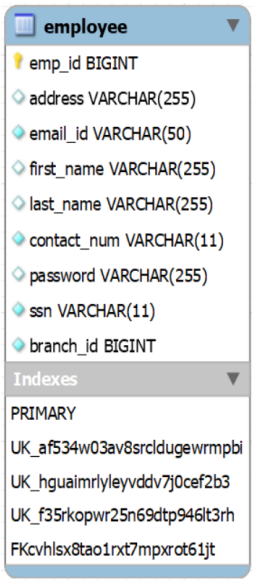
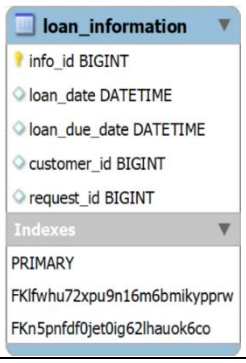
The 'loan' database is designed as the EER diagram. It is created with the help of Structured Query Language (SQL) if the database does not exist. Database contains 10 tables which has relations between them.

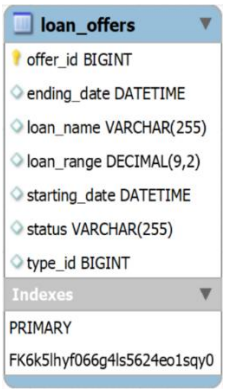
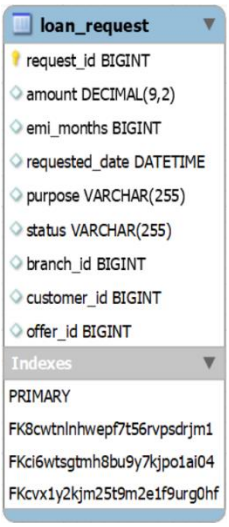
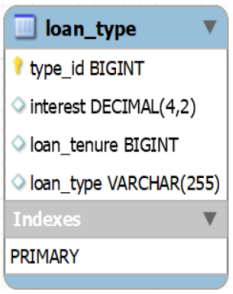
```
-- /* drop database if exists
DROP DATABASE loan;
-- /* create database loan
CREATE DATABASE IF NOT EXISTS loan;
```

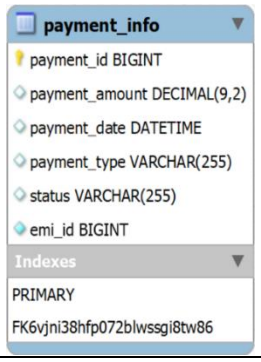
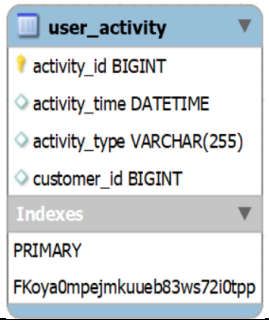
9 Data Definition Language (DDL) of Tables

9.1 Create

Table Name	SQL query	EER Model
branch	CREATE TABLE `branch` (`branch_id` bigint NOT NULL AUTO_INCREMENT, `address` varchar(255) DEFAULT NULL, `branch_code` varchar(255) DEFAULT NULL, `city` varchar(255) DEFAULT NULL, `name` varchar(255) DEFAULT NULL, `state` varchar(255) DEFAULT NULL, PRIMARY KEY (`branch_id`));	
customer	CREATE TABLE `customer` (`customer_id` bigint NOT NULL AUTO_INCREMENT, `address` varchar(255) DEFAULT NULL, `dob` datetime DEFAULT NULL, `email_id` varchar(50) NOT NULL, `first_name` varchar(255) DEFAULT NULL, `gender` varchar(255) DEFAULT NULL, `last_name` varchar(255) DEFAULT NULL, `contact_num` varchar(11) NOT NULL, `occupation` varchar(255) DEFAULT NULL, `password` varchar(255) DEFAULT NULL, `ssn_number` varchar(11) NOT NULL, PRIMARY KEY (`customer_id`), UNIQUE KEY `UK_p1nyof8six1aupbuhnlax3tkk` (`email_id`), UNIQUE KEY `UK_hkdfchj3embfpp4il3faxtmob` (`contact_num`), UNIQUE KEY `UK_o20411kg49mn48qltiide7j1a` (`ssn_number`)) ;	

emi	CREATE TABLE `emi` (`emi_id` bigint NOT NULL AUTO_INCREMENT, `current_balance` decimal(9,2) DEFAULT NULL, `payment` decimal(9,2) DEFAULT NULL, `payment_date` datetime DEFAULT NULL, `period` bigint DEFAULT NULL, `status` varchar(255) DEFAULT NULL, `customer_id` bigint DEFAULT NULL, `info_id` bigint DEFAULT NULL, PRIMARY KEY (`emi_id`));	
employee	CREATE TABLE `employee` (`emp_id` bigint NOT NULL AUTO_INCREMENT, `address` varchar(255) DEFAULT NULL, `email_id` varchar(50) NOT NULL, `first_name` varchar(255) DEFAULT NULL, `last_name` varchar(255) DEFAULT NULL, `contact_num` varchar(11) NOT NULL, `password` varchar(255) DEFAULT NULL, `ssn` varchar(11) NOT NULL, `branch_id` bigint NOT NULL, PRIMARY KEY (`emp_id`), UNIQUE KEY `UK_af534w03av8srcldugewrmpbi` (`email_id`), UNIQUE KEY `UK_hguaimrlyleyvddv7j0cef2b3` (`contact_num`), UNIQUE KEY `UK_f35rkopwr25n69dtp946lt3rh` (`ssn`));	
loan_information	CREATE TABLE `loan_information` (`info_id` bigint NOT NULL AUTO_INCREMENT, `loan_date` datetime DEFAULT NULL, `loan_due_date` datetime DEFAULT NULL, `customer_id` bigint DEFAULT NULL, `request_id` bigint DEFAULT NULL, PRIMARY KEY (`info_id`));	

loan_offers	CREATE TABLE `loan_offers` (`offer_id` bigint NOT NULL AUTO_INCREMENT, `ending_date` datetime DEFAULT NULL, `loan_name` varchar(255) DEFAULT NULL, `loan_range` decimal(9,2) DEFAULT NULL, `starting_date` datetime DEFAULT NULL, `status` varchar(255) DEFAULT NULL, `type_id` bigint DEFAULT NULL, PRIMARY KEY (`offer_id`),);	
loan_requests	CREATE TABLE `loan_request` (`request_id` bigint NOT NULL AUTO_INCREMENT, `amount` decimal(9,2) DEFAULT NULL, `emi_months` bigint DEFAULT NULL, `requested_date` datetime DEFAULT NULL, `purpose` varchar(255) DEFAULT NULL, `status` varchar(255) DEFAULT NULL, `branch_id` bigint DEFAULT NULL, `customer_id` bigint DEFAULT NULL, `offer_id` bigint DEFAULT NULL, PRIMARY KEY (`request_id`));	
loan_type	CREATE TABLE `loan_type` (`type_id` bigint NOT NULL AUTO_INCREMENT, `interest` decimal(4,2) DEFAULT NULL, `loan_tenure` bigint DEFAULT NULL, `loan_type` varchar(255) DEFAULT NULL, PRIMARY KEY (`type_id`));	

payment_info	CREATE TABLE `payment_info`(`payment_id` bigint NOT NULL AUTO_INCREMENT , `payment_amount` decimal(9,2) DEFAULT NULL, `payment_date` datetime DEFAULT NULL, `payment_type` varchar(255) DEFAULT NULL, `status` varchar(255) DEFAULT NULL, `emi_id` bigint NOT NULL, PRIMARY KEY (`payment_id`));	
user_activity	CREATE TABLE `user_activity` (`activity_id` bigint NOT NULL AUTO_INCREMENT, `activity_time` datetime DEFAULT NULL, `activity_type` varchar(255) DEFAULT NULL, `customer_id` bigint DEFAULT NULL, PRIMARY KEY (`activity_id`));	

9.2 Adding foreign Key

- Adding foreign keys to emi tables from customer table and loan information table.

```
ALTER TABLE `loan`.`emi`
ADD INDEX `customer_id_idx` (`customer_id` ASC, `info_id` ASC) visible;
ALTER TABLE `loan`.`emi`
ADD CONSTRAINT `customer_id` FOREIGN KEY(`customer_id` , `info_id`)
REFERENCES `loan`.`customer` (`customer_id` , `customer_id`),
ADD CONSTRAINT `info_id` FOREIGN KEY (`customer_id`)
REFERENCES `loan`.`loan_information` (`info_id`)
```

- Adding foreign key to employee table from branch table.

```
ALTER TABLE `loan`.`employee`
ADD CONSTRAINT `branch_id` FOREIGN KEY (`branch_id`)
REFERENCES `loan`.`branch` (`branch_id`);
```

- Adding foreign keys to loan information table from customer table and loan request table.

```
ALTER TABLE `loan`.`loan_information`
ADD INDEX `request_id_idx` (`request_id` ASC, `customer_id` ASC) visible;
ALTER TABLE `loan`.`loan_information`
ADD CONSTRAINT `customer_id` FOREIGN KEY ()
```

```
REFERENCES `loan`.`customer` ()
ADD CONSTRAINT `request_id` FOREIGN KEY (`request_id` , `customer_id`)
REFERENCES `loan`.`loan_request` (`request_id` , `customer_id`)
```

- Adding foreign key to loan offers table from loan_type table and loan request table.

```
ALTER TABLE `loan`.`loan_offers`
ADD CONSTRAINT `type_id` FOREIGN KEY (`type_id`)
REFERENCES `loan`.`loan_type` (`type_id`)
```

- Adding foreign key to loan request table from branch table, customer and loan offer table.

```
ALTER TABLE `loan`.`loan_request`
ADD CONSTRAINT `branch_id` FOREIGN KEY (`branch_id`)
REFERENCES `loan`.`branch` (`branch_id`),
ADD CONSTRAINT`customer_id` FOREIGN KEY (`customer_id`)
REFERENCES `loan`.`customer` (`customer_id`),
ADD CONSTRAINT `offer_id` FOREIGN KEY(`branch_id`)
REFERENCES`loan`.`loan_offers` (`offer_id`);
```

- Adding foreign key to payment info table from emi table.

```
ALTER TABLE `loan`.`payment_info`
ADD CONSTRAINT `emi_id` FOREIGN KEY (`emi_id`)
REFERENCES `loan`.`emi` ( `emi_id`);
```

- Adding foreign key to user_activity table from customer table.

```
ALTER TABLE `loan`.`user_activity`
ADD CONSTRAINT `customer_id` FOREIGN KEY (`customer_id`)
REFERENCES `loan`.`customer` (`customer_id`)
```

9.3 Add Column

The below query is to add new column middle_name to employee table of varchar(255) data type after last_name attribute.

Figure 3: Add Column

```
ALTER TABLE `loan`.`employee`
ADD COLUMN `middle_name` VARCHAR(255) NULL AFTER `last_name`;
```

Table: employee

Columns:

emp_id	bigint AI PK
address	varchar(255)
email_id	varchar(50)
first_name	varchar(255)
last_name	varchar(255)
contact_num	varchar(11)
password	varchar(255)
ssn	varchar(11)
branch_id	bigint

Table: employee

Columns:

emp_id	bigint AI PK
address	varchar(255)
email_id	varchar(50)
first_name	varchar(255)
last_name	varchar(255)
middle_name	varchar(255)
contact_num	varchar(11)
password	varchar(255)
ssn	varchar(11)
branch_id	bigint

9.4 Drop Column

The below query is used to delete the attribute middle_name from the employee table.

Figure 4: Drop Column

```
ALTER TABLE `loan`.`employee` DROP COLUMN `middle_name`;
```

9.5 Change Column Type

The below query is used to change the column type of middle_name attribute from varchar(255) to char(30).

Figure 5: Change Column Type

```
ALTER TABLE `loan`.`employee`
CHANGE COLUMN `middle_name` `middle_name` CHAR(30) NULL DEFAULT NULL ;
```

9.6 Rename Column Name

The below query is used to rename the column name of middle_name to m_name in employee table.

Figure 6: Rename the column name

```
ALTER TABLE `loan`.`employee`
CHANGE COLUMN `middle_name` `m_name` CHAR(30) NULL DEFAULT NULL ;
```

9.7 Change Order of Columns

The below query is used to change the order of employee table.

Figure 7: Changing Order of columns

```
ALTER TABLE `loan`.`employee`
CHANGE COLUMN `email_id` `email_id` VARCHAR(50) NOT NULL AFTER `contact_num`,
CHANGE COLUMN `emp_id` `emp_id` BIGINT NOT NULL AUTO_INCREMENT AFTER `branch_id`,
CHANGE COLUMN `first_name` `first_name` VARCHAR(255) NULL DEFAULT NULL AFTER `emp_id`,
CHANGE COLUMN `password` `password` VARCHAR(255) NULL DEFAULT NULL AFTER `first_name`,
CHANGE COLUMN `address` `address` VARCHAR(255) NULL DEFAULT NULL AFTER `password`;
```

10 Data Manipulation Language(DML)

10.1 Insert:

10.1.1 Branch:

The below query is to insert records into branch table.

```
INSERT INTO `branch` VALUES
('1','396 Gregory Street','BRDPRT','Bridgeport','Bank of Lo
an','Connecticut'),
('2','225 Taft Avenue','NWHVN','New Haven','Bank of Loan','
Connecticut'),
('3','678 Jhon Street','PRKAVN','New York','Bank of Loan','
New York' ),('4','345 Main Street','MILFORD','Milford','Ban
k of Loan','Connecticut'),
('5','634 Abraham Street','SRTFRD','Stratford','Bank of Loa
n','Connecticut'),
('6','342 Link Street','WSTPRT','Westport','Bank of Loan','
Connecticut'),
('7','798 East Street','NRWLK','Norwalk','Bank of Loan','Ne
w Jersy'),
('8','234 South Avenue','FRFLD','Fairfield','Bank of Loan',
'Texas'),
('9','789 Elm Street','WSTHVN','West Haven','Bank of Loan',
'Florida'),
('10','346 Lambart Street','ORNGE','Orange','Bank of Loan',
'Connecticut');
```

10.1.2 Customer:

The query used below is to insert records into customer table.

```
INSERT INTO `customer` VALUES
('1','225 Taft Avenue','1998-05-25
07:28:00’,‘arifshaik@gmail.com','Arif','M', 'Shaik','9877896544',
'Student','arif@123','9876543210'),
('2','245 Fair Avenue','1995-03-26    05:34:33',   'jagadeshjay@gmail.com', 'Jagadishwar',
'M', 'Velma', '2398447646','Student','jagadessh@123', '4239784734'),
('3','396 Gregory Street','1997-0428 08:01:12',   'samba.ch97@gmail.com','Samba',   'M',
'Chennamsetty', '8008051986','Student','samba@123','1234567890') ,
('4','19 Park Town',  '1995-07  10 02:04:43',  'sivakrishna@gmail.com',   'Siva',  'M',  'CH' ,
'5834205343', 'Student','siva@123','4534987384'),
('5','189 Old Tavern Street','1997-08-23 16:09:33', 'tejasri@gmail.com',      'Teja',      'F' ,
'Ravula',                                                                    '2983749243',
'Student','tejasri@123','3482309844'),
('6','234 Burnum Avenue',   '1998-11-12 19:55:55',   'vamsik@gmail.com'   ,'Vamsi',   'M'
, 'Kakkera' , '4023984033','Student','vamsik@123','2430982034'),
```

```
('7','424 Woodruff Road','1996-03-
29 03:23:34','harich@gmail.com', 'Hari','M', 'Chennamsetty',
'3495803458','Employee','hari@123','2340086749'),
('8','34 Roses Mill Road',  '1994-07  12 04:23:45'  ,'saich@gmail.com'  ,  'Sai','M'
, 'Chennamsetty', '5385039485','Employee','sai@123','3804557859'),
('9','67 Oxford Road','1999-04-12 21:53:53',  'harsha@gmail.com',  'Harsha','M',
'Chennamsetty', '3068480439', 'Employee', 'harsha@123', '5349435794'),
('10','74 Peck Ln Road', '2000-02-19 12:43:59', 'dattu@gmail.com', 'Dattu','M', 'Thota',
'5084300808','Student','dattu@123','4038638094');
```

10.1.3 **Employee:**

The below query is used to insert records into the employee table.

```
INSERT INTO `employee` VALUES
('1','225 Taft Avenue','jamesm@gmail.com','James','Mary', '
4820480324','james@123' ,'4376526345','1'),
('2','245 Fair Avenue','robertpa@gmail.com','Robert','Patri
cia', '5423736554', 'robert@123','1348012343','2'),
('3','396 Gregory Street','michaeljen@gmail.com','Michael',
'Jennifer', '6546345645','michael@123','4567456737','3'),
('4','19 Park Town','davidlin@gmail.com','David','Linda', '
7654725445','david@123' ,'3752342545','4'),
('5','189 Old Tavern Street','willianeli@gmail.com','Willia
m','Elizabeth', '3678854654','william@123','8535436568','5'
),
('6','234 Burnum Avenue','richardbar@gmail.com','Richard','
Barbara', '3567467465','richard@123','9687436245','6'),
('7','424 Woodruff Road','josephsus@gmail.com','Joseph','Su
san', '8462345634', 'joseph@123','9676346643','6'),
('8','34 Roses Mill Road','thomasjes@gmail.com','Thomas','J
essica', '3778456435','thomas@123','9876543546','7'),
('9','67 Oxford Road','charlessa@gmail.com','Charles','Sara
h', '4363847624', 'charles@123','4274678245','8'),
('10','74 Peck Ln Road','johnka@gmail.com','John','Karen',
'2367345253','jhon@123' ,'3485877935','8');
```

10.1.4 **Loan Type:**

The query below used is to insert records into loan_type table.

```
INSERT INTO `loan_type` VALUES
('1','2.70','10','Education Loan'), ('2','1.2','5','Car Loa
n'),
('3','3.2','4','Home Loan'), ('4','5.2','3','Gold Loan'),
('5','3.8','6','Education Loan'), ('6','2.9','4','Car Loan'
),
('7','2.2','15','Home Loan'),('8','1.9','12','Gold Loan'),
```

```
('9','2.7','10','Education Loan'),('10','6.2','8','Car Loan
');
```

10.1.5 **Loan Offers:**

The query below used is to insert records into loan_offers table.

```
INSERT INTO `loan_offers` VALUES
('1','2022-06-28 12:13:12', 'Vidya Loan', '200000.00', '2022-05-25 08:03:12', 'A', '2'),
('2','2022-07-12 08:03:12','MSME Loan','120000.00', '2022-05-26 08:03:12','A','3'),
('3','2022-08-23 23:03:12','PMMY Loan','18000','2022-06-12 08:03:12' ,'A','4'),
('4','2022-09-12 11:03:12','CGFMSE Loan','90000', '2022-05-30 08:03:12','A','6')
('5','2022-10-21 09:03:12','NSIC Loan','23000','2022-06-03 08:03:12' ,'A','1'),
('6','2022-11-12 05:03:12','CLCSS Loan','13000', '2022-06-30 08:03:12','A','9'),
('7','2022-12-23 12:03:12','FDF Loan','7800','2022-06-21 08:03:12','A','7'),
('8','2023-10 23 02:03:12','MMR Loan','6700','2022-06-21 08:03:12','A','3'),
('9','2024-01-13 11:03:12','RTL Loan','4200','2022-06-17 08:03:12','A','10'),
('10','2023-12-27 12:04:56','DLM Loan','6700','2022-08-21 08:03:12','A','1');
```

10.1.6 **Loan Request:**

The query below used is to insert records into loan_request table.

```
INSERT INTO `loan_request` VALUES
('1','3000.00','24','2022-05-
25 08:03:16','To Study','I','1','1','1'),
('2','4500','12','2022-05-
22 16:23:32','Buy Car','A','2','5','7'),
('3','5600','36','2022-05-
22 18:03:53','Home Construction','A','5','7', '9'),
('4','5500','24','2022-05-18 02:03:45','Education','I','9',
'3','6'),
('5','7000','24','2022-05-
20 10:13:17','Home Construction','I','5','9', '3'),
('6','5300','48','2022-04-
30 10:13:17','Buy Car','A','7','5','3'),
('7','6900','12','2022-05-
19 18:56:56','Home Construction','I','5','3', '9'),
('8','5000','60','2022-05-19  21:45:43',  'To buy Vehicle',
'A','2','5', '9'),
('9','9800','24','2022-05-
21 11:59:31','To Study','I','5','9','1'),
('10','2300','6',                                  '2022-05-
25 20:28:45','Car Loan','A','6','8','4');
```

10.1.7 Loan Information:

The query below used is to insert records into loan_info table.

```
INSERT INTO `loan_information` VALUES
('1','2022-05-28 08:03:12','2024-05-28 08:03:12','1','1'),
('2','2022-05-22 18:23:22','2024-05-28 18:23:22','2','8'),
('3','2022-04-18 21:12:19','2023-04-28 21:12:19','6','3'),
('4','2022-05-13 16:26:59','2023-05-13 16:26:59','1','9'),
('5','2022-05-18 16:26:59','2023-05-18 16:26:59','6','1'),
('6','2022-05-21 23:45:42','2023-05-21 23:45:42','4','3'),
('7','2022-04-29 21:53:59','2023-04-29 21:53:59','6','5'),
('8','2022-05-07 15:32:32','2023-05-07 15:32:32','8','7'),
('9','2022-04-26 22:56:16','2023-04-26 22:56:16','3','3'),
('10','2022-05-13 16:26:59','2023-05-13 16:26:59','9','8');
```

10.1.8 Emi:

The query below used is to insert records into emi table.

```
INSERT INTO `emi` VALUES
('1','3000.00','300.00','2022-06-
28 08:03:12','1','C','1','3'),
('2','2700.00','300.00','2022-07-
28 08:03:12','2','C','1','3'),
('3','2400.00','300.00','2022-08-
28 08:03:12','3','C','1','3'),
('4','2100.00','300.00','2022-09
28 08:03:12','4','C','1','3'),
('5','1800.00','300.00','2022-10-
28 08:03:12','5','C','1','3'),
('6','1500.00','300.00','2022-11-
28 08:03:12','6','C','1','3'),
('7','1200.00','300.00','2022-12-
28 08:03:12','7','C','1','3'),
('8','900.00','300.00','2023-01-
28 08:03:12','8','C','1','3'),
('9','600.00','300.00','2023-02-
28 08:03:12','9','C','1','3'),
('10','300.00','300.00','2023-03-
28 08:03:12','10','U','1','3');
```

10.1.9 Payment Info:

The query below used is to insert records into payment_info table.

```
INSERT INTO `payment_info` VALUES
('1','300.00','2022-05-
25 08:19:28','Online Banking','P','1'),
```

```
('2','300.00','2022-05-
25 08:21:33','Online Banking','C','1'),
('3','300.00','2022-05-24 18:29:56','Credit','C','3'),
('4','300.00','2022-05-15 08:19:28','Debit','P','4'),
('5','300.00','2022-05-15 09:57:34','Debit','C','4'),
('6','300.00','2022-05-24 03:43:55','Credit','C','5'),
('7','300.00','2022-05-
15 13:54:29','Online Banking','C','7'),
('8','300.00','2022-05-04 04:23:56','Zelle','P','2'),
```

10.1.10 User Activity:

The query below used is to insert records into user_activity table.

```
INSERT INTO `user_activity` VALUES
('1','2022-05-28 08:23:12','Login','1'),
('2','2022-05 28 20:52:22','Logout','1'),
('3','2022-05-28 20:53:15','Login','7'),
('4','2022-05-28 20:54:22','Pwd Changed','7'),
('5','2022-05-28 21:04:08','Loan Applied','7'),
('6','2022-05 28 21:14:57','Logout','7'),
('7','2022-05-28 21:24:33','Login','9'),
```

10.2 Update

Query below is used to update employee ssn where employee id is equal to 1.

Figure 8: Update data

```
UPDATE `loan`.`employee` SET `ssn` = '4376526347' WHERE (`emp_id` = '1');
```

Figure 9: Update using where clause (Before)

```
UPDATE loan.employee set password ="pwd@123" where last_name like "s%";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

emp_id	address	email_id	first_name	last_name	contact_num	password	ssn	branch_id
1	225 Taft Avenue	jamesm@gmail.com	James	Mary	4820480324	james@123	4376526347	1
2	245 Fair Avenue	robertpa@gmail.com	Robert	Patricia	5423736554	robert@123	1348012343	2
3	396 Gregory Street	michaeljen@gmail.com	Michael	Jennifer	6546345645	michael@123	4567456737	3
4	19 Park Town	davidlin@gmail.com	David	Linda	7654725445	david@123	3752342545	4
5	189 Old Tavern Street	willianeli@gmail.com	William	Elizabeth	3678854654	william@123	8535436568	5
6	234 Burnum Avenue	richardbar@gmail.com	Richard	Barbara	3567467465	richard@123	9687436245	6
7	424 Woodruff Road	josephsus@gmail.com	Joseph	Susan	8462345634	joseph@123	9676346643	6
8	34 Roses Mill Road	thomasjes@gmail.com	Thomas	Jessica	3778456435	thomas@123	9876543546	7
9	67 Oxford Road	charlessa@gmail.com	Charles	Sarah	4363847624	charles@123	4274678245	8
10	74 Peck Ln Road	johnka@gmail.com	John	Karen	2367345253	jhon@123	3485877935	8
11	74 Peck Ln Road	johnka1@gmail.com	John	Karen	2367345323	jhon@123	3485837935	99

Figure 10: After Update

```
5 •  UPDATE loan.employee set password ="pwd@123" where last_name like "s%";
```

sult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

emp_id	address	email_id	first_name	last_name	contact_num	password	ssn	branch_id
1	225 Taft Avenue	jamesm@gmail.com	James	Mary	4820480324	james@123	4376526345	1
2	245 Fair Avenue	robertpa@gmail.com	Robert	Patricia	5423736554	robert@123	1348012343	2
3	396 Gregory Street	michaeljen@gmail.com	Michael	Jennifer	6546345645	michael@123	4567456737	3
4	19 Park Town	davidlin@gmail.com	David	Linda	7654725445	david@123	3752342545	4
5	189 Old Tavern Street	willianeli@gmail.com	William	Elizabeth	3678854654	william@123	8535436568	5
6	234 Burnum Avenue	richardbar@gmail.com	Richard	Barbara	3567467465	richard@123	9687436245	6
7	424 Woodruff Road	josephsus@gmail.com	Joseph	Susan	8462345634	pwd@123	9676346643	6
8	34 Roses Mill Road	thomasjes@gmail.com	Thomas	Jessica	3778456435	thomas@123	9876543546	7
9	67 Oxford Road	charlessa@gmail.com	Charles	Sarah	4363847624	pwd@123	4274678245	8
10	74 Peck Ln Road	johnka@gmail.com	John	Karen	2367345253	jhon@123	3485877935	8
11	74 Peck Ln Road	johnka1@gmail.com	John	Karen	2367345323	jhon@123	3485837935	99

10.3 Delete

The below query is used to delete the employee from employee table based on employee id.

Figure 11: Deleting employee

```
1    DELETE FROM `loan`.`employee` WHERE (`emp_id` = '3');
2
```

11 Constraints

11.1 Primary Key Constrain:

The table employee is trying to insert the new record with already existing primary key of '10' of emp_id attribute, so it is throwing primary key constraint violation exception.

Figure 12: Primary Key Constrain

Message Log

```
Operation failed: There was an error while applying the SQL script to the database.
Executing:
INSERT INTO `loan`.`employee` (`emp_id`, `address`, `email_id`, `first_name`, `last_name`,
`contact_num`, `password`, `ssn`, `branch_id`) VALUES ('10', '74 Peck Ln Road', 'johnka@gmail.com',
'John', 'Karen', '2367345253', 'jhon@123', '3485877935', '12');

ERROR 1062: 1062: Duplicate entry '10' for key 'employee.PRIMARY'
SQL Statement:
INSERT INTO `loan`.`employee` (`emp_id`, `address`, `email_id`, `first_name`, `last_name`,
`contact_num`, `password`, `ssn`, `branch_id`) VALUES ('10', '74 Peck Ln Road', 'johnka@gmail.com',
'John', 'Karen', '2367345253', 'jhon@123', '3485877935', '12')
```

11.2 Unique Key Constrain:

The table employee is trying to insert new record with the already existing email johnka@gmail.com of email_id attribute, so it is throwing unique key constraint violation exception.

Figure 13: Unique Key Constrain

Message Log

```
Operation failed: There was an error while applying the SQL script to the database.
Executing:
INSERT INTO `loan`.`employee` (`emp_id`, `address`, `email_id`, `first_name`, `last_name`,
`contact_num`, `password`, `ssn`, `branch_id`) VALUES ('11', '74 Peck Ln Road', 'johnka@gmail.com',
'John', 'Karen', '2367345253', 'jhon@123', '3485877935', '12');

ERROR 1062: 1062: Duplicate entry 'johnka@gmail.com' for key
'employee.UK_af534w03av8srcldugewrmpbi'
SQL Statement:
INSERT INTO `loan`.`employee` (`emp_id`, `address`, `email_id`, `first_name`, `last_name`,
`contact_num`, `password`, `ssn`, `branch_id`) VALUES ('11', '74 Peck Ln Road', 'johnka@gmail.com',
'John', 'Karen', '2367345253', 'jhon@123', '3485877935', '12')
```

12 Data Query Language(DQL)

12.1 Select

Branch

The below query is used to fetch the all records in branch table.

Figure 14: Selecting branch data

```
4 •   SELECT * FROM loan.branch;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

branch_id	address	branch_code	city	name	state
1	396 Gregory Street	BRDPRT	Bridgeport	Bank of Loan	Connecticut
2	225 Taft Avenue	NWHVN	New Haven	Bank of Loan	Connecticut
3	678 Jhon Street	PRKAVN	New York	Bank of Loan	New York
4	345 Main Street	MILFRD	Milford	Bank of Loan	Connecticut
5	634 Abraham Street	SRTFRD	Stratford	Bank of Loan	Connecticut
6	342 Link Street	WSTPRT	Westport	Bank of Loan	Connecticut
7	798 East Street	NRWLK	Norwalk	Bank of Loan	New Jersy
8	234 South Avenue	FRFLD	Fairfield	Bank of Loan	Texas
9	789 Elm Street	WSTHVN	West Haven	Bank of Loan	Florida
10	346 Lambart Street	ORNGE	Orange	Bank of Loan	Connecticut

Customer

The below query is used to fetch the all records of customer table.

Figure 15: Selecting customer data

```
5 • SELECT * FROM loan.customer;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	address	dob	email_id	first_name	gender	last_name	contact_num	occupation	password	ssn_number
1	225 Taft Avenue	1998-05-25 07:28:00	arifshaik@gmail.com	Arif	M	Shaik	9877896544	Student	arif@123	9876543210
2	245 Fair Avenue	1995-03-26 05:34:33	jagadeshjay@gmail.com	Jagadishwar	M	Velma	2398447646	Student	jagadessh@123	4239784734
3	396 Gregory Street	1997-04-28 08:01:12	samba.ch97@gmail.com	Samba	M	Chennamsetty	8008051986	Student	samba@123	1234567890
4	19 Park Town	1995-07-10 02:04:43	sivakrishna@gmail.com	Siva	M	CH	5834205343	Student	siva@123	4534987384
5	189 Old Tavern Street	1997-08-23 16:09:33	tejasri@gmail.com	Teja	F	Ravula	2983749243	Student	tejasri@123	3482309844
6	234 Burnum Avenue	1998-11-12 19:55:55	vamsik@gmail.com	Vamsi	M	Kakkera	4023984033	Student	vamsik@123	2430982034
7	424 Woodruff Road	1996-03-29 03:23:34	harich@gmail.com	Hari	M	Chennamsetty	3495803458	Employee	hari@123	2340086749
8	34 Roses Mill Road	1994-07-12 04:23:45	saich@gmail.com	Sai	M	Chennamsetty	5385039485	Employee	sai@123	3804557859
9	67 Oxford Road	1999-04-12 21:53:53	harsha@gmail.com	Harsha	M	Chennamsetty	3068480439	Employee	harsha@123	5349435794
10	74 Peck Ln Road	2000-02-19 12:43:59	dattu@gmail.com	Dattu	M	Thota	5084300808	Student	dattu@123	4038638094

Employee

The below query is used to fetch all the records of employee table.

Figure 16: Selecting employee data

```
5 • SELECT * FROM loan.employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

emp_id	address	email_id	first_name	last_name	contact_num	password	ssn	branch_id
1	225 Taft Avenue	jamesm@gmail.com	James	Mary	4820480324	james@123	4376526345	1
2	245 Fair Avenue	robertpa@gmail.com	Robert	Patricia	5423736554	robert@123	1348012343	2
3	396 Gregory Street	michaeljen@gmail.com	Michael	Jennifer	6546345645	michael@123	4567456737	3
4	19 Park Town	davidlin@gmail.com	David	Linda	7654725445	david@123	3752342545	4
5	189 Old Tavern Street	willianeli@gmail.com	William	Elizabeth	3678854654	william@123	8535436568	5
6	234 Burnum Avenue	richardbar@gmail.com	Richard	Barbara	3567467465	richard@123	9687436245	6
7	424 Woodruff Road	josephsus@gmail.com	Joseph	Susan	8462345634	joseph@123	9676346643	6
8	34 Roses Mill Road	thomasjes@gmail.com	Thomas	Jessica	3778456435	thomas@123	9876543546	7
9	67 Oxford Road	charlessa@gmail.com	Charles	Sarah	4363847624	charles@123	4274678245	8
10	74 Peck Ln Road	johnka@gmail.com	John	Karen	2367345253	jhon@123	3485877935	8

Loan Type

The below query is used to fetch all the records of loan type table.

Figure 17: Selecting loan type data

```
5 • SELECT * FROM loan.loan_type;
```

Result Grid | Filter Rows:

type_id	interest	loan_tenure	loan_type
1	2.70	10	Education Loan
2	1.20	5	Car Loan
3	3.20	4	Home Loan
4	5.20	3	Gold Loan
5	3.80	6	Education Loan
6	2.90	4	Car Loan
7	2.20	15	Home Loan
8	1.90	12	Gold Loan
9	2.70	10	Education Loan
10	6.20	8	Car Loan

Loan Offers

The below query is used to fetch all the records of loan Offers table.

Figure 18: Selecting loan offers data

```
5 •  SELECT * FROM loan.loan_offers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

offer_id	ending_date	loan_name	loan_range	starting_date	status	type_id
1	2022-06-28 12:13:12	Vidya Loan	200000.00	2022-05-25 08:03:12	A	2
2	2022-07-12 08:03:12	MSME Loan	120000.00	2022-05-26 08:03:12	A	3
3	2022-08-23 23:03:12	PMMY Loan	18000.00	2022-06-12 08:03:12	A	4
4	2022-09-12 11:03:12	CGFMSE Loan	90000.00	2022-05-30 08:03:12	A	6
5	2022-10-21 09:03:12	NSIC Loan	23000.00	2022-06-03 08:03:12	A	1
6	2022-11-12 05:03:12	CLCSS Loan	13000.00	2022-06-30 08:03:12	A	9
7	2022-12-23 12:03:12	FDF Loan	7800.00	2022-06-21 08:03:12	A	7
8	2023-10-23 02:03:12	MMR Loan	6700.00	2022-06-21 08:03:12	A	3
9	2024-01-13 11:03:12	RTL Loan	4200.00	2022-06-17 08:03:12	A	10
10	2023-12-27 12:04:56	DLM Loan	6700.00	2022-08-21 08:03:12	A	1

Loan Request

The below query is used to fetch all the records of loan request table.

Figure 19: Selecting loan request data

```
5 •  SELECT * FROM loan.loan_request;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

request_id	amount	emi_months	requested_date	purpose	status	branch_id	customer_id	offer_id
1	3000.00	24	2022-05-25 08:03:16	To Study	I	1	1	1
2	4500.00	12	2022-05-22 16:23:32	Buy Car	A	2	5	7
3	5600.00	36	2022-05-22 18:03:53	Home Construction	A	5	7	9
4	5500.00	24	2022-05-18 02:03:45	Education	I	9	3	6
5	7000.00	24	2022-05-20 10:13:17	Home Construction	I	5	9	3
6	5300.00	48	2022-04-30 10:13:17	Buy Car	A	7	5	3
7	6900.00	12	2022-05-19 18:56:56	Home Construction	I	5	3	9
8	5000.00	60	2022-05-19 21:45:43	To buy Vehicle	A	2	5	9
9	9800.00	24	2022-05-21 11:59:31	To Study	I	5	9	1
10	2300.00	6	2022-05-25 20:28:45	Car Loan	A	6	8	4

Loan Information

The below query is used to fetch all the records of loan information table.

Figure 20: Selecting loan information data

```
5 •  SELECT * FROM loan.loan_information;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

info_id	loan_date	loan_due_date	customer_id	request_id
1	2022-05-28 08:03:12	2024-05-28 08:03:12	1	1
2	2022-05-22 18:23:22	2024-05-28 18:23:22	2	8
3	2022-04-18 21:12:19	2023-04-28 21:12:19	6	3
4	2022-05-13 16:26:59	2023-05-13 16:26:59	1	9
5	2022-05-18 16:26:59	2023-05-18 16:26:59	6	1
6	2022-05-21 23:45:42	2023-05-21 23:45:42	4	3
7	2022-04-29 21:53:59	2023-04-29 21:53:59	6	5
8	2022-05-07 15:32:32	2023-05-07 15:32:32	8	7
9	2022-04-26 22:56:16	2023-04-26 22:56:16	3	3
10	2022-05-13 16:26:59	2023-05-13 16:26:59	9	8

Emi

The below query is used to fetch all the records of loan information table.

Figure 21: Selecting EMI data

```
5 •  SELECT * FROM loan.emi;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

emi_id	current_balance	payment	payment_date	period	status	customer_id	info_id
1	3000.00	300.00	2022-06-28 08:03:12	1	C	1	3
2	2700.00	300.00	2022-07-28 08:03:12	2	C	1	3
3	2400.00	300.00	2022-08-28 08:03:12	3	C	1	3
4	2100.00	300.00	2022-09-28 08:03:12	4	C	1	3
5	1800.00	300.00	2022-10-28 08:03:12	5	C	1	3
6	1500.00	300.00	2022-11-28 08:03:12	6	C	1	3
7	1200.00	300.00	2022-12-28 08:03:12	7	C	1	3
8	900.00	300.00	2023-01-28 08:03:12	8	C	1	3
9	600.00	300.00	2023-02-28 08:03:12	9	C	1	3
10	300.00	300.00	2023-03-28 08:03:12	10	U	1	3

Payment Info

The below query is used to fetch all the records of Payment information table.

Figure 22: Selecting payment info data

```
5 •  SELECT * FROM loan.payment_info;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

payment_id	payment_amount	payment_date	payment_type	status	emi_id
1	300.00	2022-05-25 08:19:28	Online Banking	P	1
2	300.00	2022-05-25 08:21:33	Online Banking	C	1
3	300.00	2022-05-24 18:29:56	Credit	C	3
4	300.00	2022-05-15 08:19:28	Debit	P	4
5	300.00	2022-05-15 09:57:34	Debit	C	4
6	300.00	2022-05-24 03:43:55	Credit	C	5
7	300.00	2022-05-15 13:54:29	Online Banking	C	7
8	300.00	2022-05-04 04:23:56	Zelle	P	2
9	300.00	2022-05-04 06:34:45	Credit	C	2
10	300.00	2022-05-21 12:56:42	Debit	C	9

User Activity

The below query is used to fetch all the records of User Activity table.

Figure 23: Selecting User Activity data

```
5 •  SELECT * FROM loan.user_activity;
```

Result Grid | Filter Rows: | Edit:

activity_id	activity_time	activity_type	customer_id
1	2022-05-28 08:23:12	Login	1
2	2022-05-28 20:52:22	Logout	1
3	2022-05-28 20:53:15	Login	7
4	2022-05-28 20:54:22	Pwd Changed	7
5	2022-05-28 21:04:08	Loan Applied	7
6	2022-05-28 21:14:57	Logout	7
7	2022-05-28 21:24:33	Login	9
8	2022-05-28 21:25:56	EMI Paid	9
9	2022-05-28 21:31:34	Logout	9
10	2022-05-28 21:54:07	Login	5

12.2 Order By

The below query is used to get employees from employees data where last name starting with ‘s’ and getting results order by ssn id descending order.

Figure 24: Order By

```
5 •  SELECT * FROM loan.employee WHERE last_name LIKE "s%" ORDER BY ssn DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

emp_id	address	email_id	first_name	last_name	contact_num	password	ssn	branch_id
7	424 Woodruff Road	josephsus@gmail.com	Joseph	Susan	8462345634	joseph@123	9676346643	6
9	67 Oxford Road	charlessa@gmail.com	Charles	Sarah	4363847624	charles@123	4274678245	8

12.3 Joins

The query below is used to fetch the records from both the tables. The customer who requested the loan that information we will show here.

Figure 25: Joins

```
5 •  SELECT * FROM loan.customer c inner join loan.loan_request r on c.customer_id = r.customer_id;
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

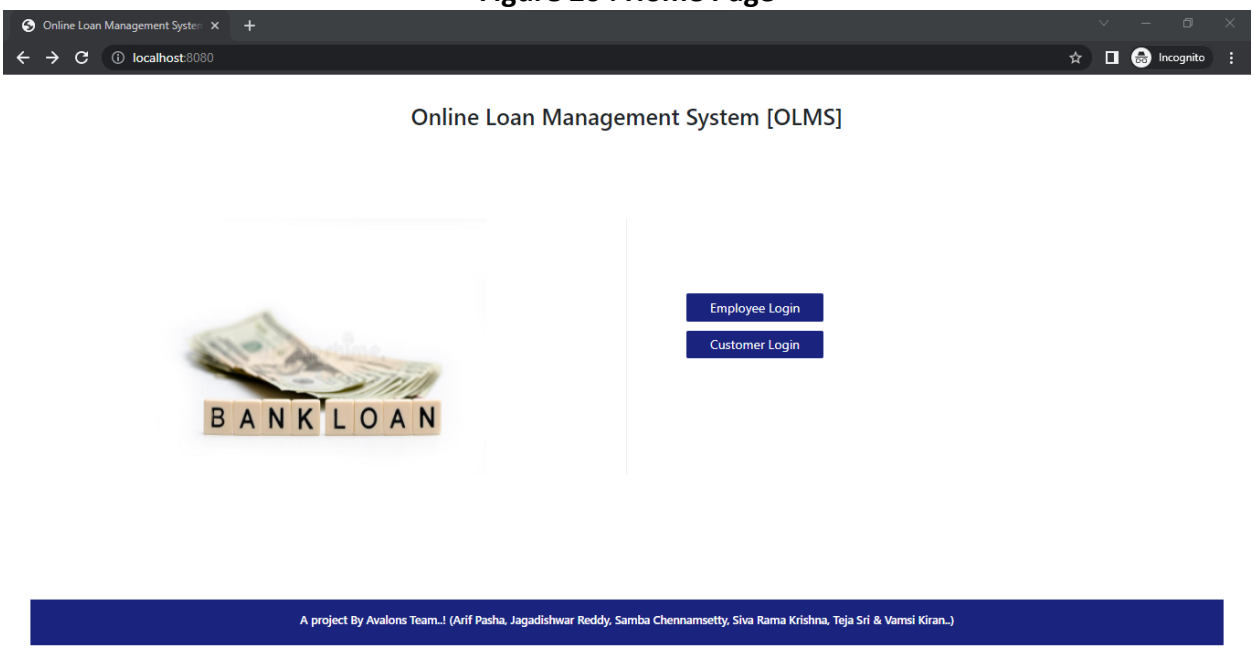
customer_id	address	dob	email_id	first_name	gender	last_name	contact_num	occupation	password	ssn_number	request_id	amount	emi_months	requested_da
1	225 Taft Avenue	1998-05-25 07:28:00	arifshaik@gmail.com	Arif	M	Shaik	9877896544	Student	arif@123	9876543210	1	3000.00	24	2022-05-25 08
5	189 Old Tavern Street	1997-08-23 16:09:33	tejasri@gmail.com	Teja	F	Ravula	2983749243	Student	tejasri@123	3482309844	2	4500.00	12	2022-05-22 16
7	424 Woodruff Road	1996-03-29 03:23:34	harich@gmail.com	Hari	M	Chennamsetty	3495803458	Employee	hari@123	2340086749	3	5600.00	36	2022-05-22 18
3	396 Gregory Street	1997-04-28 08:01:12	samba.ch97@gmail.com	Samba	M	Chennamsetty	8008051986	Student	samba@123	1234567890	4	5500.00	24	2022-05-18 02
9	67 Oxford Road	1999-04-12 21:53:53	harsha@gmail.com	Harsha	M	Chennamsetty	3068480439	Employee	harsha@123	5349435794	5	7000.00	24	2022-05-20 10
5	189 Old Tavern Street	1997-08-23 16:09:33	tejasri@gmail.com	Teja	F	Ravula	2983749243	Student	tejasri@123	3482309844	6	5300.00	48	2022-04-30 10
3	396 Gregory Street	1997-04-28 08:01:12	samba.ch97@gmail.com	Samba	M	Chennamsetty	8008051986	Student	samba@123	1234567890	7	6900.00	12	2022-05-19 18
5	189 Old Tavern Street	1997-08-23 16:09:33	tejasri@gmail.com	Teja	F	Ravula	2983749243	Student	tejasri@123	3482309844	8	5000.00	60	2022-05-19 21
9	67 Oxford Road	1999-04-12 21:53:53	harsha@gmail.com	Harsha	M	Chennamsetty	3068480439	Employee	harsha@123	5349435794	9	9800.00	24	2022-05-21 11
8	34 Roses Mill Road	1994-07-12 04:23:45	saich@gmail.com	Sai	M	Chennamsetty	5385039485	Employee	sai@123	3804557859	10	2300.00	6	2022-05-25 20

13 Graphical User Interface

13.1 Home Page

This is the home page of Online Loan Management System [OLMS]. When user opens the website, they have to select which type of user they are. In this application we have two types of users, Customer and Employee. Employee belongs to any one of the branch and customers can login directly if has account already otherwise they can create account in register page.

Figure 26 : Home Page



13.2 Database Connection

Connecting to MySQL server, we used Java Spring Framework to database. For that developer has to provide driver class name, url , user name and password to database.

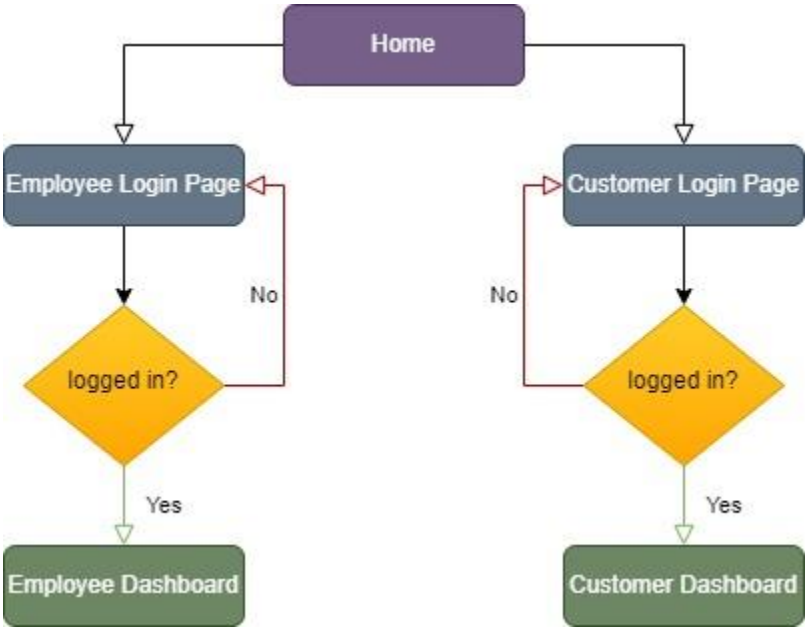
Figure 27: Database Connection

```
# Database connection properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/loan?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=root
```

13.3 Login Flow Chart

The below picture describes flow chart of application login of both employee and customer. Wrong details cause invalid credentials and remains on same page. Successful login forward to dashboards.

Figure 28: Login flow chart

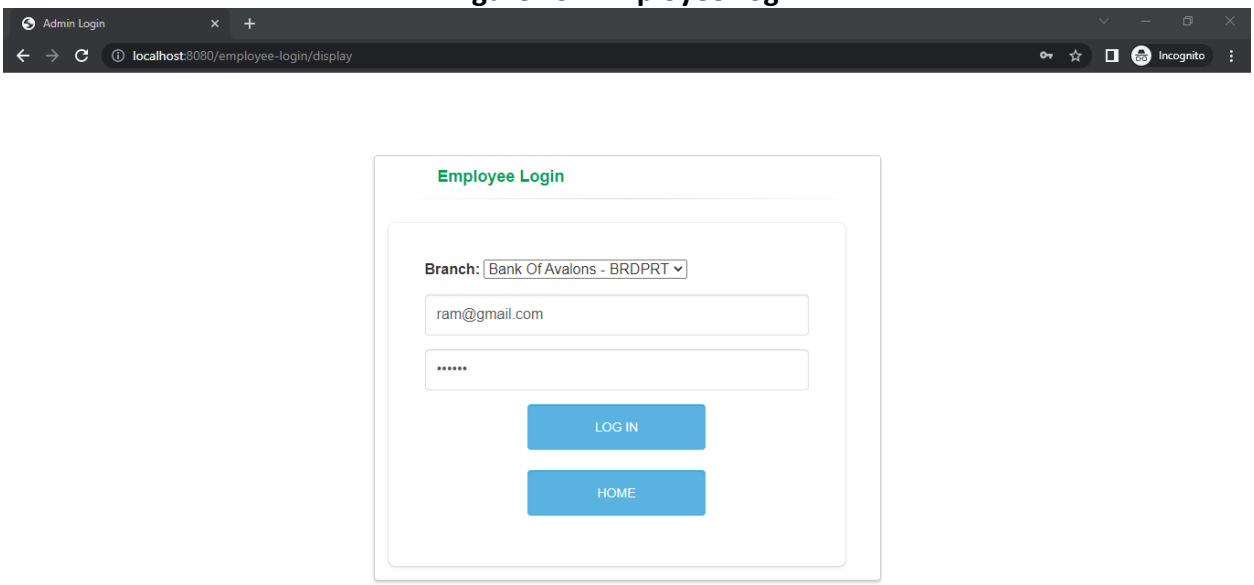


13.4 Employee Login

Employee has login to particular branch account with their credentials, wrong credentials would throw invalid credentials error. Every employee must be assigned to particular branch, that information stores in employee table with branch id as foreign key.

```
"SELECT * FROM loan.branch"
```

Figure 29 : Employee Login

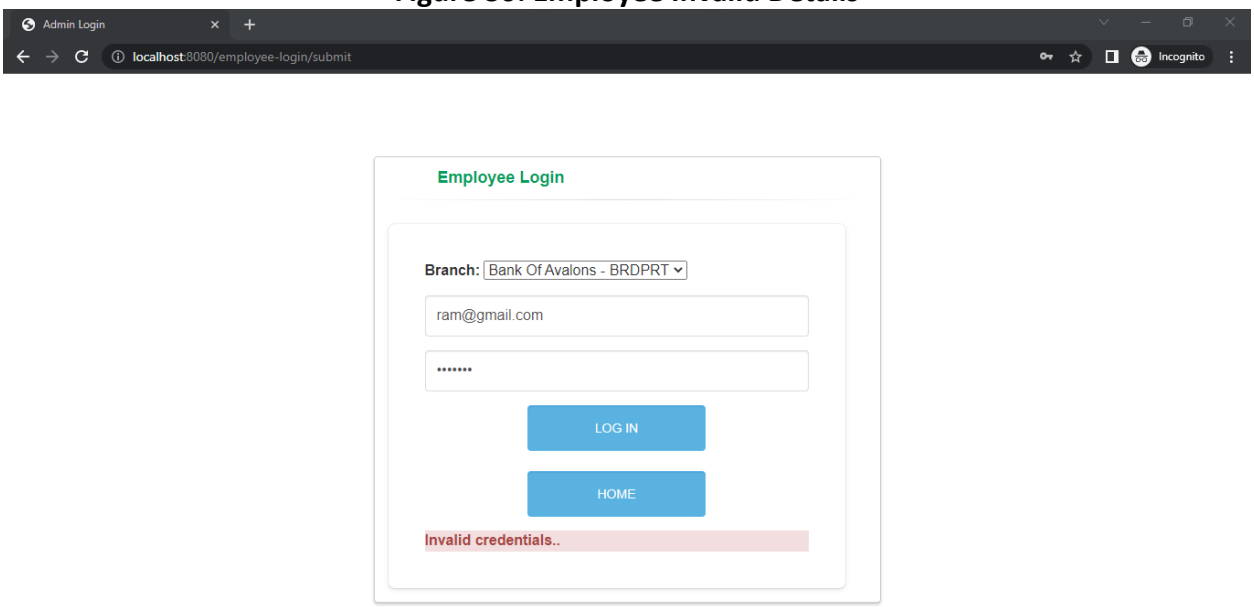


13.4.1 Invalid Credentials

Wrong credentials in employee login page would cause invalid login. Employee has to choose correct branch and right information to log into employee dashboard.

```
"SELECT * FROM loan.employee a WHERE a.email_id = :emailId and a.password =
:password and a.branch_id = :branchId"
```

Figure 30: Employee Invalid Details

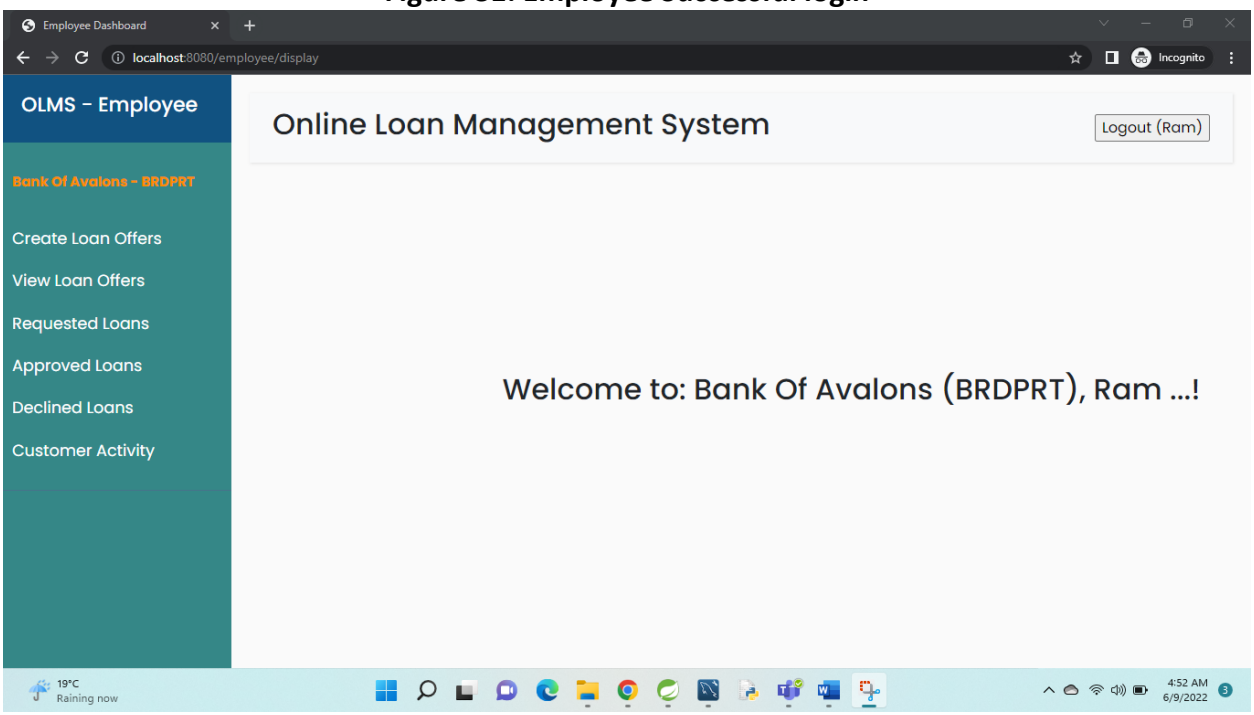


13.4.2 Valid Credentials – Employee Dashboard

Proper branch choose and correct login information would choose to successful login to employee dashboard.

```
"SELECT * FROM loan.employee a WHERE a.email_id = :emailId and a.password =
:password and a.branch_id = :branchId"
```

Figure 31: Employee Successful login

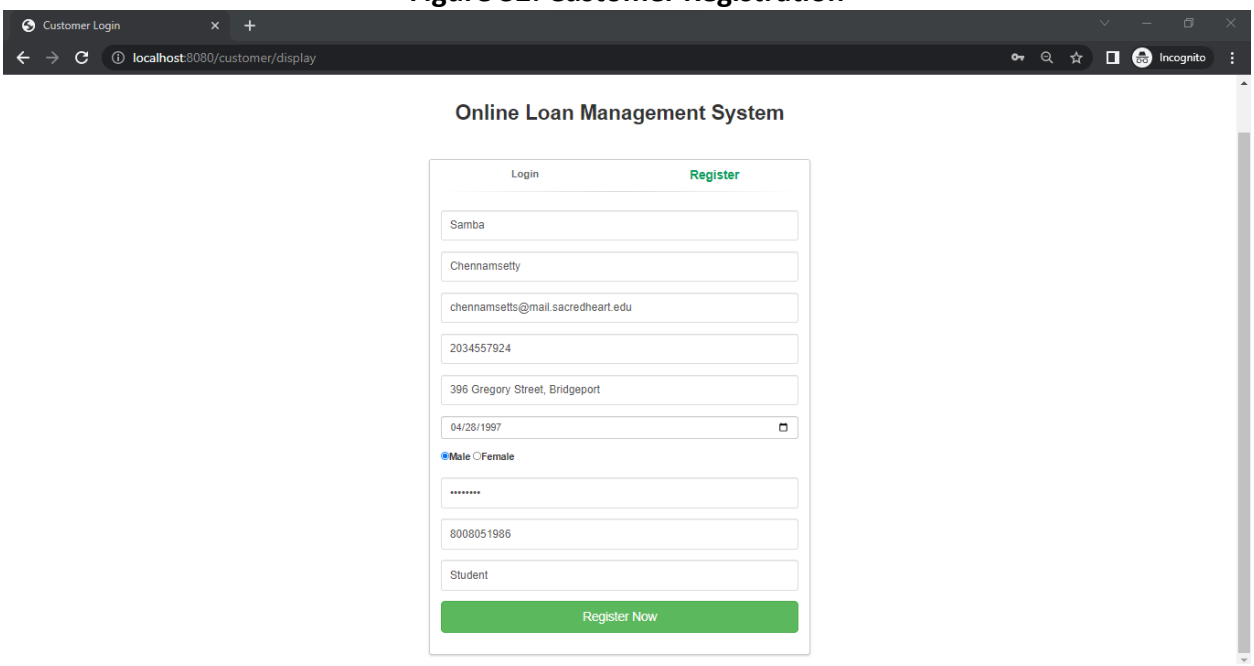


13.5 Customer Registration

Customer has to register first to login into customer dashboard. For that customer has to enter all the details which are shown customer registration page. Once customer registered successfully, can be able to login with provided details.

```
"insert into loan.customer (address, dob, email_id, first_name, gender, last_name, contact_num, password, ssn_number,occupation) values (:#{#customer.address}, :#{#customer.dob}, :#{#customer.emailId}, :#{#customer.firstName}, :#{#customer.gender}, :#{#customer.lastName}, :#{#customer.mobileNumber}, :#{#customer.password}, :#{#customer.ssnNumber}, :#{#customer.occupation})"
```

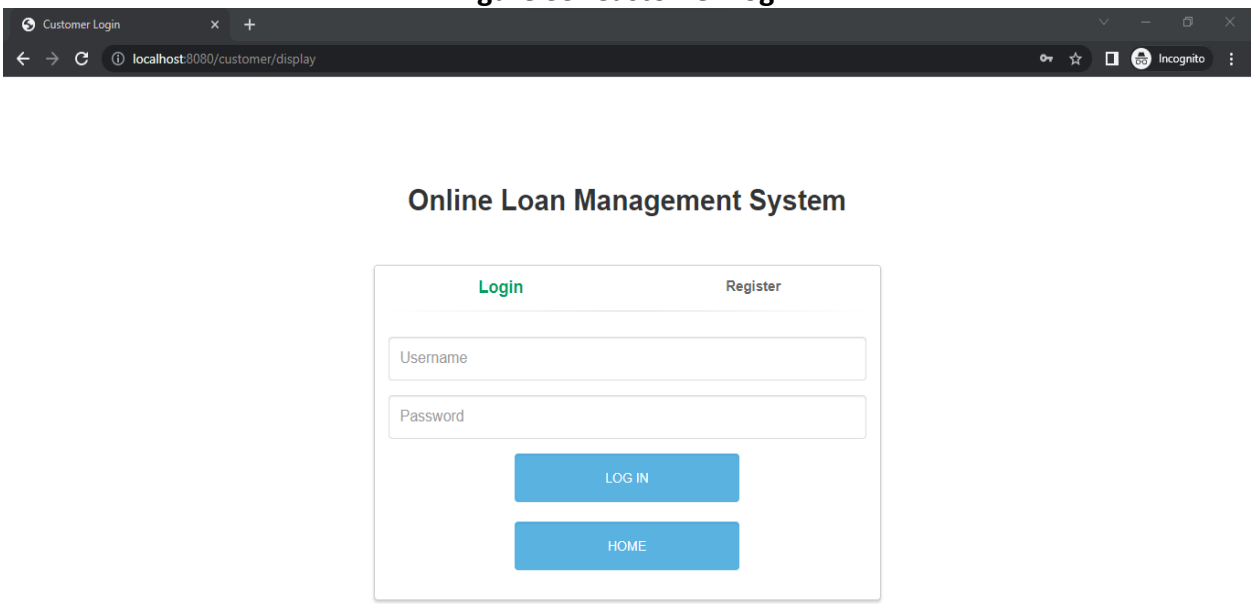
Figure 32: Customer Registration



13.6 Customer Login

Customers has to login into customer login page with their credentials. Wrong information would cause invalid credentials. If customer don't have account can create account in register page. Successful login will move to customer dashboard.

Figure 33: Customer Login

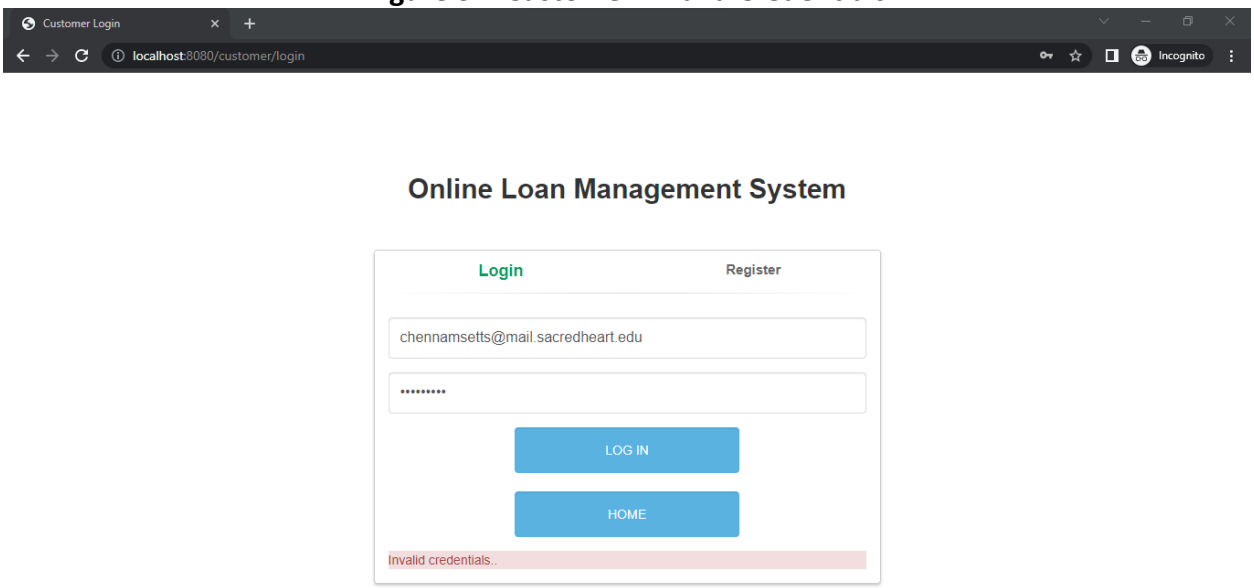


13.6.1 Invalid Credentials

Wrong credentials in customer login page would cause invalid login. Customer has to enter correct information to log into customer dashboard.

```
"select * from loan.customer c where c.email_id = :customerEmailId and c.password
= :password"
```

Figure 34: Customer Invalid Credentials

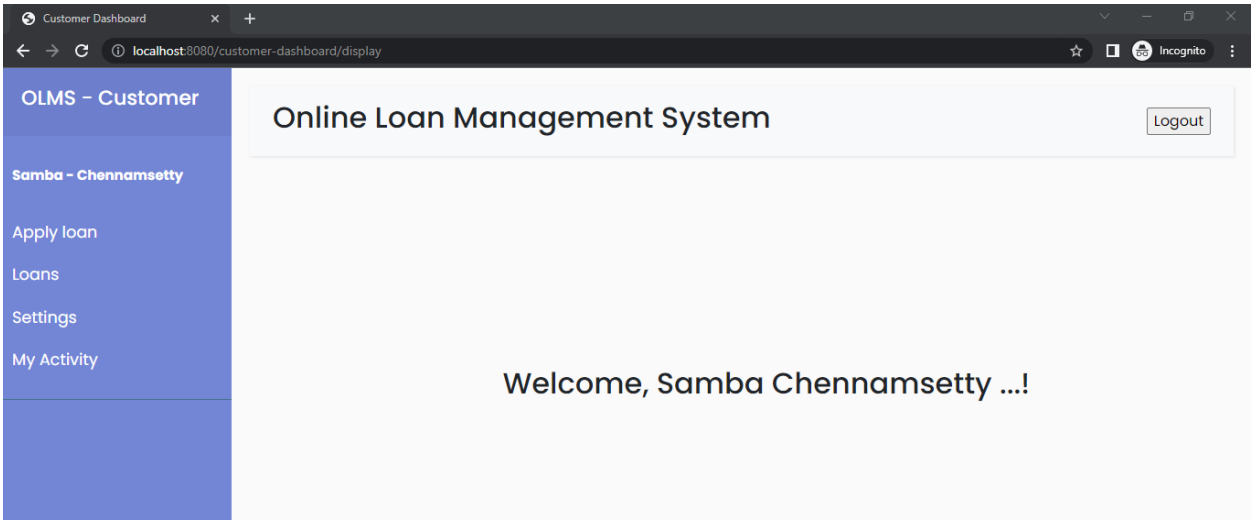


13.6.2 Valid Credentials - Customer Dashboard

Correct information of the customer will choose to successful login and customer dashboard.

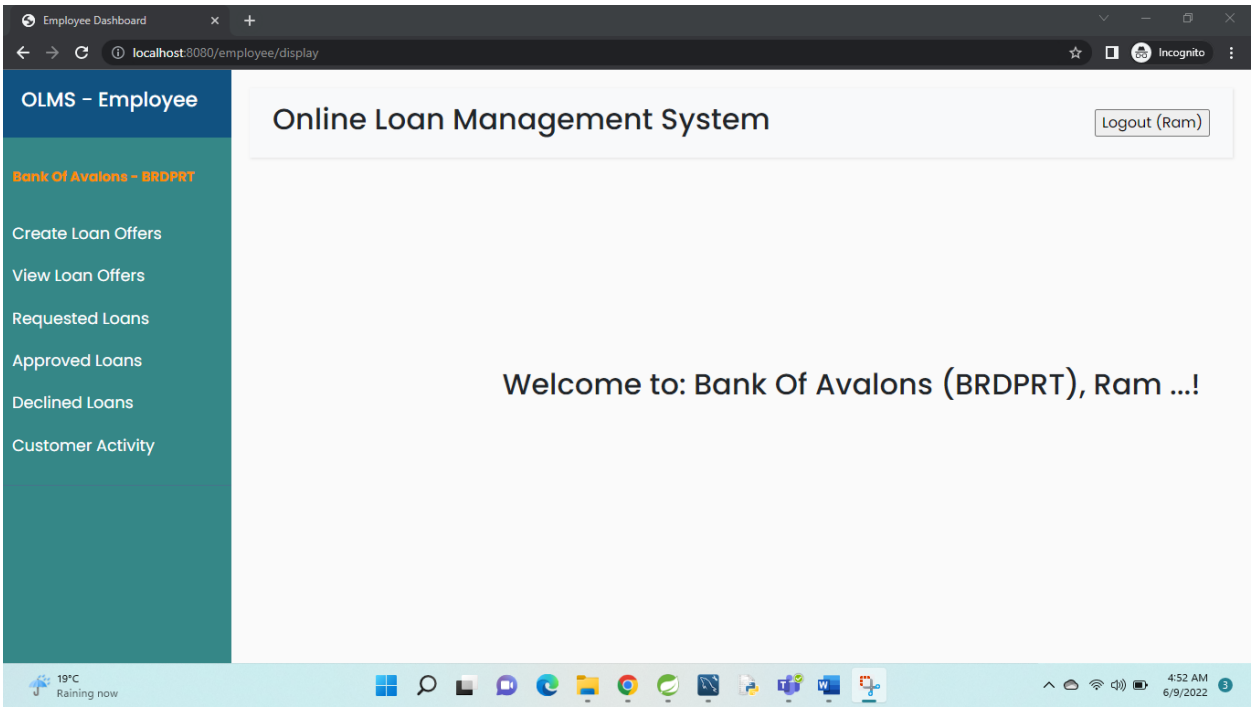
```
"select * from loan.customer c where c.email_id = :customerEmailId and c.password
= :password"
```

Figure 35: Customer Successful Login



13.7 Employee Dashboard

Employee dashboard has multiple pages for creating loan offers for customers, view loan offers, approve loan requests from customer, approved loan lists, declined loan lists and customer activity tracking.



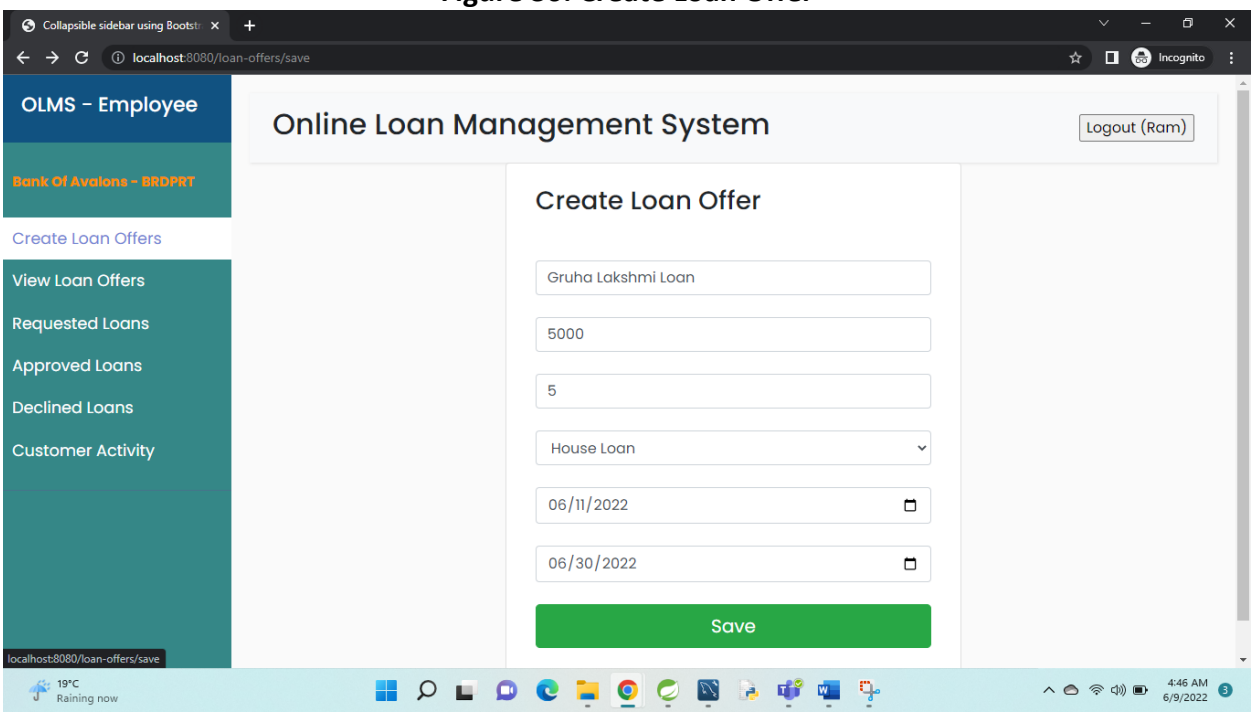
13.7.1 Create Loan Offers

Employee has to create loan offers to the customers from “Create Loan Offers” page. For that he has to select loan type from select box which are coming from loan_type table. Once admin fill the form and creates the loan offer, it would be displayed in customer loan application form and admins “View Loan Offers” page.

```
"SELECT * FROM loan_type;"
```

```
"INSERT INTO loan_offers (loan_name, loan_range, starting_date, ending_date,
status,type_id,interest)values(:#{#loanOffer.loanName},:#{#loanOffer.loanRange
},:#{#loanOffer.startingDate},:#{#loanOffer.endingDate},'A',
:#{#loanOffer.loanTypeId}, :#{#loanOffer.interest})"
```

Figure 36: Create Loan Offer

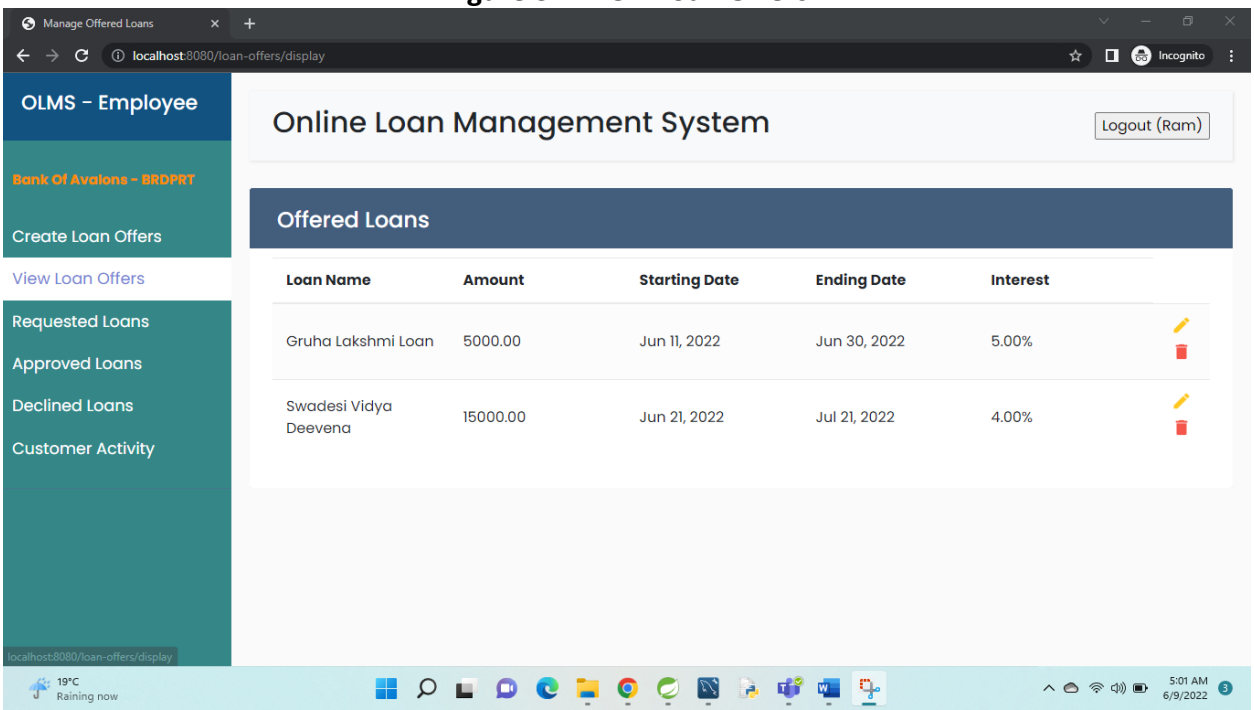


13.7.2 View Loan Offers

Admin can see created "Loan offers" from "View Loan Offers" page. From this page he can update/delete the created "loan offers"

```
"SELECT * FROM loan_offers;"
```

Figure 37: View Loan Offers



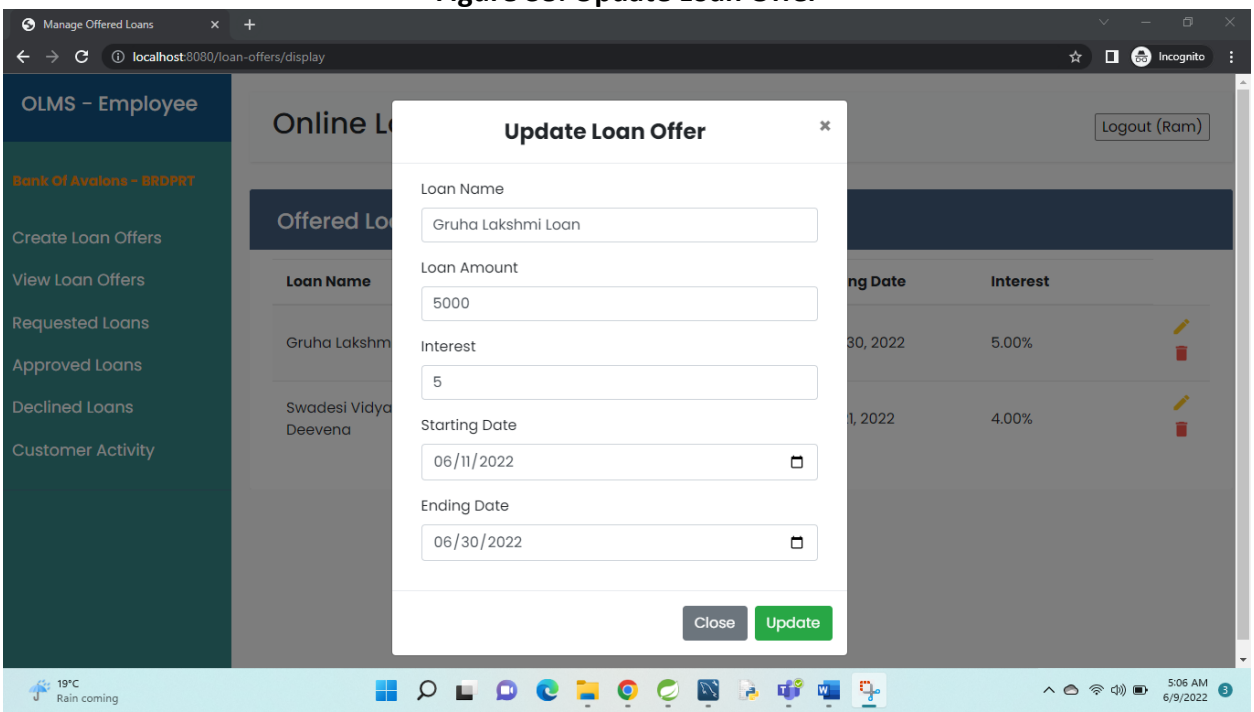
13.7.3 Update Loan Offer

Admin able to update the created loan offer from “View Loan Offers” page, for that admin has to select edit icon which is right side of each loan offers. When clicks it a modal will pop-up with the existing information. From there admin can update the information.

```
"SELECT * FROM loan_offers lo WHERE lo.offer_id = :id"
```

```
"UPDATE loan_offers lo SET lo.loan_range = :#{#loanOffer.loanRange}, lo.interest
=:#{#loanOffer.interest},lo.loan_name=:#{#loanOffer.loanName},lo.starting_date
= :#{#loanOffer.startingDate},lo.ending_date = :#{#loanOffer.endingDate} WHERE
lo.offer_id = :#{#loanOffer.offerId}"
```

Figure 38: Update Loan Offer

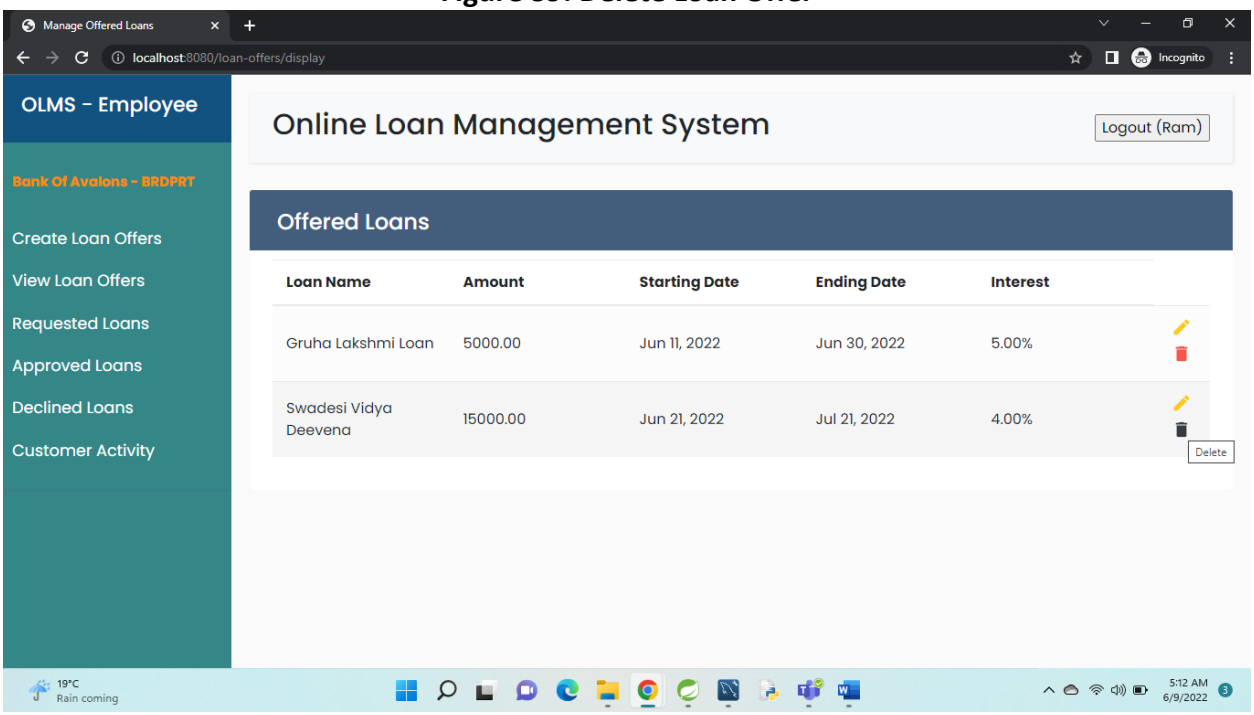


13.7.4 Delete Loan Offer

Admin able to delete the loan offer by clicking on delete icon which is right side of each loan offer.

```
"DELETE FROM loan.loan_offers lo WHERE lo.offer_id = :id"
```

Figure 39: Delete Loan Offer



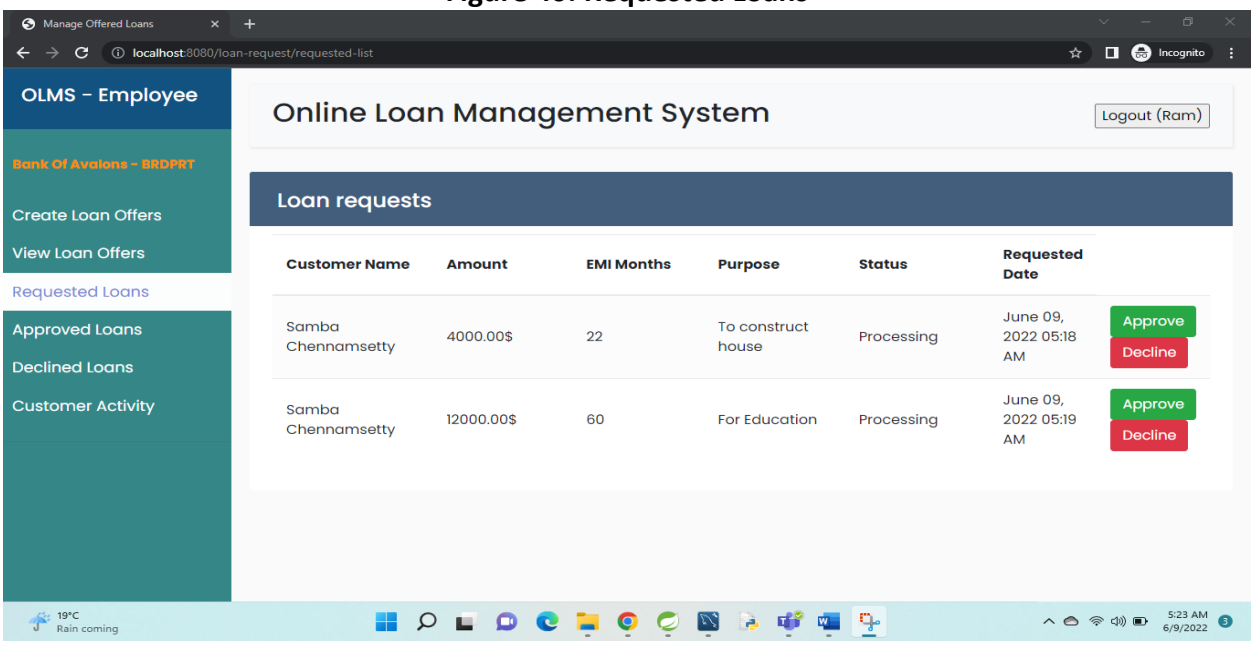
13.7.5 Requested Loans

Admin has to approve requested loan from customers. Once customer apply for the loan for the particular branch, the loan request goes to that bank only. Loan requests will be able approve/decline by the employee from “Requested Loans” page.

```
"SELECT * FROM loan_request req WHERE req.branch_id = :branchId AND req.status
= :status"
```

```
"UPDATE loan_request lr SET lr.status = :status WHERE lr.request_id = :requestId"
```

Figure 40: Requested Loans

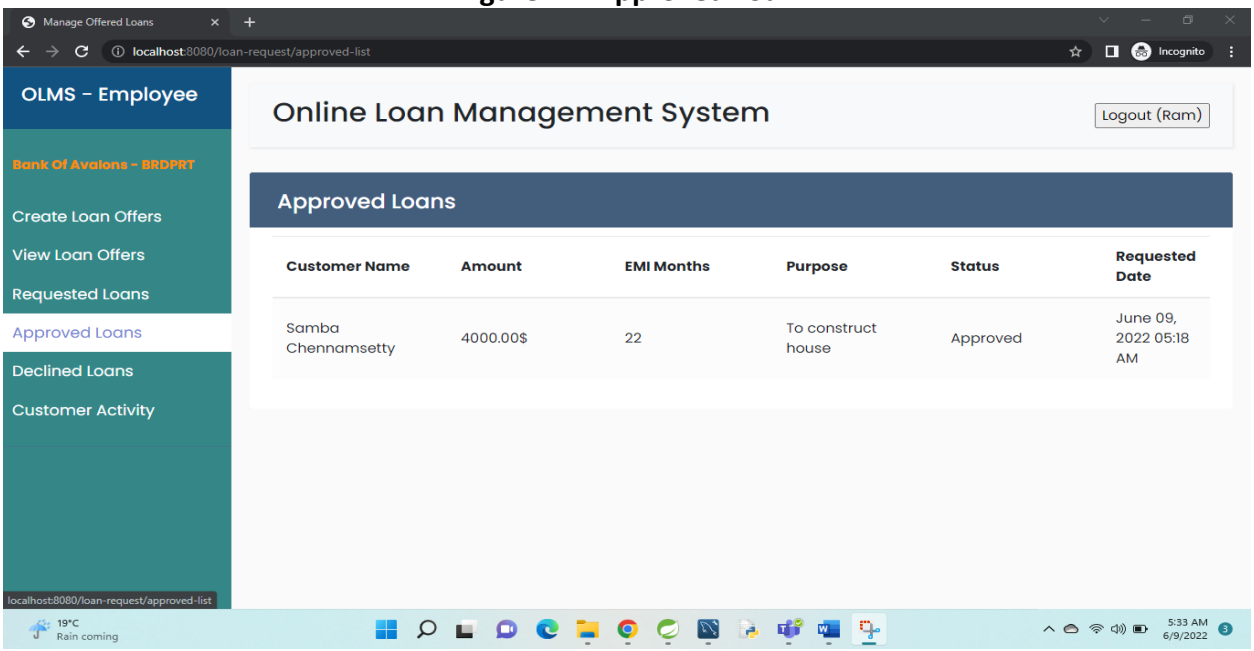


13.7.6 Approved Loans

Once admin approves the loan, that information will come to "Approved loans" page and status will updated to approved in loan request table. Simultaneously new record will be created in loan_information table with the provided information and emi's will be created in emi table. User able to see approved loans from "loans" page from customer dashboard.

```
"SELECT * FROM loan_request req WHERE req.branch_id = :branchId AND req.status
= :status"
```

Figure 41: Approved Loan

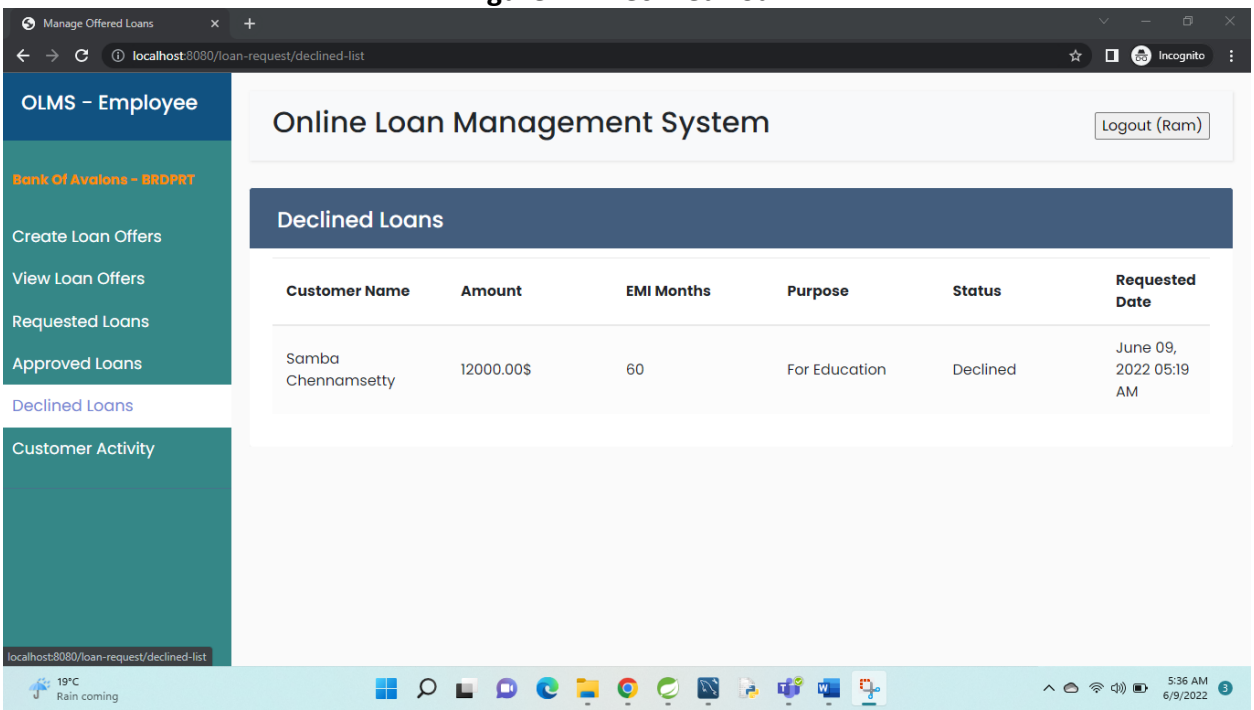


13.7.7 Declined Loans

Admin can decline loans also. Once admin decline loans, loan status will be updated to decline in loan request table and goes to “Declined Loans” page.

```
"SELECT * FROM loan_request req WHERE req.branch_id = :branchId AND req.status = :status"
```

Figure 42: Declined Loan



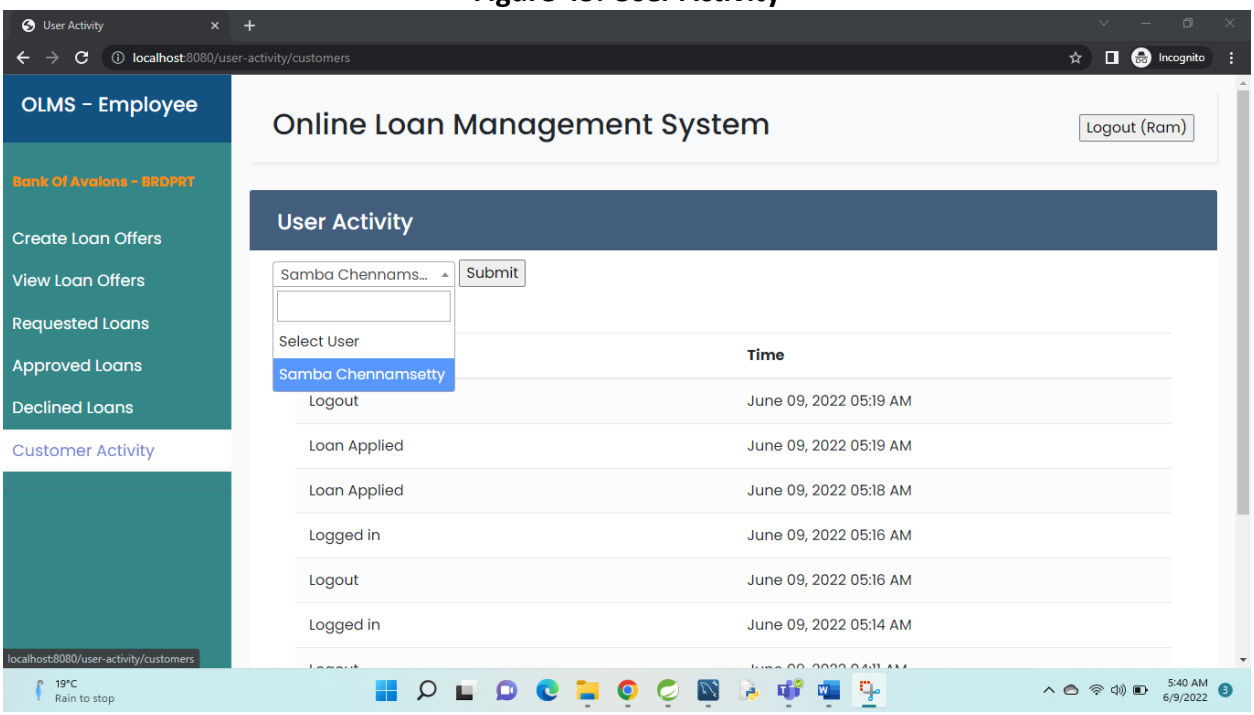
13.7.8 Customer Activity

Admin can monitor/track customer activity in the website. This activity table stores this information. Activities like logged in, logout, loan applied, payment done, profile updated stores in activity table. Admin able see this information from “Customer activity” page by selecting particular customer from the select box.

```
"select * from loan.customer"
```

```
"SELECT * FROM user_activity act WHERE act.customer_id = :customerId order by act.activity_time desc"
```

Figure 43: User Activity

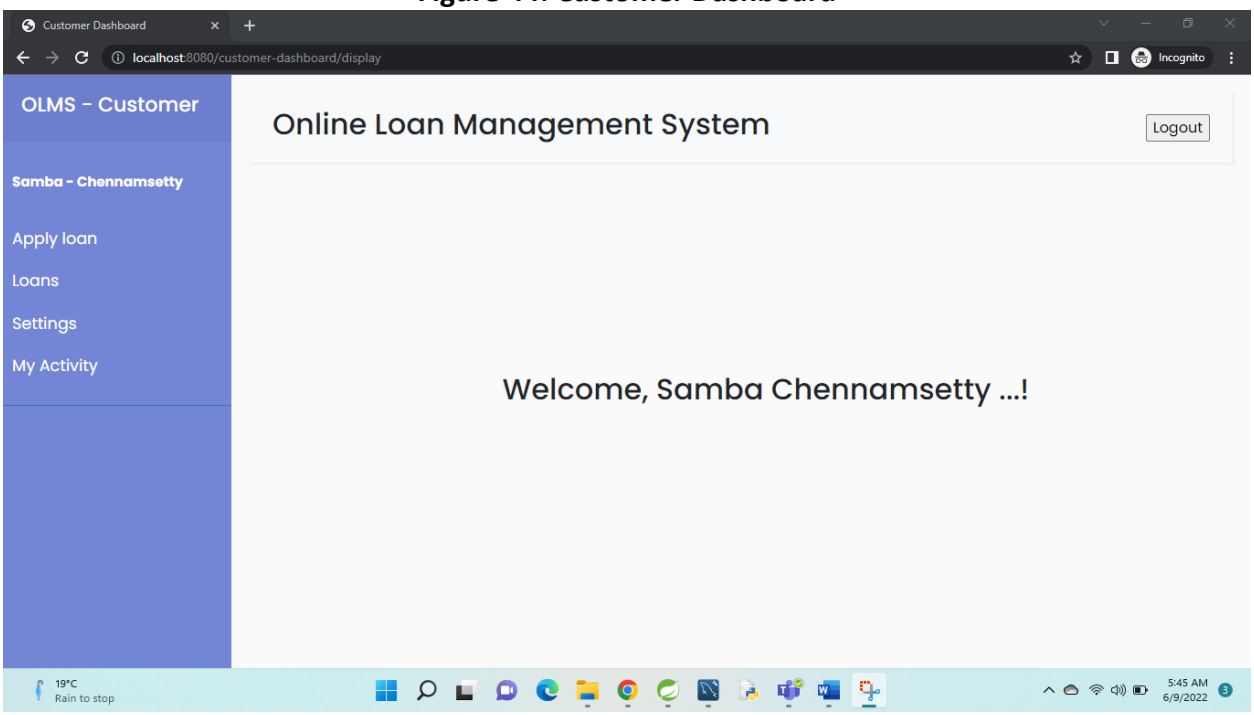


13.8 Customer Dashboard

Once customer logins successfully, redirects to customer dashboard. It has multiple pages for customer like apply loan page to apply loan for the particular branch, loans page to see applied loans, settings page to update own profile and customer activity page to see their activities.

```
"select * from loan.customer c where c.email_id = :customerEmailId and c.password
= :password"
```

Figure 44: Customer Dashboard



13.8.1 Apply Loan

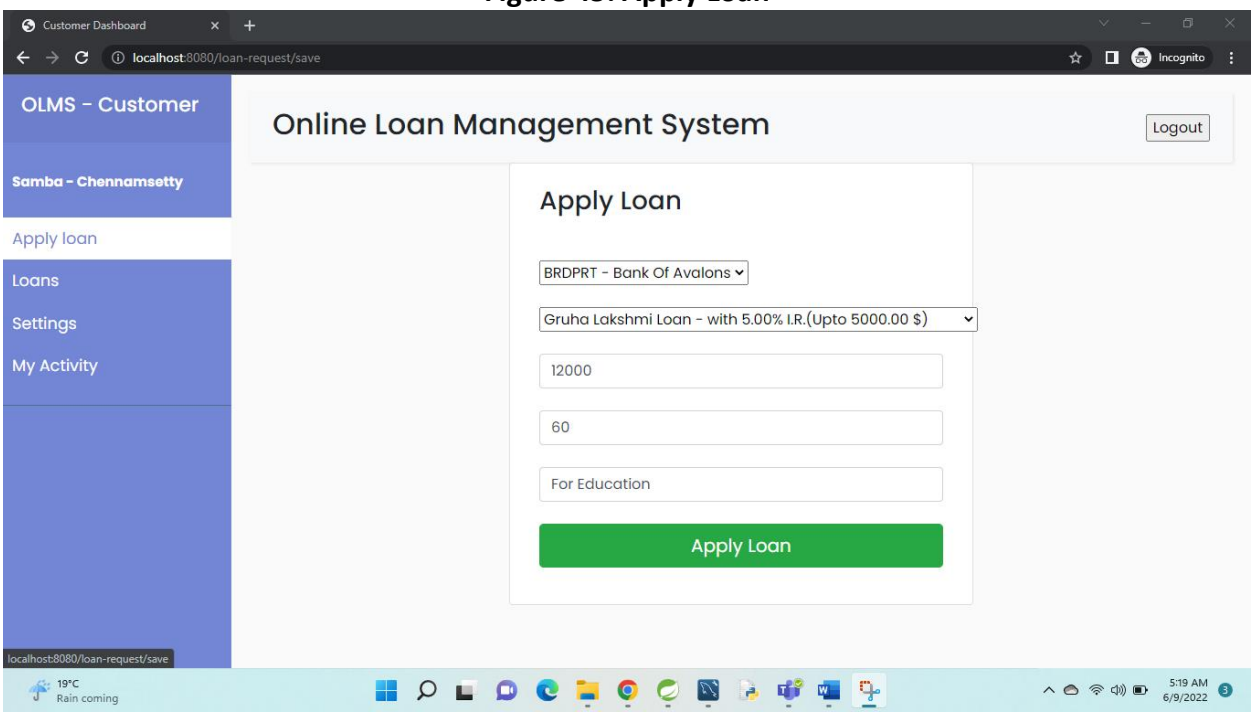
Customer has to apply loan from apply loan page. In this page they have to select to which branch and type of loan they are applying for. Once customer applied the loan this loan request goes to selected branch and goes to processing state. Further admin has to take decision on that. Admin can deny or approve the request.

```
"SELECT * FROM loan.branch"

"SELECT * FROM loan_offers;"

"INSERT INTO loan_request ( emi_months, requested_date, purpose, status, customer_id, offer_id, amount, branch_id) VALUES( :#{#loanRequest.emiMonths}, :#{#loanRequest.loanRequestedDate}, :#{#loanRequest.purpose}, :#{#loanRequest.status}, :#{#loanRequest.customerId}, :#{#loanRequest.offerId}, :#{#loanRequest.amount}, :#{#loanRequest.branchId});"
```

Figure 45: Apply Loan



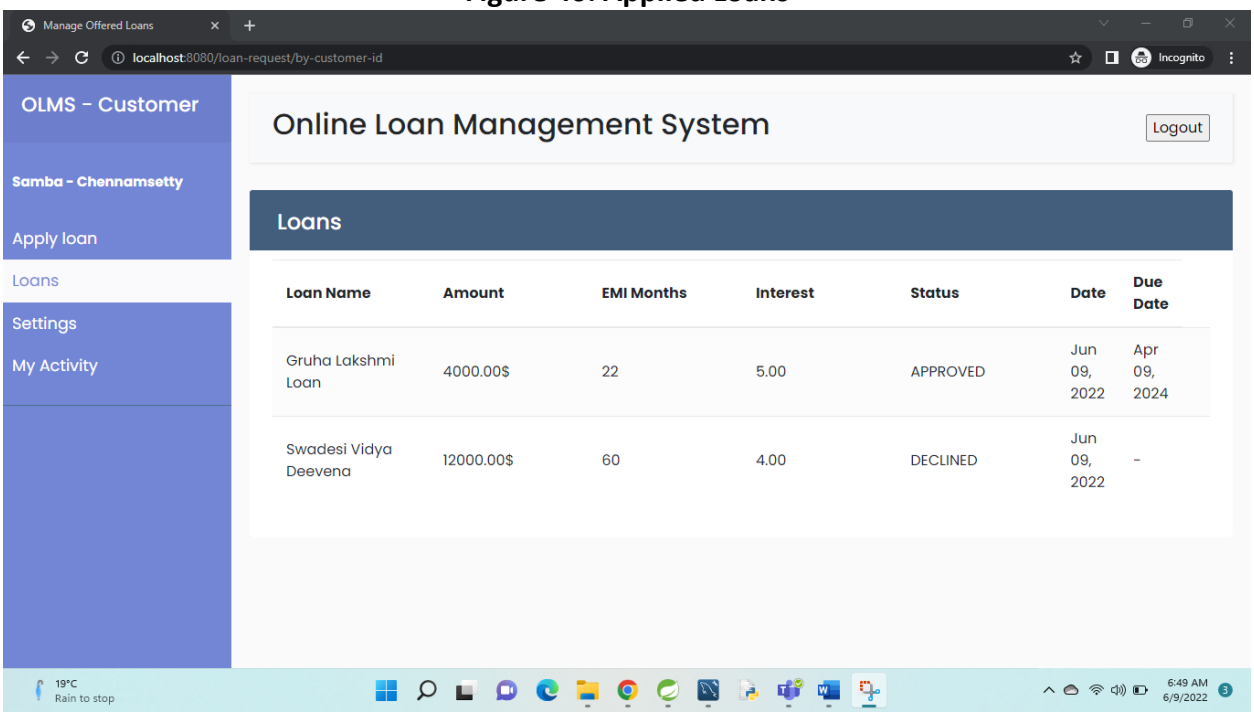
13.8.2 Loans

Customer can see applied loans from “loans” page. This page contains all the applied loans information. Once loan approved from the branch admin, the emi’s for the loan automatically created in emi table and will be able to display when customer selects particular approves loan. If declined no emis created.

```
"SELECT * FROM loan_request req WHERE req.customer_id = :customerId"
```

```
"SELECT * FROM loan_information info WHERE info.request_id = :requestId"
```

Figure 46: Applied Loans

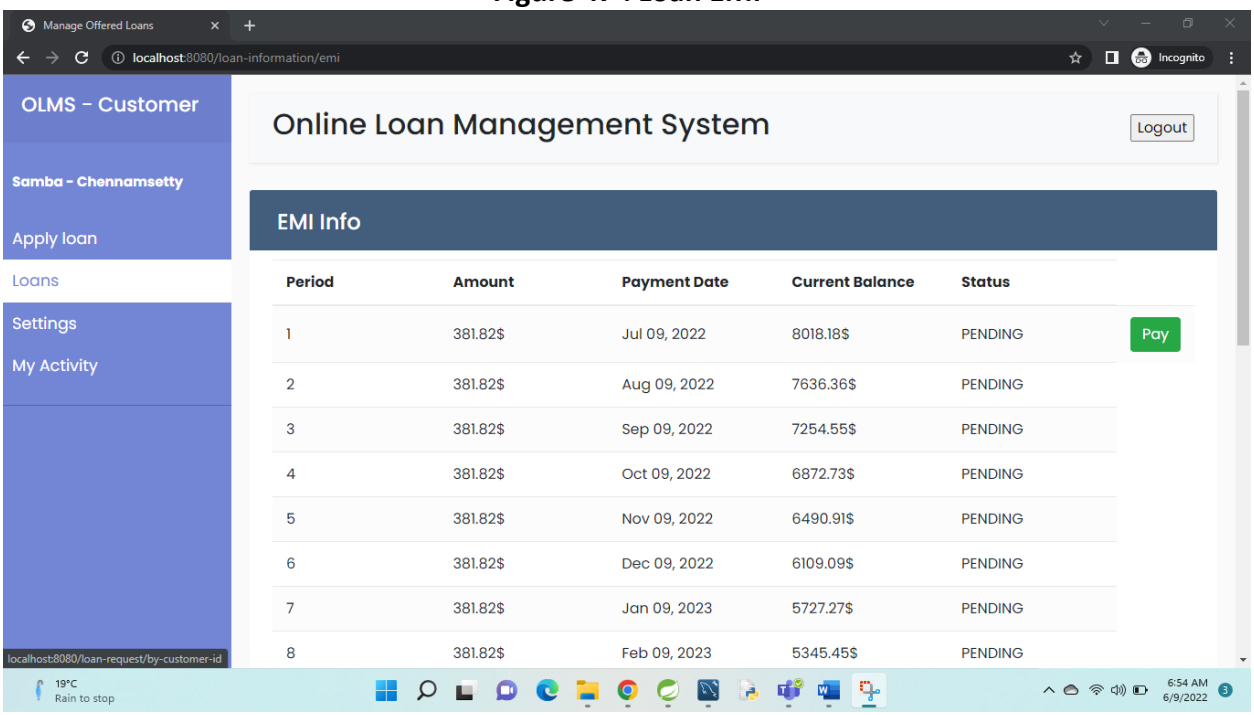


13.8.3 Emi Info

Customer has to click on approved loan to see emi's. Clicking automatically redirects to emi page where All the emi information shows. Each emi records contains how much amount has to pay and current balance and due date along with "Pay" button.

```
"SELECT * FROM emi em WHERE em.info_id = :infoId"
```

Figure 47 : Loan EMI



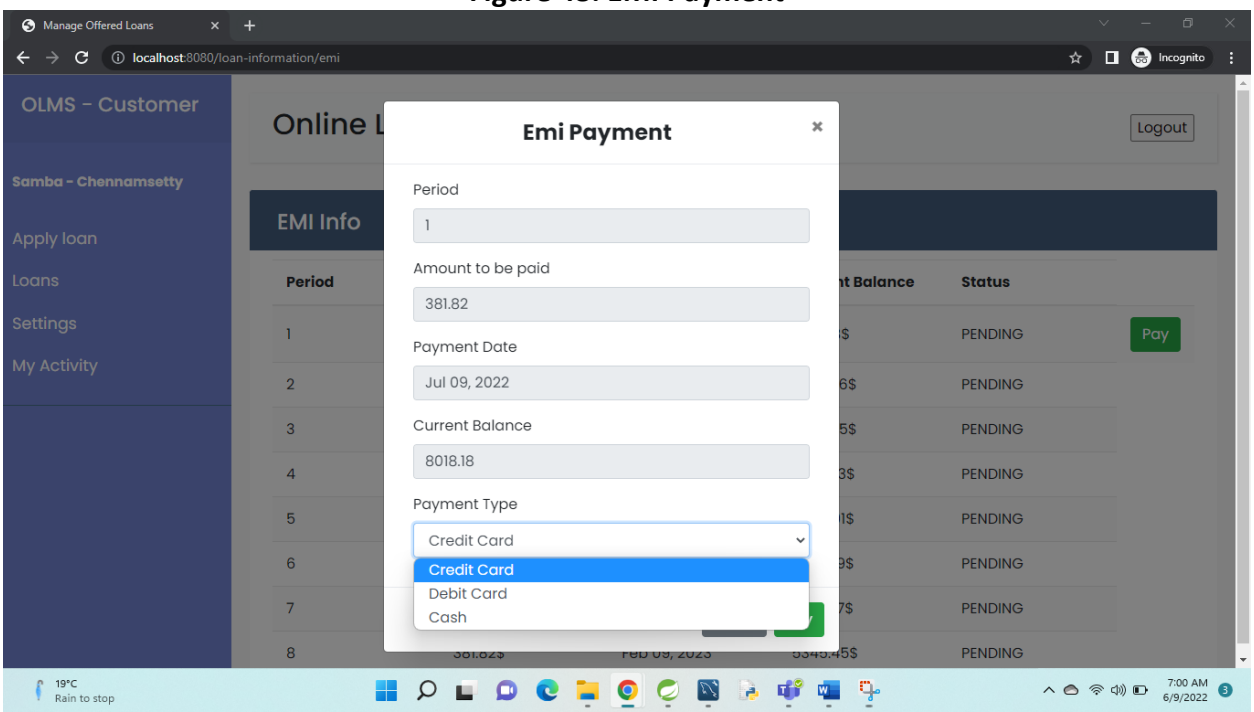
13.8.4 Pay EMI

Customer has to click on “pay” button them modal will pop-up with selected emi information along with customer has to select which type of payment (credit/dedit/cash). Once click on “pay” button emi record updates with “completed” status and record will be created in payment info table along with payment type. Customer can pay single emi at a time. Once emi payment complete pay button automatically shows to next emi.

```
"UPDATE emi em SET em.status = :status WHERE em.emi_id = :emiId"
```

```
"INSERT INTO payment_info ( payment_amount, payment_date, payment_type, status,
emi_id)      VALUES(      :#{#info.paymentAmount},      :#{#info.paymentDate},
:#{#info.paymentType}, :#{#info.status}, :#{#info.emiId});"
```

Figure 48: EMI Payment



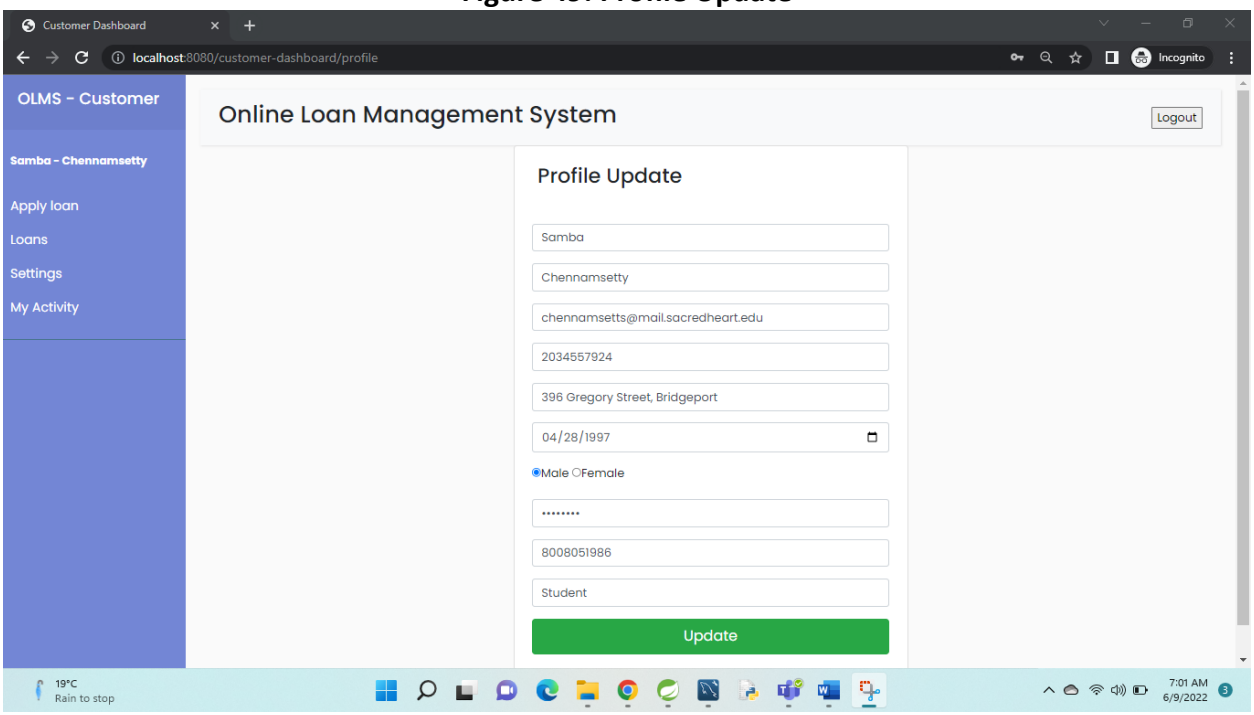
13.8.5 Settings

Customers able to update their profiles by using “settings” page. Settings page will get current customer information and ask customer to update any information.

```
"select * from loan.customer c where c.customer_id = :customerId"
```

```
"UPDATE loan.customer cus SET cus.address = :#{#customer.address}, cus.dob = :#{#customer.dob}, cus.email_id = :#{#customer.emailId}, cus.first_name = :#{#customer.firstName}, cus.gender =:#{#customer.gender}, cus.last_name = :#{#customer.lastName}, cus.contact_num = :#{#customer.mobileNumber}, cus.password = :#{#customer.password}, cus.ssn_number = :#{#customer.ssnNumber}, cus.occupation = :#{#customer.occupation} WHERE cus.customer_id = :#{#customer.customerId}"
```

Figure 49: Profile Update

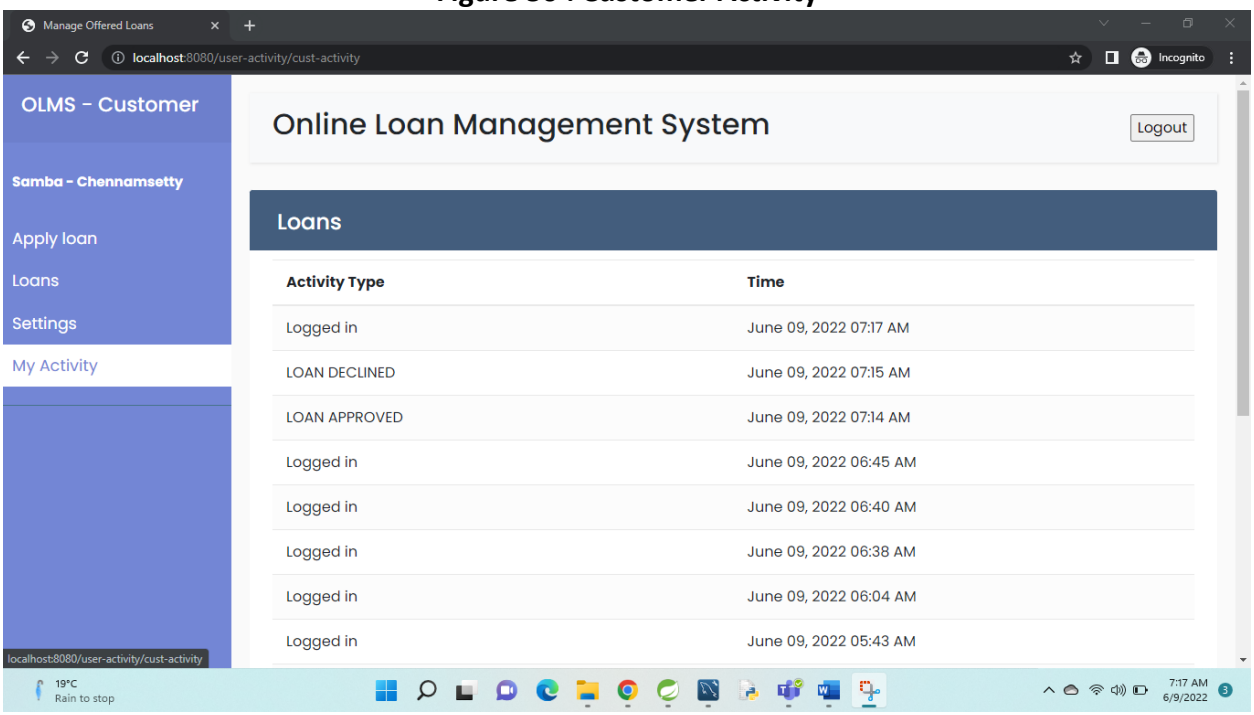


13.8.6 Customer Activity

Customer can see his activity by using "My activity" page. This page shows complete activity of the customer like logged in, logout, loan applied, payment paid etc.

```
"SELECT * FROM user_activity act WHERE act.customer_id = :customerId order by act.activity_time desc"
```

Figure 50 : Customer Activity



14 GitHub Repository

https://github.com/samba-chennamsetty/online-loan-management-system-avalons

15 References

[1] "About Happy Money Loan Company," [Online]. Available: https://happymoney.com/company.

[2] "About PenFred Credit Union," [Online]. Available: https://www.penfed.org/personal/personal-loans.

[3] "Theme Of Light Stream," [Online]. Available: https://www.lightstream.com/about-us.