

BANK LOAN MANAGEMENT SYSTEM
A MINI PROJECT REPORT

Submitted by

SINDHU KALEESWARAN [RA2011026010082]
SHIVANI R [RA2011026010060]
AVANITH KANAMARLAPUDI [RA2011026010073]

Under the guidance of

Dr. R.A KARTHIKA
(Assistant Professor, Dept of Computational Intelligence)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING With specialization in AIML



SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203 APRIL 2023



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report “**Bank Loan Management System**” is the bonafide work of “**Sindhu Kaleeswaran(RA2011026010082), Shivani R (RA2011026010060) and Avanith Kanamarlapudi(RA2011026010073)**” of III Year/VI Sem B.Tech(CSE) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

SIGNATURE

Department of Computational Intelligence

Dr. R. Annie Uthra

HEAD OF THE DEPARTMENT



SIGNATURE

Dr.R.A. Karthika

Assistant professor

CINTEL

ABSTRACT

The Bank Loan Management System is a software application that is designed to simplify the process of managing loans and EMI payments for its customers. The system enables banks to efficiently manage loan applications, approvals, and disbursements, while also enabling customers to easily apply for loans, make EMI payments, and view their loan details. With the Bank Loan and EMI Management System, customers can track their outstanding loan amounts, view their payment history, and make EMI payments conveniently through online payment channels. The system also provides banks with real-time data on loan status, payment schedules, and defaulters, enabling them to take proactive measures to manage their loan portfolio effectively. Overall, the Bank Loan and EMI Management System streamlines the loan management process, improves customer satisfaction, and reduces the risk of loan defaults, making it a valuable tool for banks of all sizes.

TABLE OF CONTENTS

Chapter No	Title	Page No
	ABSTRACT	iii
	TABLE OF CONTENTS	iv
	LIST OF FIGURES	vi
	ABBREVIATIONS	vii
1	CHAPTER 1 – INTRODUCTION	
1.1	About	1
1.2	Problem Statement	1
1.3	1 1.3 Objectives	2
1.4	Scope and Applications	2
1.5	General & Unique Services in Database	2
1.6	Software requirements specifications	3
2	CHAPTER 2 – LITERATURE SURVEY	
2.1	Existing system	4
2.2	Existing vs Proposed system	4
3	CHAPTER 3 - SYSTEM ARCHITECTURE AND DESIGN	
3.1	Architecture Diagram	5
	3.1.1 Frontend (UI) Design	7
	3.1.2 Backend (Database Design)	7
3.2	Use case Diagram	8
3.3	ER Diagram	9
4	CHAPTER 4 – MODULES AND FUNCATIONALITY	
4.1	System Modules and its Functions	11
4.2	Connectivity used for database access	12
5	CHAPTER 5 – CODING AND TESTING	
5.1	Front end code	13
5.2	Back end code	13
5.3	Database	14

5.4	Use cases	
	5.4.1 Home Page	
	5.4.2 Add customer	
	5.4.3 Customer List	
	5.4.4. Avail New Loan	
	5.4.5. Loan List	
	5.4.6 EMI instalment	
6	CHAPTER 6 – RESULTS AND DISCUSSIONS	
6.1	Results	17
6.2	Benefits	17
7	CHAPTER 7 – CONCLUSION	
7.1	Conclusion	19
7.2	Future Enhancements	19
	REFERENCES	20

LIST OF FIGURES

Fig no	Figure Name	Page No
3.1	Architecture Diagram of Loan Management System	6
3.1.1	Frontend Portal	7
3.1.2	Backend Portal	7
3.2	Use Case Diagram	8
3.3	ER Diagram	10
5.1	Front End	11
5.2	Back End	13
5.3	Database of the Portal	14
5.4.1	Home Page	
5.4.2	Add customer	
5.4.3	Customer List	
5.4.4	Avail New Loan	
5.4.5	Loan List	
5.4.6	EMI instalment	
6.1	Results Tab	17

ABBREVIATIONS

SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
ER DIAGRAM	Entity- Relationship Diagram
JS	JavaScript
Node JS	Node JavaScript
UI	User Interface

CHAPTER 1 INTRODUCTION

1.1 About the System

The Bank Loan Management System is a comprehensive software solution designed to streamline the loan management process for banks and improve the overall customer experience. The system is built with advanced features and functionalities that enable banks to manage loan applications, approvals, and disbursements efficiently, while also providing customers with a secure and convenient platform to apply for loans, make EMI payments, and view their loan details.

The system consists of multiple modules that are designed to perform specific functions, such as loan management, customer management and EMI management. Each module is integrated with the others, allowing for seamless data flow.

One of the key benefits of the Bank Loan and EMI Management System is that it simplifies the loan application process for customers, making it easy for them to apply for loans and track their loan details. The system also allow customers to make EMI payments online, reducing the need for manual payment processing and ensuring timely payment of loans.

Functional Specification:

- **Loan Management Module:** This module enables banks to manage loan applications, approvals, and disbursements. It should be able to track loan amount, interest rates, and repayment schedules as well as provide detailed loan histories and repayment schedules.
- **Customer Management Module:** This module should allow customers to view their personal details along with their bank details. Customers should also be able to apply for loans, make EMI payments, and view their payment history through the system's secure online portal.
- **EMI Management Module:** This module enables banks to set up EMI payments, ensuring that customers make timely payments on their loans. The module should also send automated reminders to customers who have missed payments and allow bank officials to take appropriate action in case of loan defaults.

1.2 Problem Statement

The problem statement addressed by the Bank Loan Management System is the need for banks to manage loan applications, approvals, and disbursements more efficiently and provide customers with a convenient and secure platform to apply for loans and make EMI payments. The traditional loan management process can be time-consuming and error-prone, leading to delays in loan processing and disbursements, increased operational costs, and lower customer satisfaction.

The Bank Loan and EMI Management System aims to address these challenges by providing a comprehensive software solution that enables banks to automate and streamline loan management processes. The system offers advanced features such as loan application tracking, EMI management, reporting, and analytics that enable banks to manage their loan portfolios more efficiently and make informed decisions based on data.

For customers, the system provides a secure and user-friendly platform to apply for loans, make EMI payments, and view their loan details, improving their overall experience and reducing the need for manual processes.

By addressing these challenges, the Bank Loan and EMI Management System can help banks reduce operational costs, increase efficiency, and enhance customer satisfaction, ultimately leading to improved business outcomes.

1.3 Objectives

- **Streamline Loan Management Processes:** The system aims to automate loan management processes, such as loan applications, approvals, and disbursements, to reduce manual effort and improve efficiency.
- **Enable Efficient EMI Management:** The system should enable banks to set up and manage automated EMI payments, send reminders to customers who have missed payments, and take appropriate action in case of loan defaults.
- **Improve Customer Experience:** The system should provide customers with a convenient and secure platform to apply for loans, make EMI payments, and view their loan details. This will improve customer satisfaction and reduce the need for manual processes.

1.4 Scope and Application

- **Loan Application Management:** The system should enable banks to manage loan applications efficiently, including document verification, credit checks, and loan approvals.
- **EMI Management:** The system should enable banks to manage EMI payments, automate payment reminders, and track defaulters.
- **Customer Management:** The system should provide a platform for customers to apply for loans, make EMI payments, and view their loan details.

1.5 General and Unique Services

General Services:

1. **Loan Application Management:** The system allows banks to efficiently manage loan applications, including document verification, credit checks, and loan approvals.

2. **EMI Management:** The system enables banks to set up and manage automated EMI payments, send reminders to customers who have missed payments, and take appropriate action in case of loan defaults.
3. **Customer Management:** The system provides customers with a secure and user-friendly platform to apply for loans, make EMI payments, and view their loan details.
4. **Security:** The system has advanced security features such as two-factor authentication, encryption, and secure payment gateways to protect customer data and prevent fraud.

Unique Services:

1. **Customizable Workflows:** The system is customizable to meet the specific needs of different banks, including branding, workflows, and reporting.
2. **Integration with Third-Party Applications:** The system can integrate with other third-party applications, such as accounting software and CRM systems, to ensure seamless data flow and streamline business processes.
3. **Scalability:** The system is scalable to accommodate the growing needs of banks, such as an increasing number of customers, loans, and transactions.

1.6 Software Requirements

The software requirements for our bank loan management system are as follows:

- **Operating System:** The system should be compatible with Windows, Mac OS, and Linux operating systems.
- **Web Server:** Node.js should be installed on the web server to enable server-side programming.
- **Database Management System:** MySQL Workbench should be installed to manage the database used by the system.
- **Framework:** React should be used to build the front-end of the system, while Node.js should be used to build the back-end.
- **Programming Languages:** JavaScript should be used for both front-end and back-end programming.
- **APIs and Libraries:** The system should use APIs and libraries, such as Express.js and Axios, to enable communication between the front-end and back-end.
- **Authentication and Authorization:** The system should include authentication and authorization features to ensure secure access to the system.
- **Payment Gateway:** The system should integrate with a payment gateway, such as PayPal or Stripe, to enable online payments.
- **Security:** The system should have built-in security features, such as SSL encryption, to protect user data.
- **Testing:** The system should undergo rigorous testing to ensure it functions as intended and is free from bugs.
- **Mobile Compatibility:** The system should be able to be managed via phone apps.

CHAPTER 2 LITERATURE SURVEY

2.1 Existing System:

The existing loan and EMI management systems used by banks are often manual and paper-based. This means that loan application and approval processes are time-consuming and inefficient, and there is a higher chance of errors in data entry and processing. EMI management is also manual, with banks having to manually track customer payments and send reminders for missed payments. Reporting and analytics are often limited, making it difficult for banks to make informed decisions about their loan portfolio. Overall, the existing system is slow, inefficient, and prone to errors, resulting in poor customer experience and higher operational costs for banks.

2.2 Proposed System:

The proposed Bank Loan and EMI Management System aims to address the shortcomings of the existing system by automating loan management processes, enabling efficient EMI management, providing accurate reporting and analytics, and enhancing customer experience. The proposed system uses advanced technologies like React, Node.js, and MySQL workbench to provide a secure, user-friendly, and customizable platform for banks to manage their loan portfolio. The proposed system is scalable, allowing banks to handle an increasing number of customers, loans, and transactions. The proposed system also provides real-time assistance to customers with loan applications, EMI payments, and other queries. The proposed system also has advanced security features such as two-factor authentication, encryption, and secure payment gateways, ensuring customer data protection and fraud prevention.

Overall, the proposed Bank Loan and EMI Management System is a significant improvement over the existing manual and paper-based systems, providing efficient, secure, and automated loan management processes that improve customer experience and reduce operational costs for banks.

CHAPTER 3 SYSTEM ARCHITECTURE AND DESIGN

3.1 ARCHITECTURE DIAGRAM:

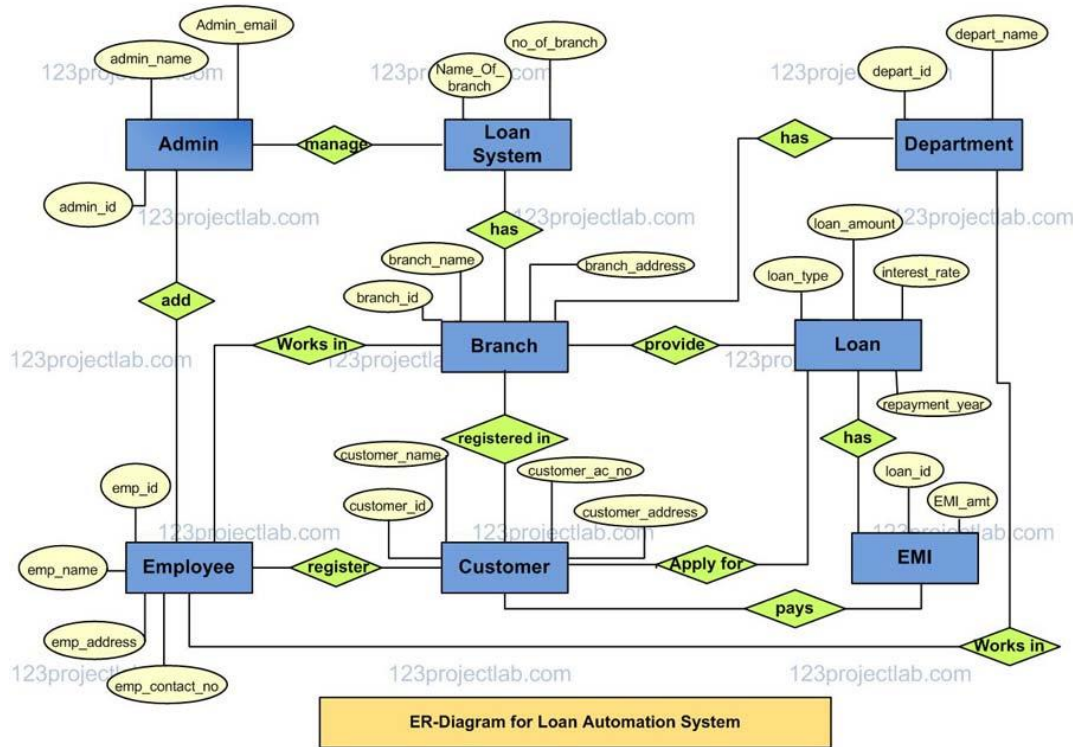


Fig 3.1 Architecture Diagram of Online Examination System

The system comprises of:

Loan Application Management System: This system is responsible for managing loan applications from customers. It includes a user interface for customers to apply for loans, and an interface for bank employees to verify customer information, perform credit checks, and approve loan applications.

EMI Payment Management System: This system is responsible for managing EMI payments from customers. It includes a user interface for customers to make EMI payments, and an interface for bank employees to track EMI payments, send reminders to customers for missed payments, and take appropriate action in case of loan defaults.

3.1.1 FRONTEND (UI)DESIGN:

Front end design is implemented using HTML,CSS and ReactJS which enables banking system interface for users interact with this UI providing a better user experience.



3.1.2 BACKEND (DATABASE)DESIGN:

MySQL Workbench is a powerful tool for designing, modeling, and managing MySQL databases. In the Bank Loan System, MySQL Workbench is used to manage the database that stores all the data related to loans, customers, and EMI payments.

MySQL Workbench allows developers to design and model the database schema, create tables, and define relationships between tables. It also provides a graphical user interface for managing data in the database, including querying data, adding, updating, and deleting records.

In the Bank Loan and EMI Management System, MySQL Workbench is used to manage the loan application process, track EMI payments, and generate reports. It is also used to ensure the security and integrity of customer data by implementing various security measures such as encryption, access control, and backup and recovery.

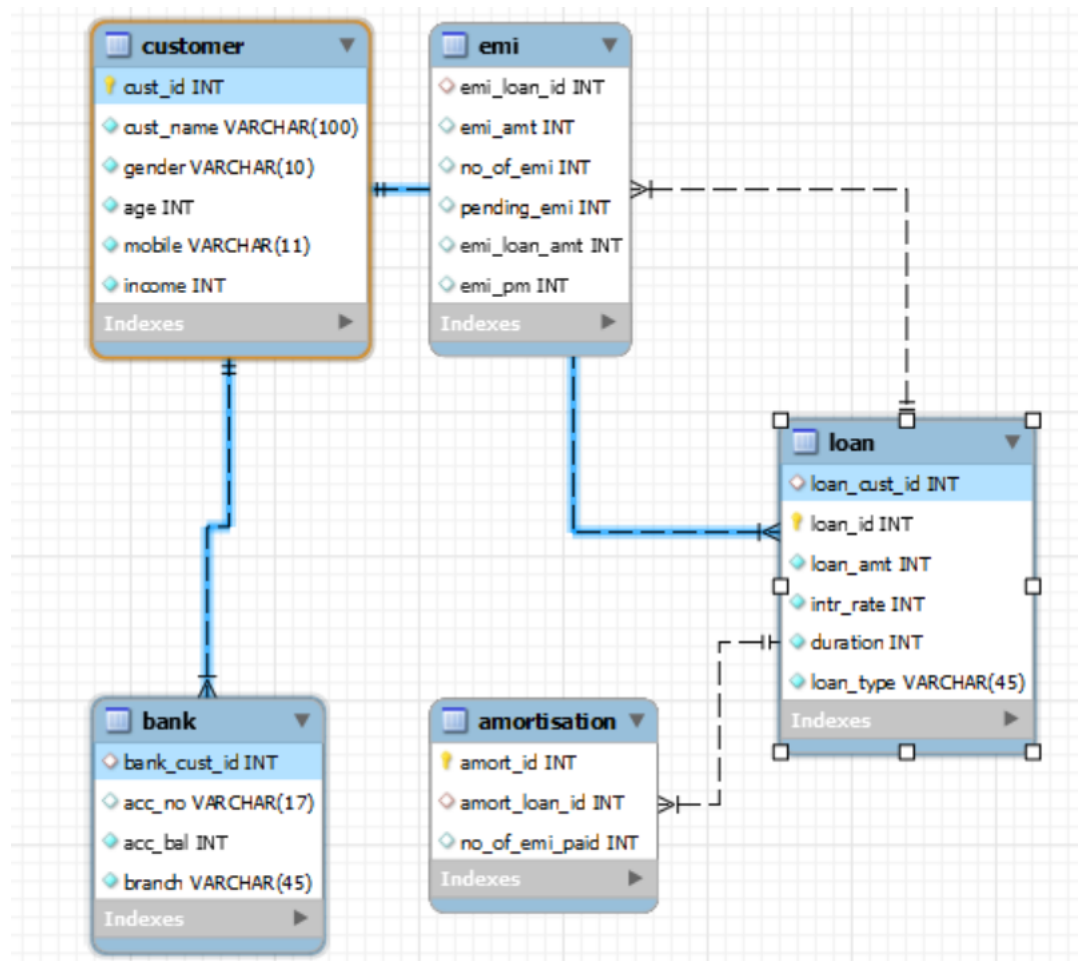


Fig 3.1.2 Backend Portal

3.2 USE CASE DIAGRAM:

Actors:

1. Customer: A person who applies for a loan and makes EMI payments.
2. Bank Employee: A person who verifies customer information, approves loans, and manages EMI payments.

Use Cases:

1. Apply for Loan: A customer can apply for a loan by providing necessary details.
2. Verify Customer Information: A bank employee can verify customer information and check for creditworthiness.
3. Approve Loan Application: A bank employee can approve or reject the loan application based on the customer's creditworthiness.
4. Make EMI Payment: A customer can make EMI payments for their loan.

5. Track EMI Payments: A bank employee can track EMI payments and send reminders to customers for missed payments.
6. Generate Reports: A bank employee can generate reports on loan disbursements, EMI payments, defaulters, and other key metrics.

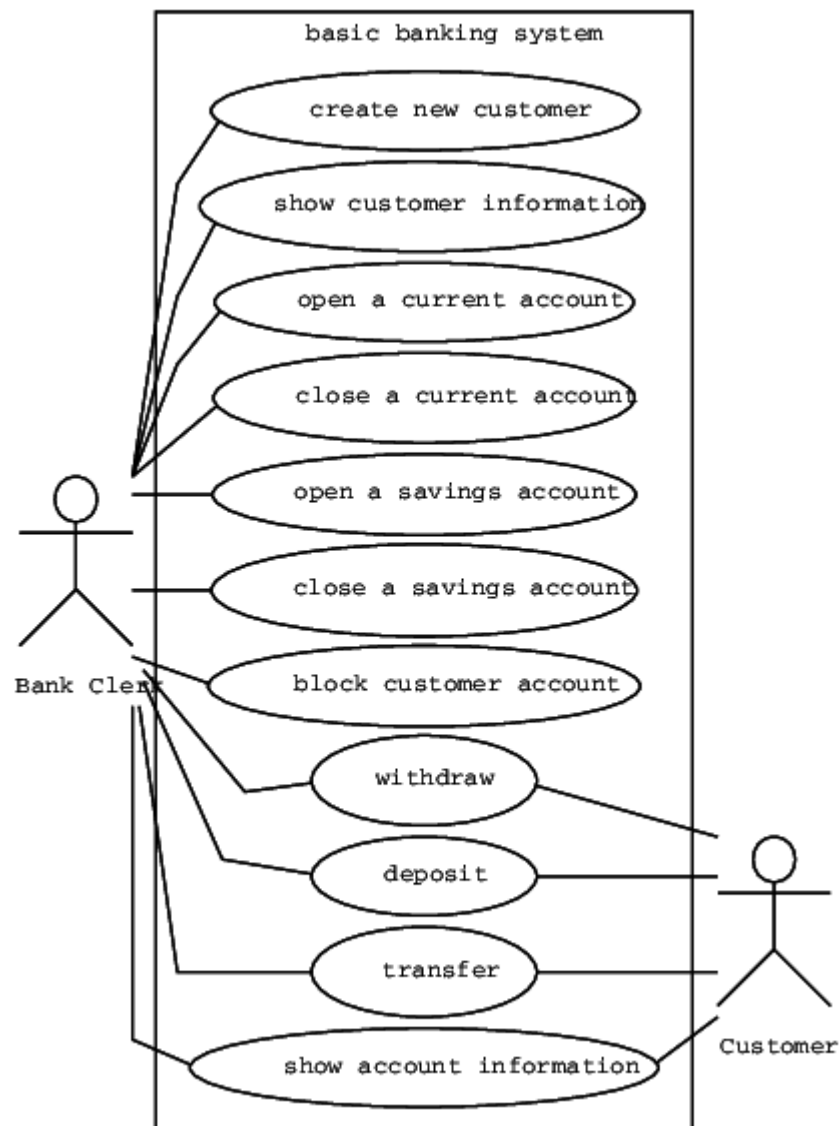


Fig 3.2 Use Case Diagram

3.3 ER DIAGRAM:

The ER diagram for our system consist of various entities which are elucidated below.

1. Customer Entity: This entity stores customer details, such as name, address, contact information, and other relevant details.
2. Loan Entity: This entity stores loan details, such as loan amount, interest rate, loan type, and other relevant details. The loan entity is related to the customer entity, where each loan is associated with a customer.
3. EMI Entity: This entity stores EMI payment details, such as EMI amount, due date, and other relevant details. The EMI entity is related to the loan entity, where each EMI payment is associated with a specific loan.
4. Payment Entity: This entity stores details about the payments made by customers, such as payment amount, date, and other relevant details. The payment entity is related to the EMI entity, where each payment is associated with a specific EMI.
5. Employee Entity: This entity stores employee details, such as name, contact information, and other relevant details.
6. Approval Entity: This entity stores loan approval details, such as approval date, employee who approved the loan, and other relevant details. The approval entity is related to the loan entity, where each loan approval is associated with a specific loan.

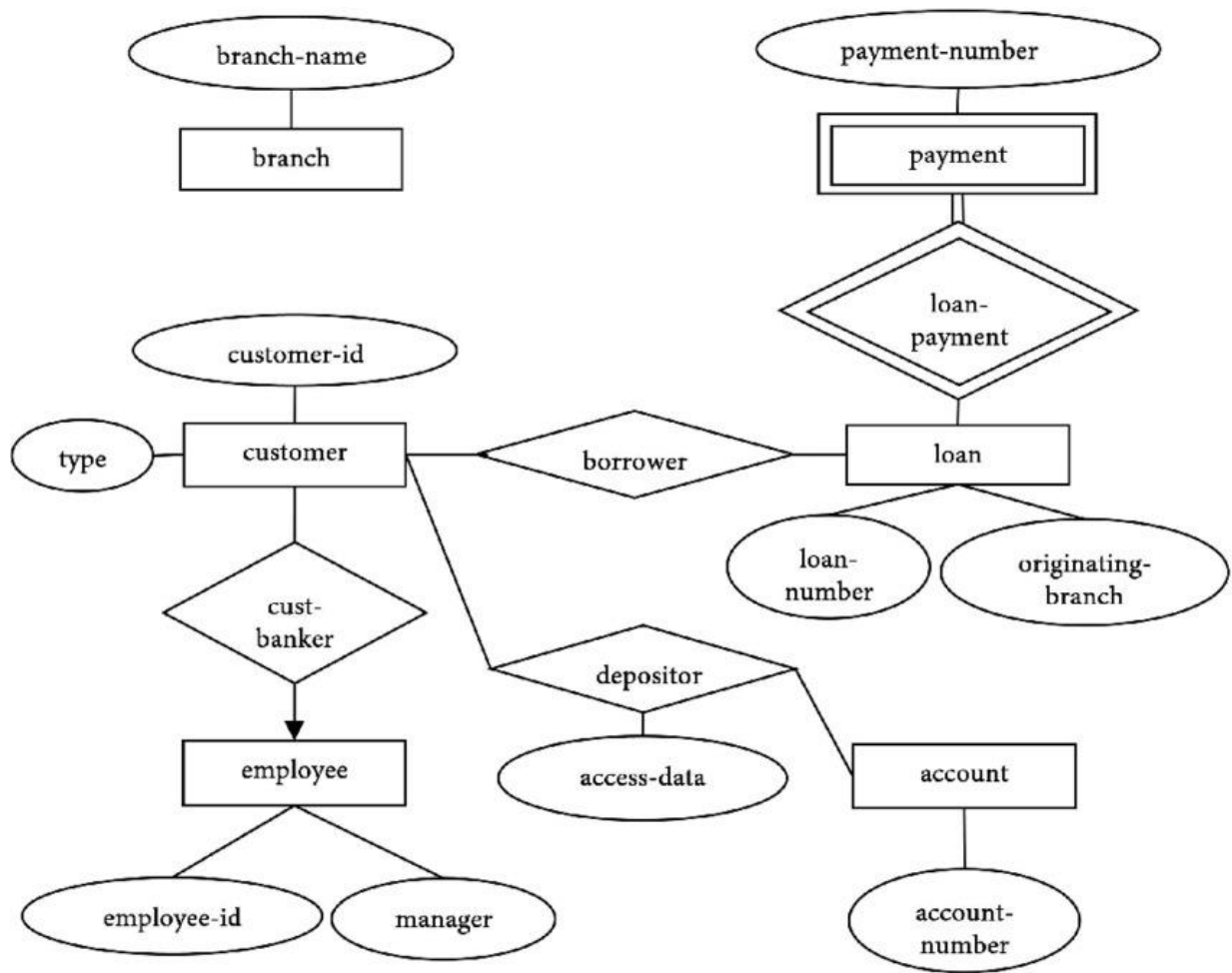


Fig 3.3 ER Diagram

CHAPTER 4 MODULES AND FUNCTIONALITIES

4.1 System Modules and its Functions:

The Bank Loan and EMI Management System consists of several modules that perform specific functionalities. Here are the modules and their functionalities:

1. Customer Management Module: This module manages customer information, including personal details, contact information, and loan application history.

Functionalities:

- Add a new customer
- Update customer information
- View customer details
- Delete customer information

2. Loan Management Module: This module manages loan applications, approvals, and disbursements.

Functionalities:

- Accept loan application
- Verify customer information
- Approve or reject loan application
- Disburse loan amount

3. EMI Payment Management Module: This module manages EMI payments and tracks missed payments.

Functionalities:

- Schedule EMI payments
- Track EMI payments
- Send reminders for missed payments
- Mark late or missed payments

4. Payment Management Module: This module manages payments made by customers.

Functionalities:

- Record customer payments
- Update payment details
- View payment history

5. Report Generation Module: This module generates reports on loan disbursements, EMI payments, defaulters, and other key metrics.

Functionalities:

- Generate reports on loan applications
- Generate reports on loan disbursements
- Generate reports on EMI payments
- Generate reports on defaulters

6. Authentication and Authorization Module: This module manages user authentication and access control.

Functionalities:

- Authenticate users
- Authorize user access based on roles and permissions
- Provide secure access to the system

These modules and functionalities work together to ensure the smooth functioning of the Bank Loan and EMI Management System. They enable the system to efficiently manage customer information, loan applications, EMI payments, and payment history while ensuring the security and integrity of customer data.

4.2 Database Connectivity

Connecting a MySQL Workbench database with Node.js and React.js using Axios.

1. Install the required dependencies:
 - For Node.js mysql workbench is used.
 - For React.js, axios module is being used.
2. Create a MySQL Workbench database and define the tables and relationships.
3. Create a Node.js file to establish a connection with the MySQL Workbench database using the mysql module.

CHAPTER 5 CODING AND TESTING

5.1 Frontend Code:

React.js: Navigation

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'
import 'bootstrap/dist/css/bootstrap.min.css';
import InpDetails from './inp_disp/insertCustomer';
import LoginForm from './login_page/loginform';
import SideBar from './dashboard/sidebar';
import DispDetails from './inp_disp/showdet.js';
import Sidebar from './dashboard/btssidebar.js';
import LoanDetails from './inp_disp/loan';
import EmiDet from './inp_disp/emi';
import UpdateEmi from './inp_disp/pendingemi';

const App = () => {
  return(
    <Router>
      <Routes>
        <Route path="/" element = {<LoginForm/>}></Route>
        <Route path="/dashboard" element = {<SideBar/>}></Route>
        <Route path="/sidebar" element = {<Sidebar/>}></Route>
        <Route path="/custDetail" element = {<DispDetails/>}></Route>
        <Route path="/addCust" element = {<InpDetails/>}></Route>
        <Route path="/addLoan" element={<LoanDetails/>}></Route>
        <Route path="/loanDetail" element={<EmiDet/>}></Route>
        <Route path="/repay" element={<UpdateEmi/>}></Route>

      </Routes>
    </Router>
  )
}

export default App;
```

Sidebar:

```
import './sidebar.css';
import React from "react";
import { sidebarData } from './sidebarData';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'
```

```

const SideBar = () => {
  return(
    <div className="dash">
      <ul className='sidebarList'>
        {sidebarData.map((val, key) => {
          return(
            <ul key={key} className='row'
onClick={()=>{window.location.pathname = val.link}}>
              <div>
                {val.title}
              </div>
            </ul>
          );
        })}
      </ul>
    </div>
  )
}

export default SideBar;

```

EMI:

```

import React from 'react'
import './customer.css';
import Axios from 'axios';
import {useState} from 'react';
import { NavLink } from 'react-router-dom';
import {
  CDBSidebar,
  CDBSidebarContent,
  CDBSidebarFooter,
  CDBSidebarHeader,
  CDBSidebarMenu,
  CDBSidebarMenuItem,
} from 'cdbreact';

const EmiDet = () => {
  const [EmiList, setEmiList] = useState([]);

  Axios.get("http://localhost:3002/getLoan").then((response) => {
    setEmiList(response.data);
  })

  return (
    <div style={{ display: 'flex', height: '100vh', overflow: 'scroll initial' }}>

```

```

<CDBSidebar textcolor="#fff" backgroundColor="#333">
  <CDBSidebarHeader prefix={<i className="fa fa-bars fa-large"></i>}>
    <a href="/" className="text-decoration-none" style={{ color: 'inherit'
}}>
      Loan Management
    </a>
  </CDBSidebarHeader>

  <CDBSidebarContent className="sidebar-content">
    <CDBSidebarMenu>
      <NavLink exact to="/sidebar" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="home">Home</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/addCust" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="user">Add Customer</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/custDetail" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="table">Customer List</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/addLoan" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="chart-line">Avail New
Loan</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/loanDetail" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="table">Loan List</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/repay" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="tree">EMI installment</CDBSidebarMenuItem>
      </NavLink>
      <NavLink exact to="/" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="columns">Logout</CDBSidebarMenuItem>
      </NavLink>

    </CDBSidebarMenu>
  </CDBSidebarContent>

  <CDBSidebarFooter style={{ textAlign: 'center' }}>

    </CDBSidebarFooter>
  </CDBSidebar>
  <div className='rightside'>
  <div className='showInfo'>
    <h3 className='custdisp'>Loan Details</h3>
    <div>
      <table class="table table-striped">
        <thead>
          <tr>
            <th scope="col">EMI ID</th>

```

```

        <th scope="col">Total EMI Amount</th>
        <th scope="col">Total EMIs</th>
        <th scope="col">Pending EMIs</th>
        <th scope="col">Loan Amount</th>
        <th scope="col">EMI per month</th>

      </tr>
    </thead>

    <tbody>
    {
      EmiList.map((val,i) =>
    <tr key={i}>

      <td>{val.emi_loan_id}</td>
      <td>{val.emi_amt}</td>
      <td>{val.no_of_emi}</td>
      <td>{val.pending_emi}</td>
      <td>{val.emi_loan_amt}</td>
      <td>{val.emi_pm}</td>

    </tr>

      )
    }

    </tbody>
  </table>
</div>
</div>
</div>
)
}

export default EmiDet;

```

Insert Customer:

```

import './customer.css'
import { useState } from 'react'
import React from 'react';
import Axios from 'axios';

import {
  CDBSidebar,
  CDBSidebarContent,

```



```

    CDBSidebarFooter,
    CDBSidebarHeader,
    CDBSidebarMenu,
    CDBSidebarMenuItem,
  } from 'cdbreact';
import { NavLink } from 'react-router-dom';

const InpDetails = () => {
  const [cust_name, setcust_name] = useState("")
  const [cust_id, setcust_id] = useState(0)
  const [gender, setgender] = useState("")
  const [age, setage] = useState(0)
  const [mobile, setmobile] = useState("")
  const [income, setincome] = useState(0)
  const [acc_no, setacc_no] = useState("")
  const [acc_bal, setacc_bal] = useState(0)
  const [branch, setbranch] = useState("")

  const addCust =() => {
    Axios.post("http://localhost:3002/create", {
      cust_name : cust_name,
      cust_id : cust_id,
      gender: gender,
      age : age,
      mobile : mobile,
      income : income,
      acc_no : acc_no,
      acc_bal : acc_bal,
      branch : branch,

    }).then(() => {
      console.log("success");
    })
  }
  return (
    <div style={{ display: 'flex', height: '100vh', overflow: 'scroll initial' }}>
      <CDBSidebar textColor="#fff" backgroundColor="#333">
        <CDBSidebarHeader prefix={<i className="fa fa-bars fa-large"></i>}>
          <a href="/" className="text-decoration-none" style={{ color: 'inherit'
}}>
            Loan Management
          </a>
        </CDBSidebarHeader>

        <CDBSidebarContent className="sidebar-content">
          <CDBSidebarMenu>

```

```

    <NavLink exact to="/sidebar" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="home">Home</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/addCust" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="user">Add Customer</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/custDetail" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="table">Customer List</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/addLoan" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="chart-line">Avail New
Loan</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/loanDetail" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="table">Loan List</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/repay" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="tree">EMI installment</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/" activeClassName="activeClicked">
      <CDBSidebarMenuItem icon="columns">Logout</CDBSidebarMenuItem>
    </NavLink>

  </CDBSidebarMenu>
</CDBSidebarContent>

<CDBSidebarFooter style={{ textAlign: 'center' }}>

</CDBSidebarFooter>
</CDBSidebar>
<div className='rightside'>

<div className="details">
  <div className="information">
    <h3>Add Customers</h3>
    <label>Name:</label>
    < input type = 'text' onChange = {(event) =>
      {setcust_name(event.target.value);}
    } />
    <label>Customer ID:</label>
    <input type = 'number' onChange = {(event) =>
      {setcust_id(event.target.value);}
    } />
    <label>Gender:</label>
    <input type = 'text' onChange = {(event) =>
      {setgender(event.target.value);}
    } />
    <label>Age:</label>

```

```

        <input type = 'number' onChange = {(event) =>
            {setage(event.target.value);}}
        />
        <label>Mobile:</label>
        <input type = 'text' onChange = {(event) =>
            {setmobile(event.target.value);}}
        />
        <label>Income:</label>
        <input type = 'number' onChange = {(event) =>
            {setincome(event.target.value);}}
        />
        <label>Account Number:</label>
        <input type = 'text' onChange = {(event) =>
            {setacc_no(event.target.value);}}
        />
        <label>Account Balance:</label>
        <input type = 'number' onChange = {(event) =>
            {setacc_bal(event.target.value);}}
        />
        <label>Branch:</label>
        <input type = 'text' onChange = {(event) =>
            {setbranch(event.target.value);}}
        />
        <button onClick={addCust}>Add Customer</button>
    </div>

</div>

</div>

</div>
);
};

export default InpDetails;

```

Loan:

```

import './customer.css'
import { useState } from 'react'
import React from 'react';
import Axios from 'axios';

import {
    CDBSidebar,
    CDBSidebarContent,

```

```

    CDBSidebarFooter,
    CDBSidebarHeader,
    CDBSidebarMenu,
    CDBSidebarMenuItem,
  } from 'cdbreact';
import { NavLink } from 'react-router-dom';

const LoanDetails = () => {
  const [cust_id, setcust_id] = useState(0)
  const [loan_id, setloan_id] = useState(0)
  const [loan_amt, setloan_amt] = useState(0)
  const [int_rate, setint_rate] = useState(0)
  const [duration, setduration] = useState(0)
  const [loan_type, setloan_type] = useState("")

  const addLoan =() => {
    Axios.post("http://localhost:3002/addloan", {
      cust_id : cust_id,
      loan_id : loan_id,
      loan_amt : loan_amt,
      int_rate : int_rate,
      duration : duration,
      loan_type : loan_type,

    }).then(() => {
      console.log("success");
    })
  }
  return (
    <div style={{ display: 'flex', height: '100vh', overflow: 'scroll initial' }}>
      <CDBSidebar textColor="#fff" backgroundColor="#333">
        <CDBSidebarHeader prefix={<i className="fa fa-bars fa-large"></i>}>
          <a href="/" className="text-decoration-none" style={{ color: 'inherit'
}}>
            Loan Management
          </a>
        </CDBSidebarHeader>

        <CDBSidebarContent className="sidebar-content">
          <CDBSidebarMenu>
            <NavLink exact to="/sidebar" activeClassName="activeClicked">
              <CDBSidebarMenuItem icon="home">Home</CDBSidebarMenuItem>
            </NavLink>
            <NavLink exact to="/addCust" activeClassName="activeClicked">
              <CDBSidebarMenuItem icon="user">Add Customer</CDBSidebarMenuItem>
            </NavLink>

```

```

        <NavLink exact to="/custDetail" activeClassName="activeClicked">
          <CDBSidebarMenuItem icon="table">Customer List</CDBSidebarMenuItem>
        </NavLink>
        <NavLink exact to="/addLoan" activeClassName="activeClicked">
          <CDBSidebarMenuItem icon="chart-line">Avail New
Loan</CDBSidebarMenuItem>
        </NavLink>
        <NavLink exact to="/loanDetail" activeClassName="activeClicked">
          <CDBSidebarMenuItem icon="table">Loan List</CDBSidebarMenuItem>
        </NavLink>
        <NavLink exact to="/repay" activeClassName="activeClicked">
          <CDBSidebarMenuItem icon="tree">EMI installment</CDBSidebarMenuItem>
        </NavLink>
        <NavLink exact to="/" activeClassName="activeClicked">
          <CDBSidebarMenuItem icon="columns">Logout</CDBSidebarMenuItem>
        </NavLink>

      </CDBSidebarMenu>
    </CDBSidebarContent>

    <CDBSidebarFooter style={{ textAlign: 'center' }}>

    </CDBSidebarFooter>
  </CDBSidebar>
  <div className='rightside'>

  <div className="details">
    <div className="information">
      <h3>Add Loan</h3>

      <label>Customer ID:</label>
      <input type = 'number' onChange = {(event) =>
        {setcust_id(event.target.value);}
      }/>
      <label>Loan ID:</label>
      <input type = 'number' onChange = {(event) =>
        {setloan_id(event.target.value);}
      }/>
      <label>Loan Amount:</label>
      <input type = 'number' onChange = {(event) =>
        {setloan_amt(event.target.value);}
      }/>
      <label>Interest Rate:</label>
      <input type = 'number' onChange = {(event) =>
        {setint_rate(event.target.value);}
      }/>
      <label>Duration(in months):</label>
      <input type = 'number' onChange = {(event) =>

```

```

        {setduration(event.target.value);}
      }/>
      <label>Loan Type:</label>
      <input type = 'text' onChange = {(event) =>
        {setloan_type(event.target.value);}}
      />

      <button onClick={addLoan}>Avail Loan</button>
    </div>

  </div>

</div>
);
};

export default LoanDetails;

```

Pending EMI:

```

import './customer.css'
import { useState } from 'react'
import React from 'react';
import Axios from 'axios';

import {
  CDBSidebar,
  CDBSidebarContent,
  CDBSidebarFooter,
  CDBSidebarHeader,
  CDBSidebarMenu,
  CDBSidebarMenuItem,
} from 'cdbreact';
import { NavLink } from 'react-router-dom';

const UpdateEmi = () => {
  const [amort_id, setamort_id] = useState(0)
  const [amort_loan_id, setamort_loan_id] = useState(0)
  const [no_of_emi_paid, setno_of_emi_paid] = useState(0)

```

```

const addInstallment =() => {
  Axios.post("http://localhost:3002/amort", {
    amort_id : amort_id,
    amort_loan_id : amort_loan_id,
    no_of_emi_paid : no_of_emi_paid,

  }).then(() => {
    console.log("success");
  })
}

}
return (
  <div style={{ display: 'flex', height: '100vh', overflow: 'scroll initial' }}>
    <CDBSidebar textColor="#fff" backgroundColor="#333">
      <CDBSidebarHeader prefix={<i className="fa fa-bars fa-large"></i>}>
        <a href="/" className="text-decoration-none" style={{ color: 'inherit'
}}>
          Loan Management
        </a>
      </CDBSidebarHeader>

      <CDBSidebarContent className="sidebar-content">
        <CDBSidebarMenu>
          <NavLink exact to="/sidebar" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="home">Home</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/addCust" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="user">Add Customer</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/custDetail" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="table">Customer List</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/addLoan" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="chart-line">Avail New
Loan</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/loanDetail" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="table">Loan List</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/repay" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="tree">EMI installment</CDBSidebarMenuItem>
          </NavLink>
          <NavLink exact to="/" activeClassName="activeClicked">
            <CDBSidebarMenuItem icon="columns">Logout</CDBSidebarMenuItem>
          </NavLink>

```

```

        </CDBSidebarMenu>
    </CDBSidebarContent>

    <CDBSidebarFooter style={{ textAlign: 'center' }}>

    </CDBSidebarFooter>
</CDBSidebar>
<div className='rightside'>

<div className="details">
    <div className="information">
        <h3>Add Loan</h3>

        <label>EMI ID:</label>
        <input type = 'number' onChange = {(event) =>
            {setamort_id(event.target.value);}
        }/>
        <label>Loan ID:</label>
        <input type = 'number' onChange = {(event) =>
            {setamort_loan_id(event.target.value);}
        }/>
        <label>Number of installments:</label>
        <input type = 'number' onChange = {(event) =>
            {setno_of_emi_paid(event.target.value);}
        }/>

        <button onClick={addInstallment}>Update Installment</button>
    </div>

</div>

</div>

</div>
);
};

export default UpdateEmi;

```

Show Detail:

```

import './customer.css'
import { useState } from 'react'
import React from 'react';
import Axios from 'axios';

```



```

import {
  CDBSidebar,
  CDBSidebarContent,
  CDBSidebarFooter,
  CDBSidebarHeader,
  CDBSidebarMenu,
  CDBSidebarMenuItem,
} from 'cdbreact';
import { NavLink } from 'react-router-dom';

const DispDetails = () => {

  const [custList, setcustList] = useState([])

  //const disp = () => {
  //  //console.log(name+amt+aadhar+pan+intr+crd+apprv)
  //}

  Axios.get("http://localhost:3002/getCust").then((response) => {
    setcustList(response.data);
  })

  return (
    <div style={{ display: 'flex', height: '100vh', overflow: 'scroll initial' }}>
      <CDBSidebar textColor="#fff" backgroundColor="#333">
        <CDBSidebarHeader prefix={<i className="fa fa-bars fa-large"></i>}>
          <a href="/" className="text-decoration-none" style={{ color: 'inherit'
}}>
            Loan Management
          </a>
        </CDBSidebarHeader>

        <CDBSidebarContent className="sidebar-content">
          <CDBSidebarMenu>
            <NavLink exact to="/sidebar" activeClassName="activeClicked">
              <CDBSidebarMenuItem icon="home">Home</CDBSidebarMenuItem>
            </NavLink>
            <NavLink exact to="/addCust" activeClassName="activeClicked">
              <CDBSidebarMenuItem icon="user">Add Customer</CDBSidebarMenuItem>
            </NavLink>
            <NavLink exact to="/custDetail" activeClassName="activeClicked">
              <CDBSidebarMenuItem icon="table">Customer List</CDBSidebarMenuItem>
            </NavLink>
            <NavLink exact to="/addLoan" activeClassName="activeClicked">

```

```

        <CDBSidebarMenuItem icon="chart-line">Avail New
Loan</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/loanDetail" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="table">Loan List</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/repay" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="tree">EMI installment</CDBSidebarMenuItem>
    </NavLink>
    <NavLink exact to="/" activeClassName="activeClicked">
        <CDBSidebarMenuItem icon="columns">Logout</CDBSidebarMenuItem>
    </NavLink>

</CDBSidebarMenu>
</CDBSidebarContent>

<CDBSidebarFooter style={{ textAlign: 'center' }}>

</CDBSidebarFooter>
</CDBSidebar>
<div className='rightside'>
<div className='showInfo'>
    <h3 className='custdisp'>Customer Details</h3>
    <div>
        <table class="table table-striped">
        <thead>
            <tr>
                <th scope="col">Customer ID</th>
                <th scope="col">Customer Name</th>
                <th scope="col">Gender</th>
                <th scope="col">Age</th>
                <th scope="col">Mobile</th>
                <th scope="col">Income</th>
                <th scope="col">Account Number</th>
                <th scope="col">Account Balance</th>
                <th scope="col">Branch</th>
            </tr>
        </thead>

        <tbody>
        {
            custList.map((val,i) =>
        <tr key={i}>

                <td>{val.cust_id}</td>
                <td>{val.cust_name}</td>
                <td>{val.gender}</td>

```

```

        <td>{val.age}</td>
        <td>{val.mobile}</td>
        <td>{val.income}</td>
        <td>{val.acc_no}</td>
        <td>{val.acc_bal}</td>
        <td>{val.branch}</td>

    </tr>

    )
}

</tbody>
</table>
</div>
</div>
</div>
</div>
);
};

export default DispDetails;

```

App.js

```

import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'
import 'bootstrap/dist/css/bootstrap.min.css';
import InpDetails from './inp_disp/insertCustomer';
import LoginForm from './login_page/loginform';
import SideBar from './dashboard/sidebar';
import DispDetails from './inp_disp/showdet.js';
import Sidebar from './dashboard/btssidebar.js';
import LoanDetails from './inp_disp/loan';
import EmiDet from './inp_disp/emi';
import UpdateEmi from './inp_disp/pendingemi';

const App = () => {
  return(
    <Router>
      <Routes>
        <Route path="/" element = {<LoginForm/>}></Route>
        <Route path="/dashboard" element = {<SideBar/>}></Route>
        <Route path="/sidebar" element = {<Sidebar/>}></Route>
        <Route path="/custDetail" element = {<DispDetails/>}></Route>

```

```

    <Route path="/addCust" element = {<InpDetails/>}></Route>
    <Route path="/addLoan" element={<LoanDetails/>}></Route>
    <Route path="/loanDetail" element={<EmiDet/>}></Route>
    <Route path="/repay" element={<UpdateEmi/>}></Route>

  </Routes>
</Router>
)
}

export default App;

```

index.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();

```

5.2 Login UI:



The image shows the 'Add Customers' form within the same application. The sidebar is identical to the previous screen. The main content area has a light orange background and is titled 'Add Customers'. It contains a series of input fields for customer information: Name, Customer ID (with a dropdown arrow), Gender, Age (with a dropdown arrow), Mobile, Income (with a dropdown arrow), Account Number, Account Balance (with a dropdown arrow), and Branch. At the bottom right of the form is a button labeled 'Add Customer'.

Loan Management	Customer Details								
	Customer ID	Customer Name	Gender	Age	Mobile	Income	Account Number	Account Balance	Branch
Home	1	Sindhu	Female	20	1234567890	1200000	2334	100	busan
Add Customer	5	Yoongi	Male	29	9876543210	90000	1234567890123456	3000	Busan
Customer List	6	Yoongi	Male	29	9876543210	90000	1234567890123456	3000	Busan
Avail New Loan	7	Yoongi	Male	29	9876543210	90000	1234567890123456	3000	Busan
Loan List	10	Mark	Male	34	762386	121233	314234124	1200	California
EMI installment	234	ES	DSF	34	45145	345	452353	233	SFDGDF
Logout	11	LOL	SADV	23	31424	44	23442534	1234	SFFS
	21	asd	f	35	42345	1345	15414	134	dsf
	9	hi	f	56	2435523	435	1435	145	chennai

Loan Management	Add Loan
Home	Customer ID:
Add Customer	41
Customer List	Loan ID:
Avail New Loan	23
Loan List	Loan Amount:
EMI installment	5000
Logout	Interest Rate:
	3
	Duration(in months):
	12
	Loan Type:
	Car
	Avail Loan

Loan Management	Loan Details					
	EMI ID	Total EMI Amount	Total EMIs	Pending EMIs	Loan Amount	EMI per month
Home	7	1000	6	6	500	167
Add Customer	3	1000	5	3	4000	200
Customer List	1	2310	7	7	11000	330
Avail New Loan	9	72000	6	6	2300	12000
Loan List	91	69	3	3	2300	23
EMI installment	95	69	3	3	2300	23
Logout	1	2310	7	7	11000	330
	2	2310	7	7	11000	330
	2	2310	7	7	11000	330
	11	2310	7	7	11000	330

Loan Management

Home

Add Customer

Customer List

Avail New Loan

Loan List

EMI installment

Logout

Add Loan

EMI ID:

2


Loan ID:

23

Number of installments:

5

Update Installment



5.3 Database:

Amortisation:

Navigator

SCHEMAS

Filter objects

blms

Tables

amortisation

bank

customer

emi

loan

Views

Stored Procedures

Functions

sys

Administration Schemas

Query 1 emi amortisation amortisation blms - Schema blms.amortisation

Info Columns Indexes Triggers Foreign keys Partitions Grants DDL

Indexes in Table

Visible	Key	Type	Uni...	Columns
<input checked="" type="checkbox"/>	PRIMARY	BTREE	YES	amort_id
<input checked="" type="checkbox"/>	amort_loan_id_idx	BTREE	NO	amort_loan_id

Index Details

Drop Index

Key Name:

Index Type:

Allows NULL:

Cardinality:

Comment:

User Comment:

Packed:

Unique:

Columns in table

Column	Type	Nullable	Indexes
amort_id	int	NO	PRIMARY
amort_loan_id	int	YES	amort_loan_id_idx
no_of_emi_paid	int	YES	

Bank:

Navigator

SCHEMAS

Filter objects

blms

Tables

amortisation

bank

customer

emi

loan

Views

Stored Procedures

Functions

sys

Administration Schemas

Query 1 emi amortisation amortisation blms - Schema blms.bank

Info Columns Indexes Triggers Foreign keys Partitions Grants DDL

Indexes in Table

Visible	Key	Type	Uni...	Columns
<input checked="" type="checkbox"/>	bank_cust_id_idx	BTREE	NO	bank_cust_id

Index Details

Drop Index

Key Name:

Index Type:

Allows NULL:

Cardinality:

Comment:

User Comment:

Packed:

Unique:

Columns in table

Column	Type	Nullable	Indexes
bank_cust_id	int	YES	bank_cust_id_idx
acc_no	varchar(17)	YES	
acc_bal	int	NO	
branch	varchar(45)	NO	

Customer:

Indexes in Table

Visible	Key	Type	Uni...	Columns
<input checked="" type="checkbox"/>	PRIMARY	BTREE	YES	cust_id

Index Details

Key Name:
Index Type:
Allows NULL:
Cardinality:
Comment:
User Comment:

Packed:
Unique:

Columns in table

Column	Type	Nullable	Indexes
cust_id	int	NO	PRIMARY
cust_name	varchar(100)	NO	
gender	varchar(10)	NO	
age	int	NO	
mobile	varchar(11)	NO	
income	int	NO	

EMI:

Indexes in Table

Visible	Key	Type	Uni...	Columns
<input checked="" type="checkbox"/>	emi_loan_id_idx	BTREE	NO	emi_loan_id

Index Details

Key Name:
Index Type:
Allows NULL:
Cardinality:
Comment:
User Comment:

Packed:
Unique:

Columns in table

Column	Type	Nullable	Indexes
emi_loan_id	int	YES	emi_loan_id_idx
emi_amt	int	YES	
no_of_emi	int	YES	
pending_emi	int	YES	
emi_loan_amt	int	YES	
emi_pm	int	YES	

Loan:

Indexes in Table

Visible	Key	Type	Uni...	Columns
<input checked="" type="checkbox"/>	PRIMARY	BTREE	YES	loan_id
<input checked="" type="checkbox"/>	cust_id_idx	BTREE	NO	loan_cust_id

Index Details

Key Name:
Index Type:
Allows NULL:
Cardinality:
Comment:
User Comment:

Packed:
Unique:

Columns in table

Column	Type	Nullable	Indexes
loan_cust_id	int	YES	cust_id_idx
loan_id	int	NO	PRIMARY
loan_amt	int	NO	
intr_rate	int	NO	
duration	int	NO	
loan_type	varchar(45)	NO	

Fig 5.3 Database of the Portal

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 Results:

The results of a system can be evaluated based on its performance, efficiency, accuracy, user-friendliness, security, and other factors.

To measure the performance of the system, you can evaluate how well it handles loan requests, calculates loan amounts, interest rates, and EMI payments accurately, and provides timely responses to the users. You can also measure the system's efficiency by assessing its response time, processing speed, and ability to handle a large number of loan requests simultaneously.

The accuracy of the system can be evaluated by comparing its loan calculations with industry standards and verifying the results with actual loan payments. User-friendliness can be assessed by conducting user surveys, usability tests, and analyzing user feedback. Security can be evaluated by conducting penetration tests, vulnerability scans, and compliance audits.

The success of the bank loan and emi management system depends on how well it meets the needs of the users, provides accurate and timely loan calculations, and maintains a high level of security and privacy.

6.2 Benefits:

1. Streamlined loan processing: The system automates the loan processing workflow, reducing the time and effort required to process loan applications.
2. Increased efficiency: The system is designed to handle a large number of loan requests simultaneously, enabling the bank to process more loans in less time.
3. Improved accuracy: The system uses advanced algorithms to calculate loan amounts, interest rates, and EMI payments, ensuring accurate and error-free loan processing.
4. Enhanced customer experience: The system provides a user-friendly interface that allows customers to apply for loans, track their loan applications, and receive timely updates.
5. Faster loan disbursement: With the automated loan processing workflow, the system can disburse loans quickly, improving customer satisfaction.
6. Reduced costs: The system eliminates the need for manual loan processing, reducing the costs associated with loan processing and administration.
7. Better risk management: The system can assess the creditworthiness of loan applicants, reducing the risk of defaults and improving the bank's overall loan portfolio.
8. Real-time reporting: The system provides real-time reporting on loan applications, disbursements, and payments, enabling the bank to monitor loan performance and make informed decisions.

CHAPTER 7 CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion:

In conclusion, the bank loan and EMI management system is a valuable tool for banks and financial institutions to streamline their loan processing workflow, increase efficiency, improve accuracy, and enhance the overall customer experience. The system uses advanced algorithms to calculate loan amounts, interest rates, and EMI payments, reducing the risk of errors and enabling faster loan disbursement.

With a user-friendly interface, the system allows customers to apply for loans, track their loan applications, and receive timely updates, improving customer satisfaction. The system also provides real-time reporting on loan applications, disbursements, and payments, enabling the bank to monitor loan performance and make informed decisions.

7.2 Future Enhancements:

1. Integration with credit scoring systems: The system could be integrated with credit scoring systems to automatically assess the creditworthiness of loan applicants and determine the risk of default.
2. Chatbot integration: The system could be integrated with a chatbot to provide customers with immediate assistance and answers to their queries.
3. Mobile app development: The system could be extended to include a mobile application for customers to apply for loans, track their applications, and make loan payments on-the-go.
4. Loan recommendation engine: The system could include a loan recommendation engine that analyzes customer data and provides personalized loan recommendations based on their financial profile.
5. Social media integration: The system could be integrated with social media platforms to enable customers to apply for loans using their social media profiles and to receive loan updates through social media messaging.
6. Fraud detection and prevention: The system could be enhanced with advanced fraud detection and prevention mechanisms to protect the bank and its customers from fraudulent loan activities.
7. Machine learning and artificial intelligence: The system could be improved with machine learning and artificial intelligence algorithms to automatically optimize loan processing and improve accuracy in loan calculations.

REFERENCES

1. <https://dev.mysql.com/doc/workbench/en/wb-develop-object-management-inspector.html>
2. <https://stackoverflow.com>
3. <https://www.mysql.com/>
4. <https://www.mysqltutorial.org/mysql-nodejs/connect/>
5. Arunkumar, R., Kotreshwar, G: "Risk Management in Commercial Banks".
6. Ciuriak, D: "Applying the Best Banking Standards and Standards" - Lessons From the Past ".
7. Saunders A., and Walter, I.: "Financial Architecture Systemic Risk And Universal Banking".