

LOAN MANAGEMENT SYSTEM

A CAPSTONE PROJECT REPORT

*Submitted in partial fulfillment of the
requirement for the award of the
Degree of*

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

by

**Mupparapu Sumana (19BCE7316)
Kookutla Satwika (19BCE7585)
Karri Roshini Sri Vyshna Pranathi (19BCE7656)**

Under the Guidance of

Prof. Ajith Jublison



**SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237**

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled “**LOAN MANAGEMENT SYSTEM**” that is being submitted by **Mupparapu Sumana (19BCE7316)** , **Karri Roshini Sri Vyshna Pranathi (19BCE7656)** , **Kookutla Satwika (19BCE7585)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Prof. AJITH JUBLISON

Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner

External Examiner

Approved by

PROGRAM CHAIR

B. Tech. CSE

Computer Science Engineering

DEAN

School Of

ACKNOWLEDGEMENTS

We are very thankful to **Prof. AJITH JUBLISON**, School of computer Science and engineering, Vellore Institute of Technology, A.P.(Amaravati), for his constant encouragement, insightful comments, helpful information and wholehearted support which always helped us tremendously in completing my project. We came to know about so many things we are thankful to **Prof. AJITH JUBLISON**. The advice shared throughout the period helped us to understand the system better and kept us well-focused.

We expand our thanks to all the faculty working in our university for their great effort with timely encouragement.

We are also thankful to each of us as we are there for each other in hard times. We motivated each other to fulfill the project before the timeline. Some courses like WEB DEVELOPMENT AND JAVA helped us a lot in creating website frontend and backend. Finally, we want to thank every individual who took part to success the project.

ABSTRACT

In this paper an efficient system is presented which can automatically manage complete servicing process and at the same time monitor the changes and operations done on the Loan Management System. Loan Management System is a web-based platform. It's a single page web application which includes header and navigation bar. The scope of this project is to provide good communication between Admin and Customer. The current system is a user-friendly system, can easily track information such as customer data and all loan details. This system is designed to perform the functions of back offices of a bank and offers any sort of loan. This system moves in a systematic way, obtaining loan details processes, settlement processes and approving the payment process. This project provides an accurate and convenient way to handle loan approval. Loan Management System is very helpful for Banking Staff's, It provides a reliable and convenient platform for managing loans. All web forms are understandable and easy to handle by the bank staff as front-end is done by React-JS. Implemented machine learning model to predict the approval or rejection of a loan with some given details which helps in giving an idea about the outcome before applying for a loan. The Loan Processing System uses SQL Server for data storage which has robust stability for storing all the huge transaction details in efficient and an error free manner. "Loan Management System" provides facility to store the information of new customer, different types of loans provided by the bank, interest rates of different types on loan etc., keeps proper record of all loans given by the bank and their current status. Managing data security of the confidential data.

TABLE OF CONTENTS

S.No.	Chapter	Title	Page Number
1.		Acknowledgements	2
2.		Abstract	3
3.	1	Introduction	5
	1.1	Objectives	5
	1.2	Background and Literature Survey	5-7
	1.3	Organization of the Report	7
	1.4	Novelty of the Project	7
4.	2	Loan Management System Proposed System	7 8
	2.1	Working Methodology	8-9
	2.2	System Details Software Details	 9-12
5.	3	Results and Discussion	12-17
6.	4	Conclusion & Future Works	18
7.	5	Appendix	18-59
8.	6	References	59-60

INTRODUCTION

These days loan financing is very much growing sector that involves lots of paperwork due to this customer get very difficult to loan. This project will help company to automate loan approval process where anyone can easily apply for loan and can get approval with in five minutes. The main objective of finance system is to provide complete and easy loan solution for customer so that any apply loan easily from anywhere and anytime as per their requirement.

1.1 Objectives

The following are the objectives of this project:

- To design an efficient system which can automatically manage complete servicing process and at the same time monitor the changes and operations done on the account created by the customer.
- The main objective is to provide good interaction and communication facilities between customers and administration.
- Loan Management System has been designed to online the back-office activities of bank and finance company which offers any type of loan.
- Customer can track their details of loan progress on the website from online.

1.2 Background and Literature Survey

A similar project on the ground of Advance Java and Web Technologies and as well as the Machine Learning Algorithms. The Existing System is a manual that doesn't maintain details with proper security and can't track details easily. The Projects we see in this Loan Issue Systems it basically doesn't allow the customer to check their profile in proper way which leads customer dissatisfaction.

Many of them doesn't contains functionalities of fast retrieval information such details and maintenance of all the loan details so it involves lots of paperwork. As it doesn't give customer a user-friendly interface. Even if we need this to be done lots of time is required to manage customer information and details so it feels that existing loan system not accurate and therefore maintenance becomes very complicate. Difficulty in generating different reports as per the business requirement.

A Loan Management System is a digital platform that helps automate every stage of the loan lifecycle, from application or website to closing. The Traditional Loan Management Process is meticulous, time-consuming and requires collecting and verifying information about applicants, their trustworthiness and their credibility.

The Literature Survey of our Loan Management System helps the customers to apply for a loan and after approved it they can track their details and the loan status, and in which stage the loan is present from online.

By using this Loan Management System Admin can find customer easily and it's a paperless system. So, the system reduces the workload. The decision process becomes faster and more consistent after registration and login customer can use the system easily. In single page application customer can apply for a loan and after approved it they can track their details from online.

When the customer opens the Loan Management System the customer will register or login in the website and the authentication will be processed, after that based on the credentials entered the algorithm, will differentiate between the customer and the admin.

If it is Admin then system shows applying of loan page, loan status page whether it should be approved or to be rejected. In the default status the status is pending. Admin can manage loan amount, purpose of loan like the description of that loan, loan type like Educational Loan, Home Loan, Car Loan, Loan Period is preferrable in months, Loan Interest and then the logout.

If it is Customer then system shows to view loan amount, purpose of loan, loan type, loan period, loan Interest and view loan details. The Dashboard is the main part of this loan management system which will show customer to change passwords, to see profile and can make modifications and logout.

The main technology here in this system is the customer will get the status approved or rejected based on the details provided by the customer for applying loan by that details algorithm will predict that the accuracy whether the customer will pay or not. Then status will be updated.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology and software details.
- Chapter 3 discusses the results obtained after the project was implemented.

- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 gives references.

1.4 Novelty of the Project

In our system we are implementing an AI and ML model to predict his loan will be accepted or rejected before applying. Because if he applies his credit score will be affected.

CHAPTER 2

LOAN MANAGEMENT SYSTEM

This Chapter describes the proposed system, working methodology, software details.

2.1 Proposed System

The following flow diagram (figure 1) shows the system architecture of this project.

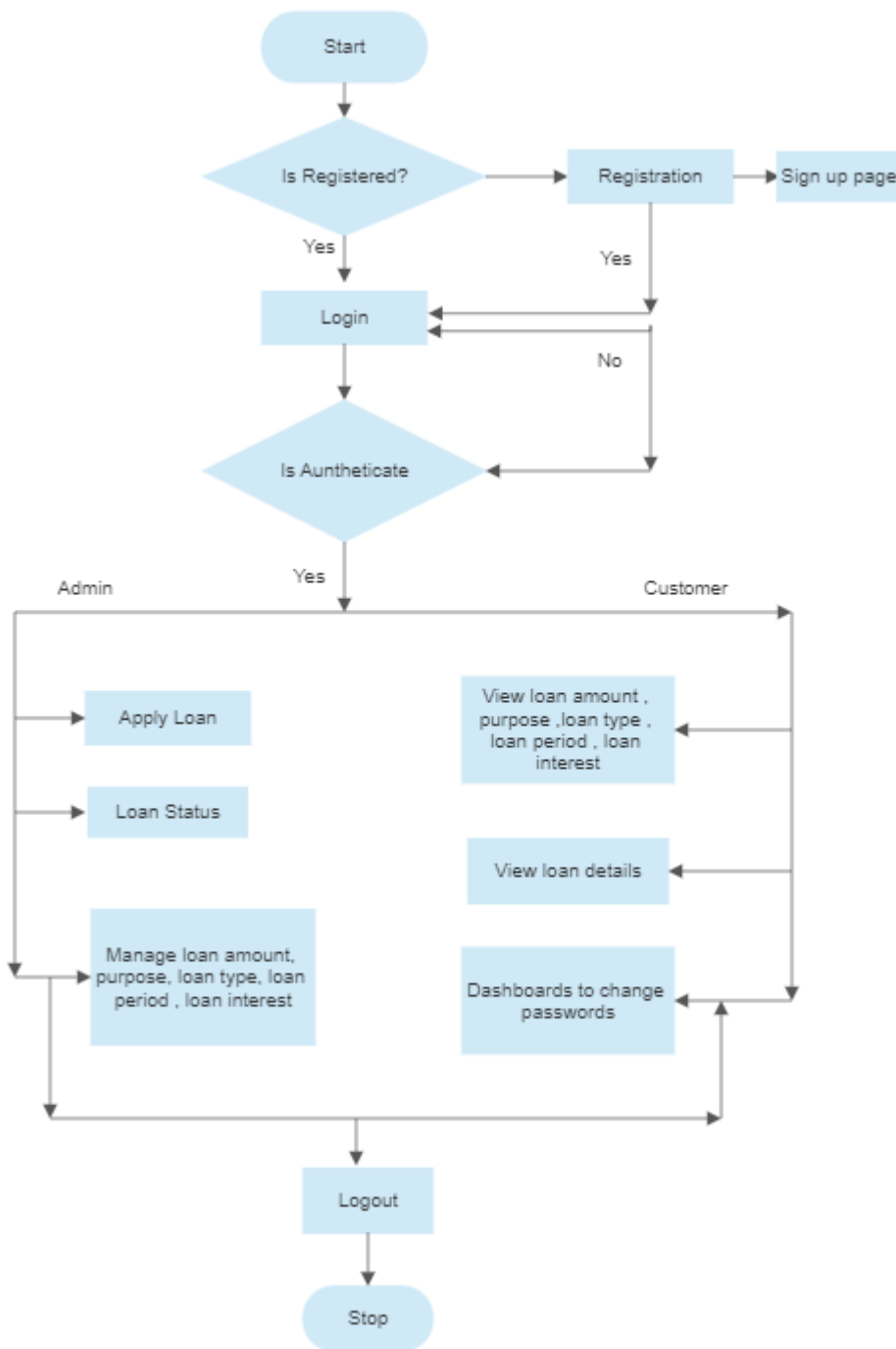


Figure 1 System Flow Diagram

2.2 Working Methodology

The system has software sections, Spring boot is one of the best backend frameworks which allows developers to create standard standalone applications that run on their own, without relying on an external web server. React JS is a declarative, efficient and flexible JavaScript library for

building user interfaces. It lets you compose complex UI's from small and isolated pieces of code called components.

MySQL Server Database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access and process data stored in a computer database, you need a database management system such as MySQL Server. It's an open-source support for every application development used.

To run front-end, you need to use command `npm i`, `npm start`. NPM mean Node Package Manager. Node.js has a set of built-in modules which you can use without any further installation. `npm install` will install dependencies in local `node_modules` folder. `npm start` which starts Create React App(CRA) is a tool to create single-page applications, with `start` argument. `npm` will begin the process to make a development server available.

This Loan Management System is of Web based platform. In this system React JS will be used for frontend, for backend MySQL and Spring boot and some of the tools like IntelliJ, visual studio, node, pycharm, datagrip. The language we used for this project is java in maven project, JDK 17. Implemented machine learning model to predict the approval or rejection of a loan with some given details which helps in giving an idea about the outcome before applying for a loan.

2.3 System Details

This section describes the software details of the system:

Software Details

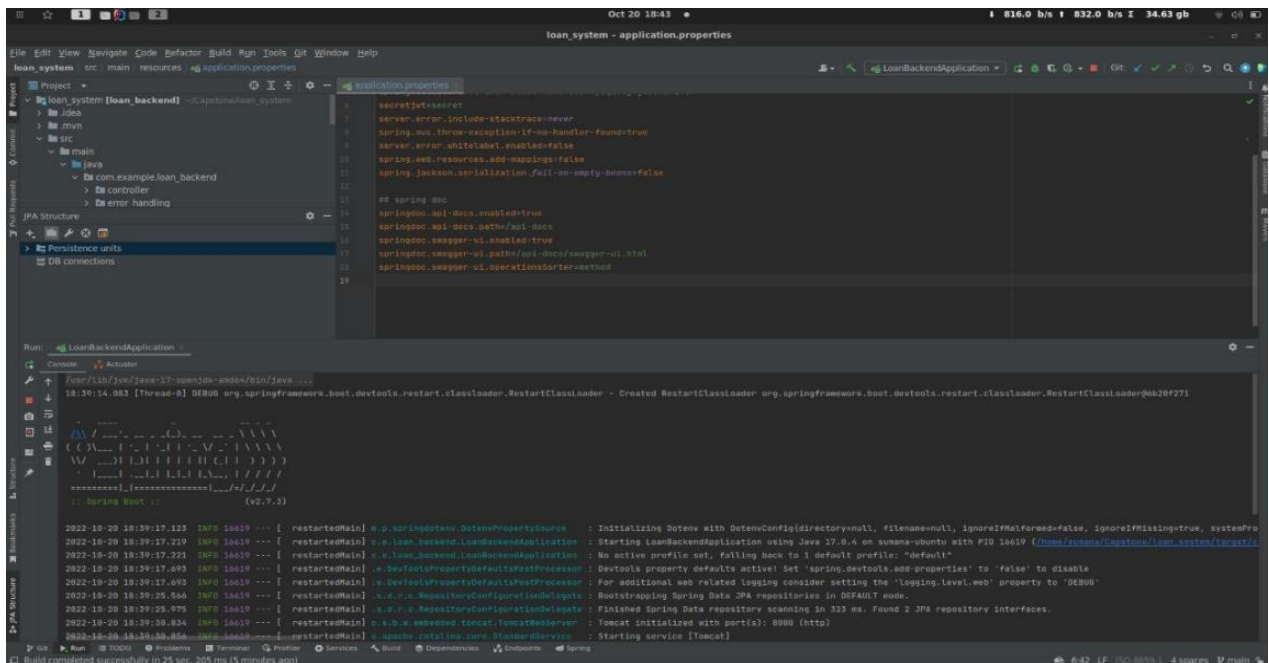
We have two type of Sides - server side and client side. In client-side web server is used any compatible browser can be used. In server side we used frond-end tools like Visual Studio Code and in back-end tools we used MySQL, IntelliJ IDEA, PyCharm, Data grip are used.

i) Spring Boot

The Spring Boot is one of the best backend frameworks that developers count on while performing backend web development tasks. It is a Spring-based framework that allows developers to write production-grade backend web applications in Java.

Developing Spring Boot Project

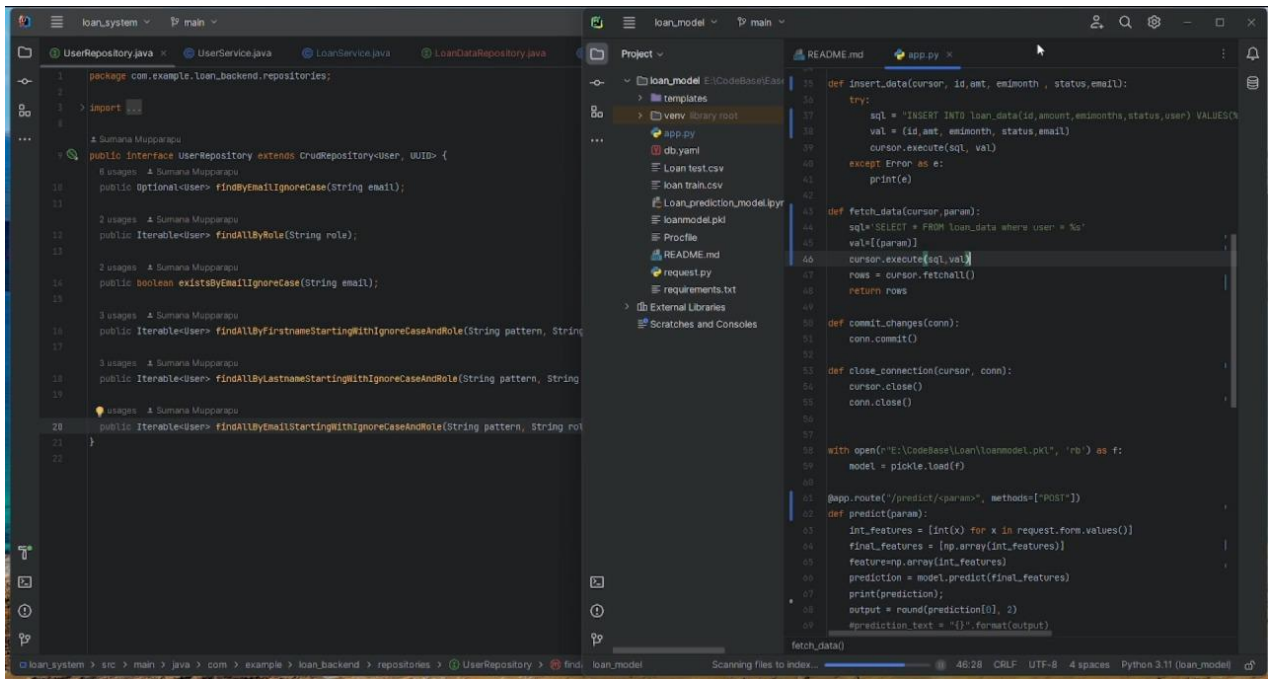
- Make a maven project and creating some of the dependencies for connecting with the database.
- Likewise add screens according to the features of the application. Load it in IntelliJ idea and then run it Remember to edit the application.properties file according to your requirements.



ii) Deploying the machine learning into a web app with flask

Predicting if a loan will be approved or not using python, building the web app using flask.

Run `pip3 install -r requirements.txt` then `Python3 app.py`

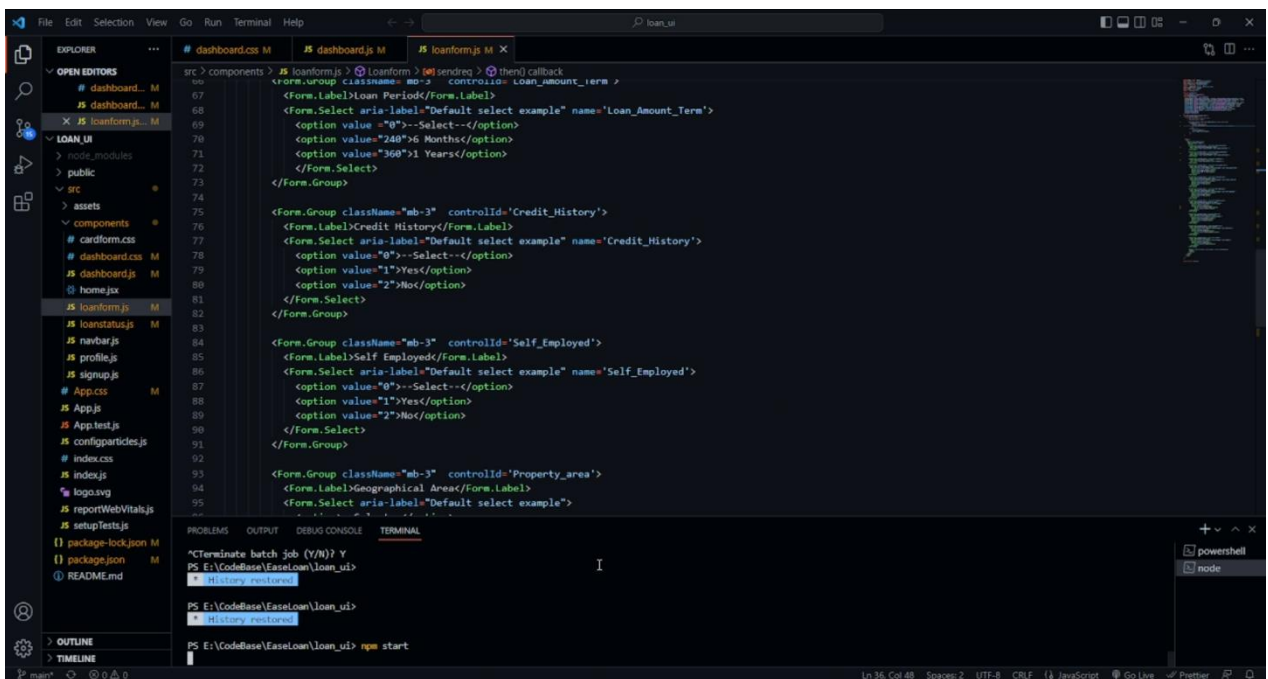


iii) React Js for UI

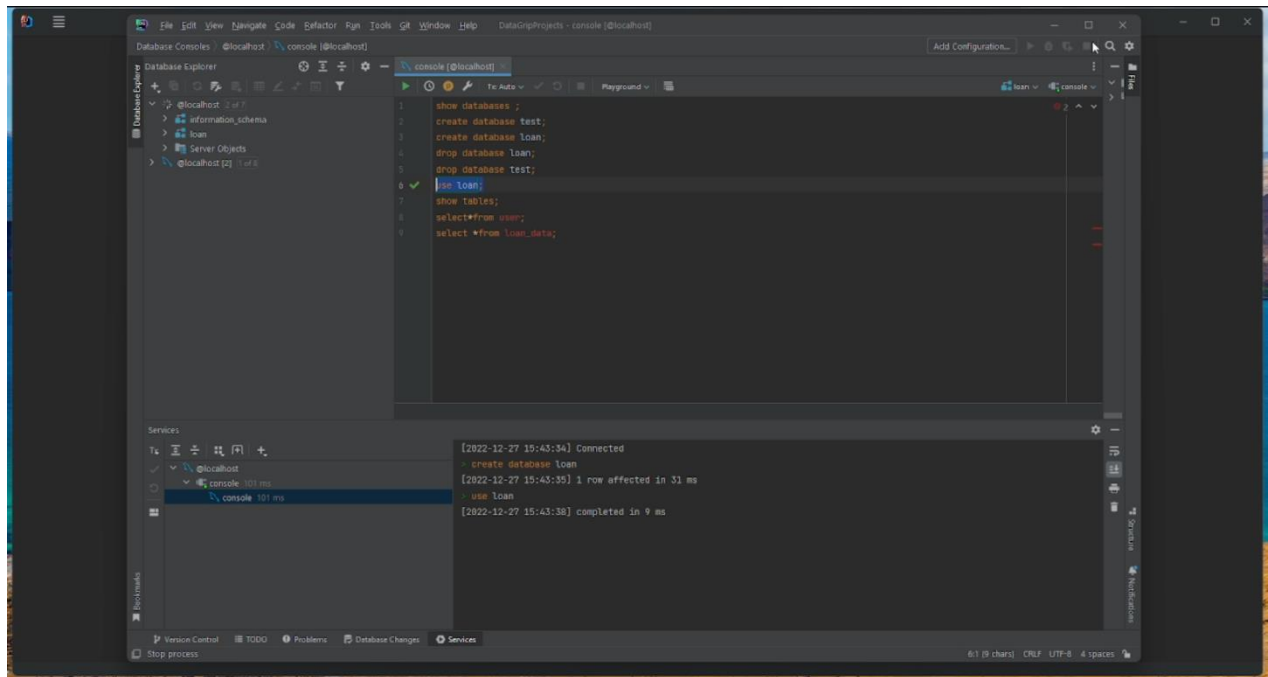
Run npm i then npm start.npm start

Runs the app in the development mode.

Open <http://localhost:3000> to view it in your browser.

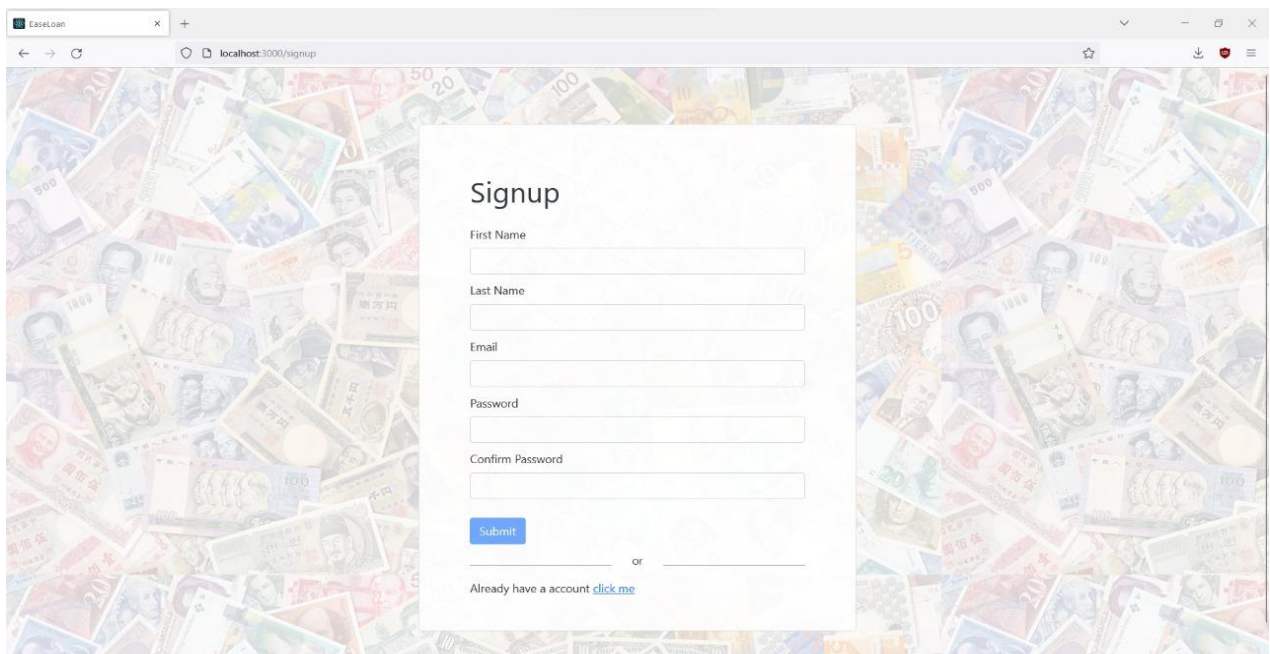


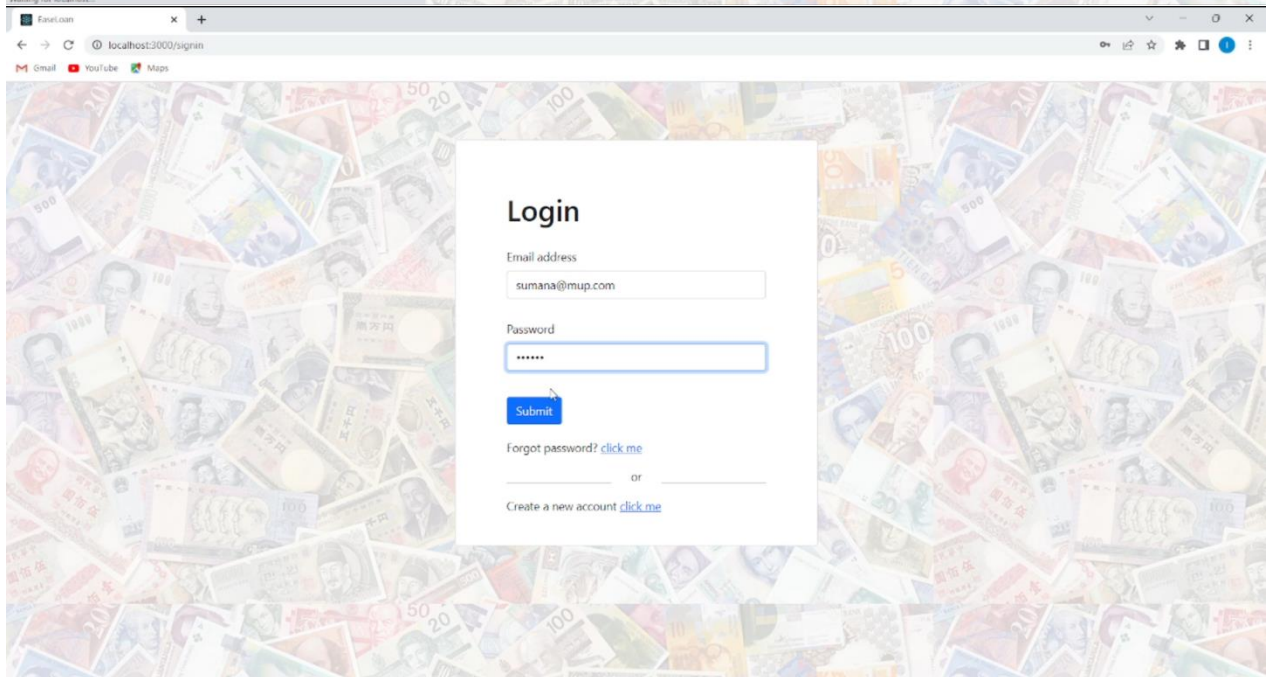
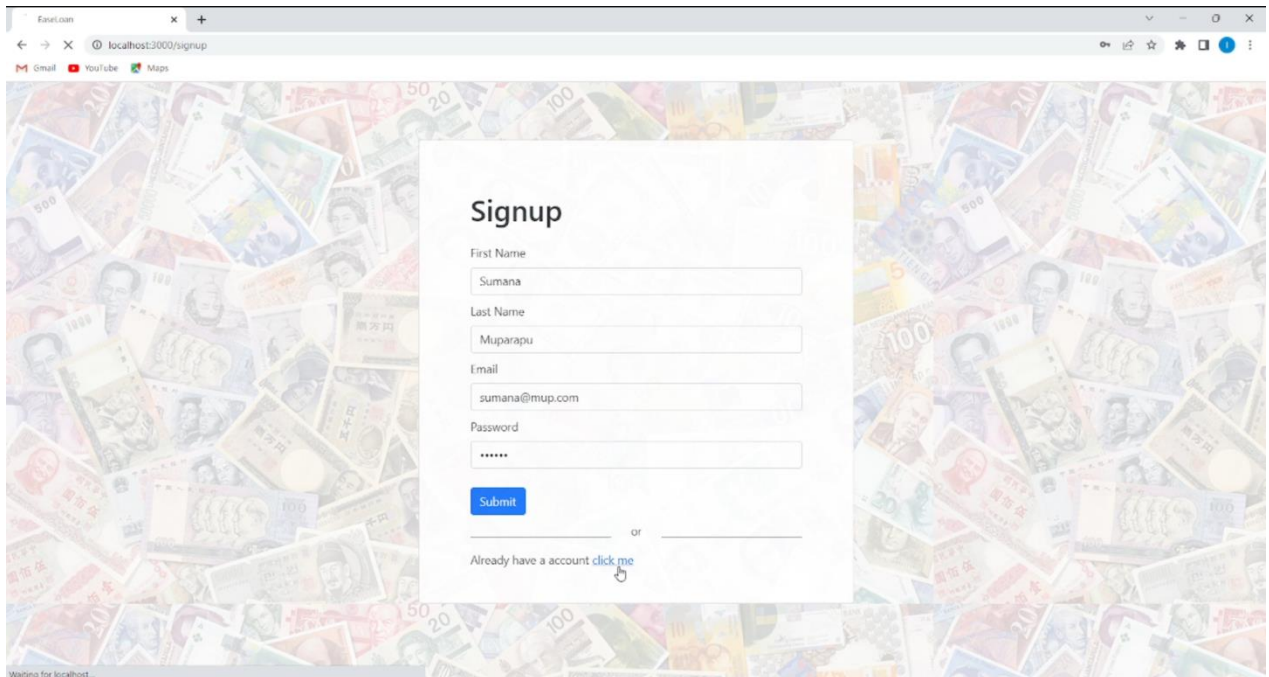
iv)Datagrip for managing databases



CHAPTER 3

RESULTS AND DISCUSSIONS





Apply Loan

Home Profile Signout

Applicant Income

12345

Co Applicant Income

12345

Loan Amount

123

Loan Period

6 Months

Credit History

--Select--

Self Employed

--Select--

Geographical Area

--Select--

Marital Status

Loan Amount

123

Loan Period

6 Months

Credit History

Yes

Self Employed

Yes

Geographical Area

Rural

Marital Status

Married

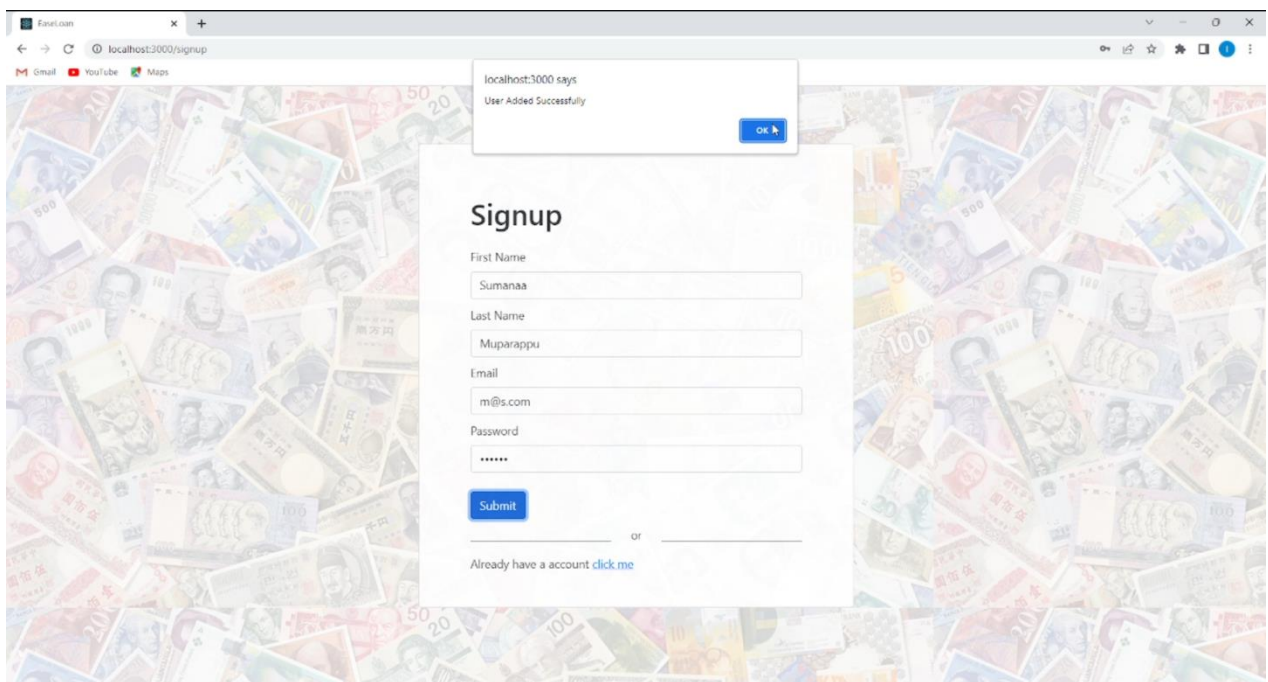
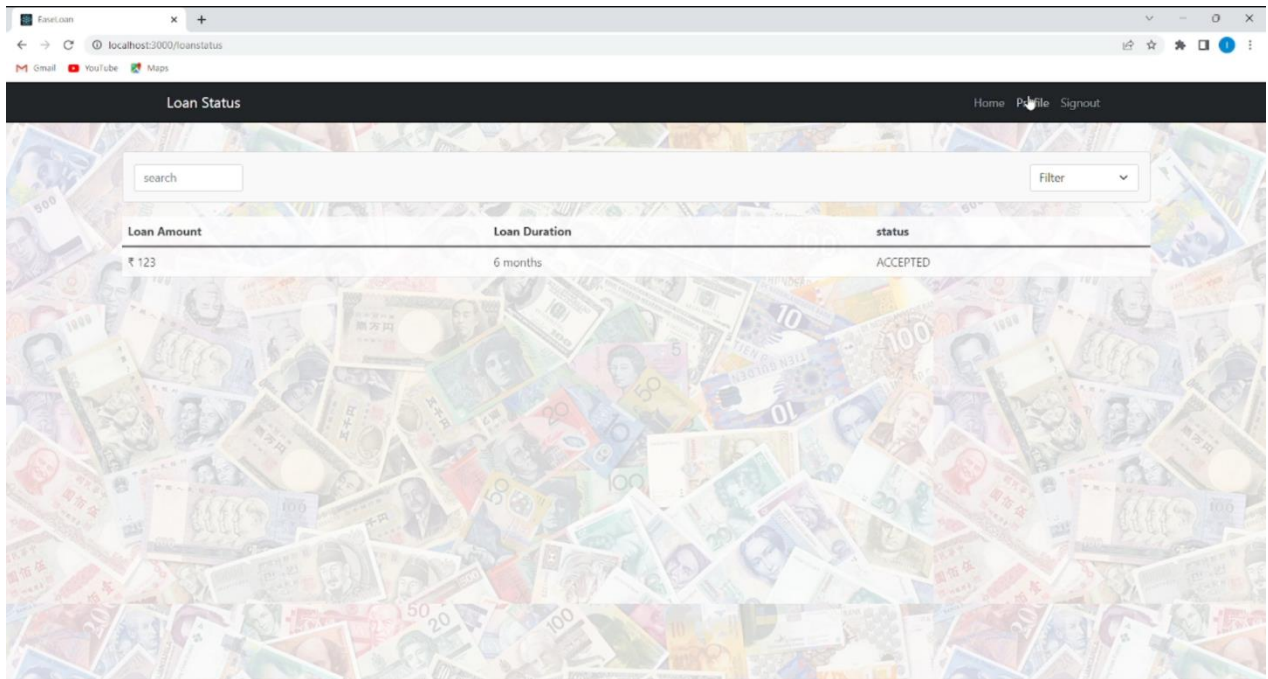
Education Status

Graduate

Gender

male

Submit



Browser: Firefox, URL: localhost:3000/signin

Login

Email address

Password

Forgot password? [click me](#)

_____ or _____

Create a new account [click me](#)

Browser: Firefox, URL: localhost:3000/loanform

Apply Loan

Home Profile Signout

Applicant Income

Co Applicant Income

Loan Amount

Loan Period

Credit History

Self Employed

Geographical Area

Marital Status

Form for loan application (loanform) displayed in a web browser. The form fields and their values are:

- Loan Amount: 12345
- Loan Period: 1 Years
- Credit History: Yes
- Self Employed: Yes
- Geographical Area: Urban
- Marital Status: Married
- Education Status: Graduate
- Gender: male

The form includes a "Submit" button.

Loan Status

Home Profile Signout

search Filter

Loan Amount	Loan Duration	status
₹ 12345	12 months	NULL

CHAPTER 4

CONCLUSION AND FUTURE WORK

Ultimately, in the loan management system, the result of all diligence in a solid loan management system is here. is software that helps the user to find out about various banks and their branches easily. This software reduces the amount of manual data import and provides more performance.

Its system is very friendly and can easily be used by anyone. It also reduces the amount of time it takes to document customer information and other modules. Finally, we can say that this software performs all the tasks correctly and performs the task.

CHAPTER 5

APPENDIX

Code

FRONTEND CODES:

github link:https://github.com/Sumana1008/Capstone/tree/main/loan_ui

loan_ui

App.js

```
import React from 'react';
import Home from './components/home';
import {BrowserRouter as Router,Route,Routes} from "react-router-dom";
import Loanform from './components/loanform';
import Signup from './components/signup';
import Profile from './components/profile';
import Loanlist from './components/loanstatus.js';
import Dashboard from './components/dashboard';
import './App.css'

function App() {
  document.title = "EaseLoan"
  return (
    <Router>
      <Routes>
        <Route exact path="/" element={<Home/>}></Route>
```

```

    <Route exact path="/signin" element={<Home/>}></Route>
    <Route exact path="/loanform" element={<Loanform/>}></Route>
    <Route exact path="/signup" element={<Signup/>}></Route>
    <Route exact path="/profile" element={<Profile/>}></Route>
    <Route exact path="/dashboard" element={<Dashboard/>}></Route>
    <Route exact path="/loanstatus" element={<Loanlist/>}></Route>
  </Routes>
</Router>
);
}

export default App;

```

App.test.js

```

import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});

```

Index.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import 'bootstrap/dist/css/bootstrap.min.css';

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();

```

Signup.js

```

import { useState, useEffect } from "react";
import React from 'react';
import { Form, Card, Button } from "react-bootstrap";
import axios from "axios";

const cardcss = {
  height: "640px",
  padding: "70px",
  width: "600px",

```

```

margin:"80px auto",
opacity: .9
}

function Signup()
{
  const [user, setUser] = useState({
    email: "",
    firstname: "",
    lastname: "",
    password: "",
  });
  const { email, firstname, lastname, password } = user;
  const onChange = (e) => {
    const { name, value } = e.target;
    setFormValues({ ...formValues, [name]: value });
    //console.log(e.target.value);
    setUser({ ...user, [e.target.name]: e.target.value });
  };
  const FormHandle = (e) => {
    e.preventDefault();
    addDataToServer(user);
    console.log(user);
  };
  const addDataToServer = (data) => {
    axios.post("http://localhost:8080/auth/signup", data).then(
      (response) => {
        console.log(response);
        alert("User Added Successfully");
      },
      (error) => {
        console.log(error);
        alert("Operation failed");
      }
    );
  };
  const initialValues = {
    email: "",
    firstname: "",
    lastname: "",
    password: "",
  };
  const [formValues, setFormValues] = useState(initialValues);
  const [formErrors, setFormErrors] = useState({});
  const [isSubmit, setIsSubmit] = useState(false);

  useEffect(() => {
    console.log(formErrors);
    if (Object.keys(formErrors).length === 0 && isSubmit) {
      console.log(formValues);
    }
  }, [formErrors]);
  /*const validate = (values) => {
    const errors = {};
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]{2,}$/i;

```

```

    if (!values.email) {
        errors.email = "Email is required!";
    } else if (!regex.test(values.email)) {
        errors.email = "This is not a valid email format!";
    }
    if (!values.firstname) {
        errors.username = "Firstname is required!";
    } if (!values.lastname) {
        errors.username = "Lastname is required!";
    }

    if (!values.password) {
        errors.password = "Password is required";
    } else if (values.password.length < 4) {
        errors.password = "Password must be more than 4 characters";
    } else if (values.password.length > 10) {
        errors.password = "Password cannot exceed more than 10
characters";
    }
    if (!values.confirmpassword) {
        errors.confirmpassword = "Confirmpassword is required";
    } else if (values.confirmpassword !== values.password) {
        errors.confirmpassword = "ConfirmPassword is not matched";
    }
    return errors;
};
*/
return (

    <Card style={cardcss}>

        <Form>
        <h1 style={{marginBottom:"25px"}}>Signup</h1>
        <Form.Group>
            <Form.Label>First Name</Form.Label>
            <Form.Control style={{marginBottom:"10px"}} id="firstname"
name="firstname" type="text" value={formValues.firstname} onChange={(e) =>
onInputChange(e)} required />
        </Form.Group>
        <Form.Group>
            <Form.Label>Last Name</Form.Label>
            <Form.Control style={{marginBottom:"10px"}} id="lastname"
type="text" name="lastname" value={formValues.lastname} onChange={(e) =>
onInputChange(e)} required/>
        </Form.Group>
        <Form.Group>
            <Form.Label>Email</Form.Label>
            <Form.Control style={{marginBottom:"10px"}} id="email"
name="email" type="email" value={formValues.email} onChange={(e) =>
onInputChange(e)} required/>
        </Form.Group>
        <Form.Group>
            <Form.Label>Password</Form.Label>
            <Form.Control style={{marginBottom:"10px"}} id="password"
type="password" name="password" value={formValues.password} onChange={(e)
=> onInputChange(e)} required/>

```



```

        </Form.Group>
        <Button style={{marginBottom:"10px"}} variant="primary"
className="mt-3" onClick={FormHandle} disabled={!firstname || !email
|| !password || !lastname}>Submit</Button>
        </Form>
        <div style={{display: 'flex', flexDirection: 'row', alignItems:
'center'}}>
            <div style={{flex: 1, height: '1px', backgroundColor: 'grey'}}
/>
            <p style={{width: '70px', textAlign: 'center'}}>or</p>
            <div style={{flex: 1, height: '1px', backgroundColor: 'grey'}}
/>
        </div>
        <Card.Text>Already have a account <a href="signin">click
me</a></Card.Text>
    </Card>

    );
}

export default Signup;

```

Profile.js'

```

import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import {Card} from "react-bootstrap";
import React from 'react';
//import { useNavigate } from 'react-router-dom';
import Navbar from "../navbar"
function Profile() {
    const firstname=localStorage.getItem("fname");
    const lastname=localStorage.getItem("lname")
    const email=localStorage.getItem("user")

    const cardcss = {
        height:"400px",
        padding:"70px",
        width:"500px",
        margin:"80px auto",
    }

    return (
        <div>
            <Navbar name="Profile"/>
            <Card style={cardcss}>
                <Form>
                    <Form.Group className="mb-3" controlId="formbasicFirstName">
                        <Form.Label>FirstName </Form.Label>
                        <Form.Control placeholder={firstname} disabled/>
                    </Form.Group>

                    <Form.Group className="mb-3" controlId="formBasicLastName">
                        <Form.Label>LastName </Form.Label>

```

```

        <Form.Control placeholder={lastname} disabled/>
      </Form.Group>
      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Email </Form.Label>
        <Form.Control placeholder={email} disabled/>
      </Form.Group>
    </Form>
  </Card>
</div>
);
}

export default Profile;

```

Navbar.js

```

import React from 'react';
import {Navbar,Nav,Container} from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';

function navbar(props) {

  const navigate = useNavigate();
  const signoutfunc = () => {
    localStorage.removeItem("user");
    localStorage.removeItem("token");
    navigate("/signin");
  }
  return (
    <Navbar collapseOnSelect expand="lg" bg="dark" variant="dark">
      <Container>
        <Navbar.Brand href="#home">{props.name}</Navbar.Brand>
        <Navbar.Toggle aria-controls="responsive-navbar-nav" />
        <Navbar.Collapse id="responsive-navbar-nav">
          <Nav className="me-auto"></Nav>
          <Nav>
            <Nav.Link href="/dashboard">Home</Nav.Link>
            <Nav.Link href="/profile">Profile</Nav.Link>
            <Nav.Link eventKey={2} onClick={signoutfunc}>
              Signout
            </Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default navbar;

```

Loanstatus.js

```

import React , { useState, useEffect } from 'react';
import { Button, Card,Form,Table} from "react-bootstrap";
//import { useNavigate } from 'react-router-dom';
import Navbar from "../navbar";
import axios from 'axios';

function Loanlist(){

```



```

var user=localStorage.getItem("user");
const [data, setData] = useState([]);
var myHeaders = new Headers();
myHeaders.append("Access-Control-Allow-Origin", "*");
myHeaders.append("Access-Control-Allow-Methods",
"GET, PUT, POST, DELETE, PATCH, OPTIONS");
useEffect(() => {
  axios.get('http://127.0.0.1:5000/data/'+user)
    .then(res => {
      setData(res.data);
    })
}, []);
const divcss = {
  padding:"40px 10rem"
}
const cardheader = {
  backgroundColor:"#F9F9F9",
  height:"70px",
  marginBottom:"20px"
}
console.log(data)
return(
  <div>
<Navbar name="Loan Status"/>
    <div style={divcss}>
      <Card style={cardheader}>
        <Card.Body>
          <Form.Control style={{float:"left",width:"150px"}}
placeholder="search"/>
          <Form.Select style={{float:"right",width:"150px"}}>
            <option>Filter</option>
            <option value="ACCEPTED">accepted</option>
            <option value="PENDING">pending</option>
            <option value="REJECTED">rejected</option>
          </Form.Select>
        </Card.Body>
      </Card>
      <Table style={{backgroundColor:"white", opacity:"0.8"}}>
        <thead>
          <tr>
            <th>Loan Amount</th>
            <th>Loan Duration</th>
            <th>status</th>
          </tr>
        </thead>
        <tbody>
          {data.map((item,index) => (
            <tr key={index}>
              <td>₹ {item[1]}</td>
              <td>{item[2]} months</td>
              <td>{item[3]}</td>
            </tr>
          ))}
        </tbody>
      </Table>
    </div>
  )

```

```

        </div>
    );
}
export default Loanlist;
/* {data?.map((row) => (
    <tr key={row.id}>
        <td>{row.amount}</td>
        <td>{row.emimonths}</td>
        <td>{row.status}</td>
    </tr>
    )})* */

```

Loanform.js

```

import React from 'react';
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import {Card} from "react-bootstrap";
//import { useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import Navbar from "../navbar"
import axios from "axios";
import "../cardform.css"

function Loanform() {
    const navigate = useNavigate();
    const FormHandle=(e)=>{
        const loginFormData = new FormData();
        loginFormData.append("ApplicantIncome",
document.getElementById('ApplicantIncome').value)
        loginFormData.append("CoapplicantIncome",
document.getElementById('CoapplicantIncome').value)
        loginFormData.append("LoanAmount",
document.getElementById('LoanAmount').value)
        loginFormData.append("Loan_Amount_Term",
document.getElementById('Loan_Amount_Term').value)
        loginFormData.append("Credit_History",
document.getElementById('Credit_History').value)
        loginFormData.append("Self_Employed",
document.getElementById('Self_Employed').value)
        loginFormData.append("Property_area",
document.getElementById('Property_area').value)
        loginFormData.append("Married", document.getElementById('Married').value)
        loginFormData.append("Education",
document.getElementById('Education').value)
        loginFormData.append("Gender", document.getElementById("Gender").value)
        sendreq(loginFormData)
        navigate("/dashboard")
    }
    var user=localStorage.getItem("user");
    const sendreq=(data)=> {
        // store the states in the form data
        // make axios post request

        axios.post("http://127.0.0.1:5000/predict/"+user, data).then(
            (response) => {
                console.log(response);
            }
        )
    }
}

```

```

        alert("Loan Applied Successfully,Check results in Loan
Status");
    },
    (error) => {
        console.log(error);
        alert("Operation failed");
    }
);
}

return (
    <div>
        <Navbar name="Apply Loan"/>
        <Card className='cardcs'>
            <Form >
                <Form.Group className="mb-3" controlId='ApplicantIncome'>
                    <Form.Label>Applicant Income</Form.Label>
                    <Form.Control type="Account Number" name='ApplicantIncome' />
                </Form.Group>

                <Form.Group className="mb-3" controlId="CoapplicantIncome">
                    <Form.Label>Co ApplicantIncome</Form.Label>
                    <Form.Control type="Account Number" name='CoapplicantIncome' />
                </Form.Group>

                <Form.Group className="mb-3" controlId='LoanAmount'>
                    <Form.Label>Loan Amount</Form.Label>
                    <Form.Control type="Loan Amount" name='LoanAmount' />
                </Form.Group>

                <Form.Group className="mb-3" controlId='Loan_Amount_Term'>
                    <Form.Label>Loan Period</Form.Label>
                    <Form.Select aria-label="Default select example"
name='Loan_Amount_Term'>
                        <option value ="0">--Select--</option>
                        <option value="240">6 Months</option>
                        <option value="360">1 Years</option>
                    </Form.Select>
                </Form.Group>

                <Form.Group className="mb-3" controlId='Credit_History'>
                    <Form.Label>Credit History</Form.Label>
                    <Form.Select aria-label="Default select example"
name='Credit_History'>
                        <option value="0">--Select--</option>
                        <option value="1">Yes</option>
                        <option value="2">No</option>
                    </Form.Select>
                </Form.Group>

                <Form.Group className="mb-3" controlId='Self_Employed'>
                    <Form.Label>Self Employed</Form.Label>
                    <Form.Select aria-label="Default select example"
name='Self_Employed'>
                        <option value="0">--Select--</option>

```

```

        <option value="1">Yes</option>
        <option value="2">No</option>
      </Form.Select>
    </Form.Group>

    <Form.Group className="mb-3" controlId='Property_area'>
      <Form.Label>Geographical Area</Form.Label>
      <Form.Select aria-label="Default select example">
        <option>--Select--</option>
        <option value="2">Urban</option>
        <option value="1">Semi Urban</option>
        <option value="0">Rural</option>
      </Form.Select>
    </Form.Group>

    <Form.Group className="mb-3" controlId='Married'>
      <Form.Label>Marrital Status</Form.Label>
      <Form.Select aria-label="Default select example" >
        <option value="0">--Select--</option>
        <option value="1">Married</option>
        <option value="2">Single</option>
      </Form.Select>
    </Form.Group>

    <Form.Group className="mb-3" controlId='Education'>
      <Form.Label>Education Status</Form.Label>
      <Form.Select aria-label="Default select example"
name='Education'>
        <option>--Select--</option>
        <option value="0">Graduate</option>
        <option value="1">Not Graduate</option>
      </Form.Select>
    </Form.Group>

    <Form.Group className="mb-3" controlId='Gender'>
      <Form.Label>Gender</Form.Label>
      <Form.Select aria-label="Default select example" name='Gender'>
        <option>--Select--</option>
        <option value="1">male</option>
        <option value="0">female</option>
      </Form.Select>
    </Form.Group>

    <Button variant="primary" type="submit" onClick={FormHandle}>
      Submit
    </Button>
  </Form>
</Card>
</div>
);
}

export default Loanform;

```

Dashboard.js

```

import React from 'react';
//import Button from 'react-bootstrap/Button';
import {useNavigate } from 'react-router-dom';
import Navbar from "../navbar"
import {Card} from "react-bootstrap";
import './dashboard.css';
function Dashboard() {

  const navigate = useNavigate();

  const applybutton = (e) =>{
    navigate("/loanform");
  }
  const statusbutton = () => {
    navigate("/loanstatus");
  }

  return (
    <div className="App">
      <Navbar name="DashBoard"/>
      <div className="dud md-flex flex-row mb-3 align-self-center"
style={{margin: "auto", display:"flex", alignItems:"center",
justifyContent:"center"}}>
        <Card onClick={applybutton} className="cardm">
          <Card.Body>
            <Card.Title style={{fontSize:"3.5vw",textAlign: "center",
alignSelf: "center"}}>ApplyLoan</Card.Title>
            <Card.Text>
            </Card.Text>
          </Card.Body>
        </Card>
        <Card onClick={statusbutton} className="cardm" style={{}}>
          <Card.Body>
            <Card.Title style={{fontSize:"3.5vw",opacity:"0.9",textAlign:
"center",
alignSelf: "center"}}>Loan Status</Card.Title>
            <Card.Text>
            </Card.Text>
          </Card.Body>
        </Card>
      </div>
    </div>
  );
}

export default Dashboard;

```

Dashboard.css

```

.cardm{
  width: 50%;
  height:100%;
  margin-left:10px ;
  margin-right: 10px;
  margin-top: 20vh;
  margin-bottom: 10%;
  top:50%;

```

```

padding-top: 10%;
padding-bottom: 10%;
box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
transition: 0.3s;
opacity: 0.8;
}
.cardm:hover {
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
  background-color: rgb(221, 221, 221);
}

.duv{
  display: flex;
  justify-content: center;
  align-items: center;
}

```

Home.jsx

```

import React from 'react';
import { useState } from "react";
import { Form, Card, Button } from "react-bootstrap";
import axios from "axios";
import { useNavigate } from "react-router-dom";
const Home = () =>{
  const navigate = useNavigate();
  const [email, setEmail] = useState("")
  const [password, setPassword] = useState("")
  const [user, setUser] = useState([])

  const cardcss = {
    height: "560px",
    padding: "70px",
    width: "500px",
    margin: "80px auto",
  }
  const mailchange = (e) =>{
    setEmail(e.target.value);
  }
  const passwordchange = (e) =>{
    setPassword(e.target.value);
  }
  const formsubmit = (e) =>{
    e.preventDefault()
    axios.post(
      "http://localhost:8080/auth/signin", {
        email: email,
        password: password
      }
    )
    .then((res) => {
      if(res.status === 200 || res.status === 201){
        localStorage.setItem("token", res.data.jwt);
        localStorage.setItem("user", email);
        localStorage.setItem("fname", res.data.user.firstname)
        localStorage.setItem("lname", res.data.user.lastname)

```

```

        const seconds = 60 * 60 * 1000;
        const expiry = new Date(new Date().getTime() + seconds);
        localStorage.setItem("expiry", expiry.toISOString());
        navigate("/dashboard");
        window.location.reload()
    }
})
.catch(function(error) {
    document.getElementById("bad").style.display = "block";
});
}
return (
    <div>
        <Card style={cardcss}>
            <Form>
                <Form.Label id="bad"
style={{display:"none",backgroundColor:"rgba(255, 53, 71, 0.5)",
border:"2px solid red",padding:"0px 20px"}}>Bad credentials</Form.Label>
                <h1 style={{marginBottom:"30px"}}>Login</h1>
                <Form.Group>
                    <Form.Label>Email address </Form.Label>
                    <Form.Control onChange={mailchange}
style={{marginBottom:"30px"}} type="email"/>
                </Form.Group>
                <Form.Group>
                    <Form.Label>Password </Form.Label>
                    <Form.Control onChange={passwordchange}
style={{marginBottom:"20px"}} type="password"/>
                </Form.Group>
                <Button style={{marginBottom:"20px"}} variant="primary"
type="submit" className="mt-3" onClick={formsubmit} disabled={!email
|| !password}>Submit</Button>
            </Form>
            <Card.Text>Forgot password? <a href="#">click me</a></Card.Text>
            <div style={{display: 'flex', flexDirection: 'row', alignItems:
'center'}}>
                <div style={{flex: 1, height: '1px', backgroundColor: 'grey'}}
/>
                <p style={{width: '70px', textAlign: 'center'}}>or</p>
                <div style={{flex: 1, height: '1px', backgroundColor: 'grey'}}
/>
            </div>
            <Card.Text>Create a new account <a href="signup">click
me</a></Card.Text>
        </Card>
    </div>
);
};
export default Home;

```

Configparticles.js

```

import Particles from "react-tsparticles";

const particlesconfig = () => {
    <Particles
        id="tsparticles"

```

```

options={
  background: {
    color: {
      value: "#0d47a1",
    },
  },
  fpsLimit: 60,
  interactivity: {
    detectsOn: "canvas",
    events: {
      onClick: {
        enable: true,
        mode: "push",
      },
      onHover: {
        enable: true,
        mode: "repulse",
      },
      resize: true,
    },
    modes: {
      bubble: {
        distance: 400,
        duration: 2,
        opacity: 0.8,
        size: 40,
      },
      push: {
        quantity: 4,
      },
      repulse: {
        distance: 200,
        duration: 0.4,
      },
    },
  },
  particles: {
    color: {
      value: "#ffffff",
    },
    links: {
      color: "#ffffff",
      distance: 150,
      enable: true,
      opacity: 0.5,
      width: 1,
    },
    collisions: {
      enable: true,
    },
    move: {
      direction: "none",
      enable: true,
      outMode: "bounce",
      random: false,
      speed: 6,
      straight: false,

```



```

    },
    number: {
      density: {
        enable: true,
        value_area: 800,
      },
      value: 80,
    },
    opacity: {
      value: 0.5,
    },
    shape: {
      type: "circle",
    },
    size: {
      random: true,
      value: 5,
    },
  },
  detectRetina: true,
}}
/>
};
export default particlesconfig;

```

Index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed
      on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the
      build.
      Only files inside the `public` folder can be referenced from the
      HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico"
      will
      work correctly both with client-side routing and a non-root public
      URL.

```

```

    Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <title>LoanApp</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>

```

BACKEND CODES:

github link:https://github.com/Sumana1008/Capstone/tree/main/loan_system

loan_system

AccountRoles.java

```

package com.example.loan_backend;
public enum AccountRoles {
    ROLE_ADMIN, ROLE_USER
}

```

ApiAuthEntryPoint.java

```

package com.example.loan_backend;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerExceptionResolver;
@Component("delegatedAuthenticationEntryPoint")
public class ApiAuthEntryPoint implements AuthenticationEntryPoint {
    @Autowired
    @Qualifier("handlerExceptionResolver")
    private HandlerExceptionResolver resolver;
    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException authException) throws IOException, ServletException {
        resolver.resolveException(request, response, null, authException);
    }
}

```

CustomerUserDetails.java

```
package com.example.loan_backend;
import com.example.loan_backend.models.User;
import java.util.Arrays;
import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
public class CustomUserDetails implements UserDetails {
    private List<GrantedAuthority> roles;
    private User user;
    public CustomUserDetails(User user) {
        this.roles = Arrays.asList(user.getRole().split(",")).stream().map(SimpleGrantedAuthority::new)
            .collect(Collectors.toList());
        this.user = user;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return roles;
    }
    @Override
    public String getPassword() {
        return this.user.getPassword();
    }
    @Override
    public String getUsername() {
        return this.user.getEmail();
    }
    public User getUser() {
        return this.user;
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

LoanBackendApplication.java

```
package com.example.loan_backend;
import org.springframework.boot.WebApplicationType;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
@SpringBootApplication
public class LoanBackendApplication {
```

```

public static void main(String[] args) {
    new SpringApplicationBuilder(LoanBackendApplication.class)
        .web(WebApplicationType.SERVLET)
        .run(args);
}
}

```

LoanStatus.java

```

package com.example.loan_backend;
public enum LoanStatus {
    PENDING, ACCEPTED, REJECTED
}

```

OpenApiConfig

```

package com.example.loan_backend;
import io.swagger.v3.oas.annotations.OpenAPIDefinition;
import io.swagger.v3.oas.annotations.info.Info;
@OpenAPIDefinition(info = @Info(title = "Loan", description = "Here is the documentation for loan backend, you can use
to check the response types and request types. For protected routes /signup and /signin if not registered , else use
auth/signin to generate a token and press the lock on protected routes before sending any request to them.))
public class OpenApiConfig {
}

```

request

LoanRequest.java

```

package com.example.loan_backend.request;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Positive;
import javax.validation.constraints.Size;
public class LoanRequest {
    @NotNull
    @Positive
    public double amount;
    @NotNull
    @Size(min = 5, max = 100)
    public String purpose;
    @NotNull
    @Positive
    public int emimonths;
    @NotNull
    @Positive
    public double interest;
}

```

LoginRequest.java

```

package com.example.loan_backend.request;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
public class LoginRequest {
    @NotNull
    @Email
    public String email;
    @NotNull
    @Pattern(regexp = "^[a-zA-Z0-9]{4,20}", message = "password should be between 4 and 20 with no blank spaces")
    public String password;
}

```

SignupRequest.java

```

package com.example.loan_backend.request;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
public class SignupRequest {
    @NotNull
    @Size(min = 2, max = 20, message = "lastname should be between 2 and 20")
    public String firstname;
    @NotNull
    @Size(min = 2, max = 20, message = "lastname should be between 2 and 20")
    public String lastname;
    @NotNull
    @Email
    public String email;
    @NotNull
    @Pattern(regexp = "^[a-zA-Z0-9]{4,20}", message = "password should be between 4 and 20 with no blank spaces")
    public String password;
}

```

models

Loan.java

```

package com.example.loan_backend.models;
import com.example.loan_backend.LoanStatus;
import com.example.loan_backend.request.LoanRequest;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.annotations.Type;
import java.io.Serializable;
import java.util.UUID;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
@Entity
@Getter
@Setter
@NoArgsConstructor
public class Loan implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(columnDefinition = "VARCHAR(36)")
    @Type(type = "uuid-char")
    private UUID id;
    @Column(nullable = false)
    private String status;
    @Column(nullable = false)
    private double amount;
    @Column(nullable = false)
    private String purpose;
    @Column(nullable = false)
    private int emimonths;
    @Column(nullable = false)
}

```

```

private double interest;
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "user_id", nullable = false)
private User user;
public void setStatus(LoanStatus status) {
    this.status = String.valueOf(status);
}
public Loan(LoanRequest lr) {
    this.amount = lr.amount;
    this.emimonths = lr.emimonths;
    this.interest = lr.interest;
    this.purpose = lr.purpose;
}
}

```

User.java

```

package com.example.loan_backend.models;
import com.example.loan_backend.AccountRoles;
import com.example.loan_backend.request.SignupRequest;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonProperty.Access;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.hibernate.annotations.Type;
import java.io.Serializable;
import java.util.List;
import java.util.UUID;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
@Entity
@Getter
@Setter
@NoArgsConstructor
public class User implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(columnDefinition = "VARCHAR(36)")
    @Type(type = "uuid-char")
    private UUID id;
    @Column(nullable = false)
    private String firstname;
    @Column(nullable = false)
    private String lastname;
    @JsonProperty(access = Access.WRITE_ONLY)
    @Column(nullable = false)
    private String password;
    @JsonIgnore
    @Column(nullable = false)
    private String role;
    @Column(nullable = false, unique = true)
    private String email;
    @JsonIgnore

```

```

@OneToMany(fetch = FetchType.EAGER, mappedBy = "user")
private List<Loan> loans;
public void setRole(AccountRoles role) {
    this.role = String.valueOf(role);
}
public User(SignupRequest sr) {
    this.firstname = sr.firstname;
    this.lastname = sr.lastname;
    this.email = sr.email;
}
}

```

error handling files

EntityNotFoundException.java

```

package com.example.loan_backend.error_handling;
public class EntityNotFoundException extends RuntimeException {
    private String message;
    public EntityNotFoundException(String msg) {
        message = msg;
    }
    public String getMessage() {
        return this.message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
}

```

UncaughtApiExceptionHandler.java

```

package com.example.loan_backend.error_handling;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;
@Order(Ordered.LOWEST_PRECEDENCE)
@RestControllerAdvice
public class UncaughtApiExceptionHandler implements CustomExceptionHandler {
    @ExceptionHandler(Exception.class)
    @ResponseStatus(code = HttpStatus.SERVICE_UNAVAILABLE)
    public ResponseEntity<Object> handle(Exception ex, HttpServletRequest request,
        HttpServletResponse response) {
        if (ex instanceof NullPointerException) {
            return buildResponseEntity(new ApiError(HttpStatus.BAD_REQUEST, ex));
        }
        return buildResponseEntity(
            new ApiError(HttpStatus.INTERNAL_SERVER_ERROR, "Server Error", ex));
    }
    public ResponseEntity<Object> buildResponseEntity(ApiError err) {
        return new ResponseEntity<>(err, err.getHttpStatus());
    }
}

```

CustomExceptionHandler.java

```

package com.example.loan_backend.error_handling;
import org.springframework.http.ResponseEntity;
public interface CustomExceptionHandler {
    public ResponseEntity<Object> buildResponseEntity(ApiError err);
}

```

ApiError.java

```

package com.example.loan_backend.error_handling;
import java.time.LocalDateTime;
import java.util.List;
import org.springframework.http.HttpStatus;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
@JsonTypeInfo(include = JsonTypeInfo.As.WRAPPER_OBJECT, use = JsonTypeInfo.Id.DEDUCTION, property = "error",
visible = true)
public class ApiError {
    private HttpStatus httpStatus;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy hh:mm:ss")
    private LocalDateTime timestamp;
    private String message;
    private Object debugMessage;
    private List<ApiSubErrors> subErrors;
    // Constructors
    public ApiError() {
        timestamp = LocalDateTime.now();
    }
    public ApiError(HttpStatus status) {
        this();
        httpStatus = status;
    }
    public ApiError(HttpStatus status, Throwable ex) {
        this(status);
        message = "Unexpected Error";
        debugMessage = ex.getLocalizedMessage();
    }

    public ApiError(HttpStatus status, String msg, Throwable ex) {
        this(status);
        message = msg;
        debugMessage = ex.getLocalizedMessage();
    }
    public ApiError(HttpStatus status, String msg, Object debug) {
        this(status);
        message = msg;
        this.debugMessage = debug;
    }
    // Getters and Setters
    public HttpStatus getHttpStatus() {
        return this.httpStatus;
    }
    public void setHttpStatus(HttpStatus httpStatus) {
        this.httpStatus = httpStatus;
    }
    public LocalDateTime getTimestamp() {
        return this.timestamp;
    }
    public void setTimestamp(LocalDateTime timestamp) {
        this.timestamp = timestamp;
    }
}

```



```

public String getMessage() {
    return this.message;
}
public void setMessage(String message) {
    this.message = message;
}
public Object getDebugMessage() {
    return this.debugMessage;
}
public void setDebugMessage(String debugMessage) {
    this.debugMessage = debugMessage;
}
public List<ApiSubErrors> getSubErrors() {
    return this.subErrors;
}
public void setSubErrors(List<ApiSubErrors> subErrors) {
    this.subErrors = subErrors;
}
}

```

Controller files

AdminController.java

```

package com.example.loan_backend.controller;
import java.util.UUID;
import javax.validation.constraints.Email;
import com.example.loan_backend.response.MsgDataResponse;
import com.example.loan_backend.services.LoanService;
import com.example.loan_backend.services.UserService;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/admin")
@Validated
@SecurityRequirement(name = "bearer-key")
public class AdminController {
    @Autowired
    private UserService userService;
    @Autowired
    private LoanService loanService;
    @GetMapping(value = "/users")
    public ResponseEntity<?> getUsers() {
        return new ResponseEntity<>(new MsgDataResponse("All Users", userService.getAllUsers()), HttpStatus.OK);
    }
    @GetMapping("/getSearchUsersByFirstName/{firstname}")
    public ResponseEntity<?> getUsersStartsWithFirstName(@PathVariable String firstname) {
        return new ResponseEntity<>(new MsgDataResponse("All Users", userService.getUsersByFirstName(firstname)),
            HttpStatus.OK);
    }
    @GetMapping("/getSearchUsersByLastName/{lastname}")
    public ResponseEntity<?> getUsersStartsWithLastName(@PathVariable String lastname) {
        return new ResponseEntity<>(new MsgDataResponse("All Users", userService.getUsersByLastName(lastname)),
            HttpStatus.OK);
    }
    @GetMapping("/getSearchUsersByEmail/{email}")

```

```

public ResponseEntity<?> getUsersStartsWithEmail(@PathVariable String email) {
    return new ResponseEntity<>(new MsgDataResponse("All Users", userService.getUsersByEmail(email)),
        HttpStatus.OK);
}
@GetMapping(value = "/getAllLoans")
public ResponseEntity<?> getAllLoans() {
    return new ResponseEntity<>(new MsgDataResponse("All Loans", loanService.getAllLoans()), HttpStatus.OK);
}
@GetMapping(value = "/getLoansByUser/{email}")
public ResponseEntity<?> getLoansByUser(@PathVariable @Email String email) {
    return new ResponseEntity<>(new MsgDataResponse("All User Loans", loanService.getLoansByUserEmail(email)),
        HttpStatus.OK);
}
@PostMapping(value = "/acceptLoan/{loan_id}")
public ResponseEntity<?> acceptLoan(@PathVariable UUID loan_id) {
    loanService.acceptLoanById(loan_id);
    return new ResponseEntity<>("Loan Accepted Successfully", HttpStatus.ACCEPTED);
}
@PostMapping(value = "/rejectLoan/{loan_id}")
public ResponseEntity<Object> rejectLoan(@PathVariable UUID loan_id) {
    loanService.rejectLoanById(loan_id);
    return new ResponseEntity<>("Loan Rejected Successfully", HttpStatus.ACCEPTED);
}
@GetMapping(value = "/getPendingLoans")
public ResponseEntity<?> getPendingLoans() {
    return new ResponseEntity<>(new MsgDataResponse("All Pending Loans", loanService.getAllPendingLoans()),
        HttpStatus.OK);
}
}

```

LoanController.java

```

package com.example.loan_backend.controller;
import com.example.loan_backend.models.Loan;
import com.example.loan_backend.request.LoanRequest;
import com.example.loan_backend.response.MsgDataResponse;
import com.example.loan_backend.services.LoanService;
import com.example.loan_backend.services.UserService;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RequestMapping("/loan")
@RestController
@SecurityRequirement(name = "bearer-key")
public class LoanController {
    @Autowired
    private UserService userService;
    @Autowired
    private LoanService loanService;
    @PostMapping(value = "/reqLoan")
    public ResponseEntity<Object> addloan(@Valid @RequestBody LoanRequest lr) {
        Loan loan = new Loan(lr);
    }
}

```

```

String email = SecurityContextHolder.getContext().getAuthentication().getName();
loan.setUser(userService.getUniqueUserByEmail(email).get());
loanService.saveLoan(loan);
return new ResponseEntity<>(loan, HttpStatus.CREATED);
}
@GetMapping(value = "/getAllLoans")
public ResponseEntity<?> getMyLoans() {
String email = SecurityContextHolder.getContext().getAuthentication().getName();
return new ResponseEntity<>(new MsgDataResponse("All Loans", loanService.getLoansByUserEmail(email)),
    HttpStatus.OK);
}
}

```

ErrorControllerImpl.java

```

package com.example.loan_backend.controller;
import javax.servlet.http.HttpServletRequest;
import org.springframework.boot.web.servlet.error.ErrorController;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import io.swagger.v3.oas.annotations.Hidden;
@Hidden
@RestController
public class ErrorControllerImpl implements ErrorController {
    @RequestMapping("/error")
    public void handleError(HttpServletRequest request) throws Throwable {
        if (request.getAttribute("javax.servlet.error.exception") != null) {
            throw (Throwable) request.getAttribute("javax.servlet.error.exception");
        }
    }
}

```

AuthController.java

```

package com.example.loan_backend.controller;
import com.example.loan_backend.AccountRoles;
import com.example.loan_backend.CustomUserDetails;
import com.example.loan_backend.models.User;
import com.example.loan_backend.repositories.UserRepository;
import com.example.loan_backend.request.LoginRequest;
import com.example.loan_backend.request.SignupRequest;
import com.example.loan_backend.response.LoginResponse;
import com.example.loan_backend.response.MsgDataResponse;
import com.example.loan_backend.services.CustomUserDetailsService;
import com.example.loan_backend.util.JwtUtil;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RequestMapping("/auth")
@RestController
public class AuthController {
    @Autowired

```

```

private UserRepository userRepository;
@Autowired
private AuthenticationManager authenticationManager;
@Autowired
private CustomUserDetailsService customUserDetailsService;
@Autowired
private PasswordEncoder passwordEncoder;
private JwtUtil jwtUtil;
@PostMapping("/signup")
public ResponseEntity<?> signup(@Valid @RequestBody SignupRequest sr) {
    User user = new User(sr);
    if (userRepository.existsByEmailIgnoreCase(user.getEmail()))
        throw new BadCredentialsException("Account Already exists");
    user.setRole(AccountRoles.ROLE_USER);
    user.setPassword(passwordEncoder.encode(sr.password));
    userRepository.save(user);
    return new ResponseEntity<>(new MsgDataResponse("Account Created Successfully", null), HttpStatus.CREATED);
}
@PostMapping("/signin")
public ResponseEntity<?> signin(@Valid @RequestBody LoginRequest request) {
    authenticationManager
        .authenticate(new UsernamePasswordAuthenticationToken(request.email, request.password));
    CustomUserDetails user = customUserDetailsService.loadUserByUsername(request.email);

    String jwt = jwtUtil.generateToken(user);
    return new ResponseEntity<>(new LoginResponse(jwt, user.getUser()), HttpStatus.OK);
}
}

```

repositories files

UserRepository.java

```

package com.example.loan_backend.repositories;
import com.example.loan_backend.models.User;
import java.util.Optional;
import java.util.UUID;
import org.springframework.data.repository.CrudRepository;
public interface UserRepository extends CrudRepository<User, UUID> {
    public Optional<User> findByEmailIgnoreCase(String email);
    public Iterable<User> findAllByRole(String role);
    public boolean existsByEmailIgnoreCase(String email);
    public Iterable<User> findAllByFirstnameStartingWithIgnoreCaseAndRole(String pattern, String roles);
    public Iterable<User> findAllByLastnameStartingWithIgnoreCaseAndRole(String pattern, String roles);
    public Iterable<User> findAllByEmailStartingWithIgnoreCaseAndRole(String pattern, String roles);
}

```

LoanRepository.java

```

package com.example.loan_backend.repositories;
import com.example.loan_backend.models.Loan;
import java.util.UUID;
import org.springframework.data.repository.CrudRepository;
public interface LoanRepository extends CrudRepository<Loan, UUID> {
    public Iterable<Loan> findAllByStatus(String s);
}

```

response files

ErrorResponse.java

```

package com.example.loan_backend.repositories;
import com.example.loan_backend.models.Loan;

```

```
import java.util.UUID;
import org.springframework.data.repository.CrudRepository;
public interface LoanRepository extends CrudRepository<Loan, UUID> {
    public Iterable<Loan> findAllByStatus(String s);
}
```

LoginResponse.java

```
package com.example.loan_backend.response;
import com.example.loan_backend.models.User;
public class LoginResponse {
    public String jwt;
    public User user;
    public LoginResponse(String jwt, User user) {
        this.jwt = jwt;
        this.user = user;
    }
}
```

MsgDataResponse.java

```
package com.example.loan_backend.response;
public class MsgDataResponse {
    public String msg;
    public Object data;
    public MsgDataResponse(String msg, Object data) {
        this.msg = msg;
        this.data = data;
    }
}
```

Services files

LoanService.java

```
package com.example.loan_backend.services;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.UUID;
import javax.persistence.EntityNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.loan_backend.LoanStatus;
import com.example.loan_backend.models.Loan;
import com.example.loan_backend.models.User;
import com.example.loan_backend.repositories.LoanRepository;
import com.example.loan_backend.repositories.UserRepository;
@Service
public class LoanService {
    @Autowired
    private LoanRepository loanrepo;
    @Autowired
    private UserRepository userRepository;
    // returns list of loans
    public List<Loan> getAllLoans() {
        List<Loan> loans = new ArrayList<Loan>();
        loanrepo.findAll().forEach(loans::add);
        return loans;
    }
    // save loan
    public void saveLoan(Loan l) {
        l.setStatus(LoanStatus.PENDING);
    }
}
```

```

        loanrepo.save(l);
    }
    // get loan by id
    public Optional<Loan> getLoanById(UUID id) {
        return loanrepo.findById(id);
    }
    // delete loan by id
    public void deleteLoanById(UUID id) {
        loanrepo.deleteById(id);
    }
    // Gets all loans of a particular user
    public List<Loan> getLoansByUserEmail(String email) {
        if (email == null)
            throw new EntityNotFoundException("Email should not be null");
        Optional<User> user = userRepository.findByEmailIgnoreCase(email);
        user.orElseThrow(() -> new EntityNotFoundException("user not found"));

        return user.get().getLoans();
    }
    // Gets All PendingLoans
    public List<Loan> getAllPendingLoans() {
        List<Loan> allPendingLoans = new ArrayList<>();
        loanrepo.findAllByStatus(String.valueOf(LoanStatus.PENDING)).forEach(allPendingLoans::add);
        return allPendingLoans;
    }
    // Accept loan(admin)
    public void acceptLoanById(UUID id) {
        Optional<Loan> oldloan = loanrepo.findById(id);
        oldloan.orElseThrow(() -> new EntityNotFoundException("Loan Not Found"));
        Loan newloan = oldloan.get();
        newloan.setStatus(LoanStatus.ACCEPTED);
        loanrepo.save(newloan);
    }
    // Reject Loan(admin)
    public void rejectLoanById(UUID id) {
        Optional<Loan> oldloan = loanrepo.findById(id);
        oldloan.orElseThrow(() -> new EntityNotFoundException("Loan Not Found"));
        Loan newloan = oldloan.get();
        newloan.setStatus(LoanStatus.REJECTED);
        loanrepo.save(newloan);
    }
}

```

UserService.java

```

package com.example.loan_backend.services;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.loan_backend.AccountRoles;
import com.example.loan_backend.models.User;
import com.example.loan_backend.repositories.UserRepository;
@Service
public class UserService {
    @Autowired
    private UserRepository userRepository;
    public List<User> getUsersByFirstName(String pattern) {
        List<User> allUsers = new ArrayList<>();
    }
}

```

```

        userRepository.findAllByFirstnameStartingWithIgnoreCaseAndRole(pattern,
String.valueOf(AccountRoles.ROLE_USER))
            .forEach(allUsers::add);
        return allUsers;
    }
    public List<User> getUsersByLastName(String pattern) {
        List<User> allUsers = new ArrayList<>();
        userRepository.findAllByLastnameStartingWithIgnoreCaseAndRole(pattern,
String.valueOf(AccountRoles.ROLE_USER))
            .forEach(allUsers::add);
        return allUsers;
    }
    public List<User> getUsersByEmail(String pattern) {
        List<User> allUsers = new ArrayList<>();
        userRepository.findAllByEmailStartingWithIgnoreCaseAndRole(pattern, String.valueOf(AccountRoles.ROLE_USER))
            .forEach(allUsers::add);
        return allUsers;
    }
    public Optional<User> getUniqueUserByEmail(String email) {
        return userRepository.findByEmailIgnoreCase(email);
    }
    public List<User> getAllUsers() {
        List<User> allUsers = new ArrayList<>();
        userRepository.findAllByRole(String.valueOf(AccountRoles.ROLE_USER)).forEach(allUsers::add);
        return allUsers;
    }
}

```

CustomerUserDetailsService.java

```

package com.example.loan_backend.services;
import com.example.loan_backend.CustomUserDetails;
import com.example.loan_backend.models.User;
import com.example.loan_backend.repositories.UserRepository;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository userRepository;
    @Override
    public CustomUserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
        Optional<User> user = userRepository.findByEmailIgnoreCase(email);
        user.orElseThrow(() -> new UsernameNotFoundException("User Not Found"));
        return user.map(CustomUserDetails::new).get();
    }
}

```

loan_model

github link:https://github.com/Sumana1008/Capstone/tree/main/loan_model

TYPE OF PROBLEM

This problem is a supervised classification problem because we need to predict whether the “Loan Status” of customers are either “Yes” or “No”.

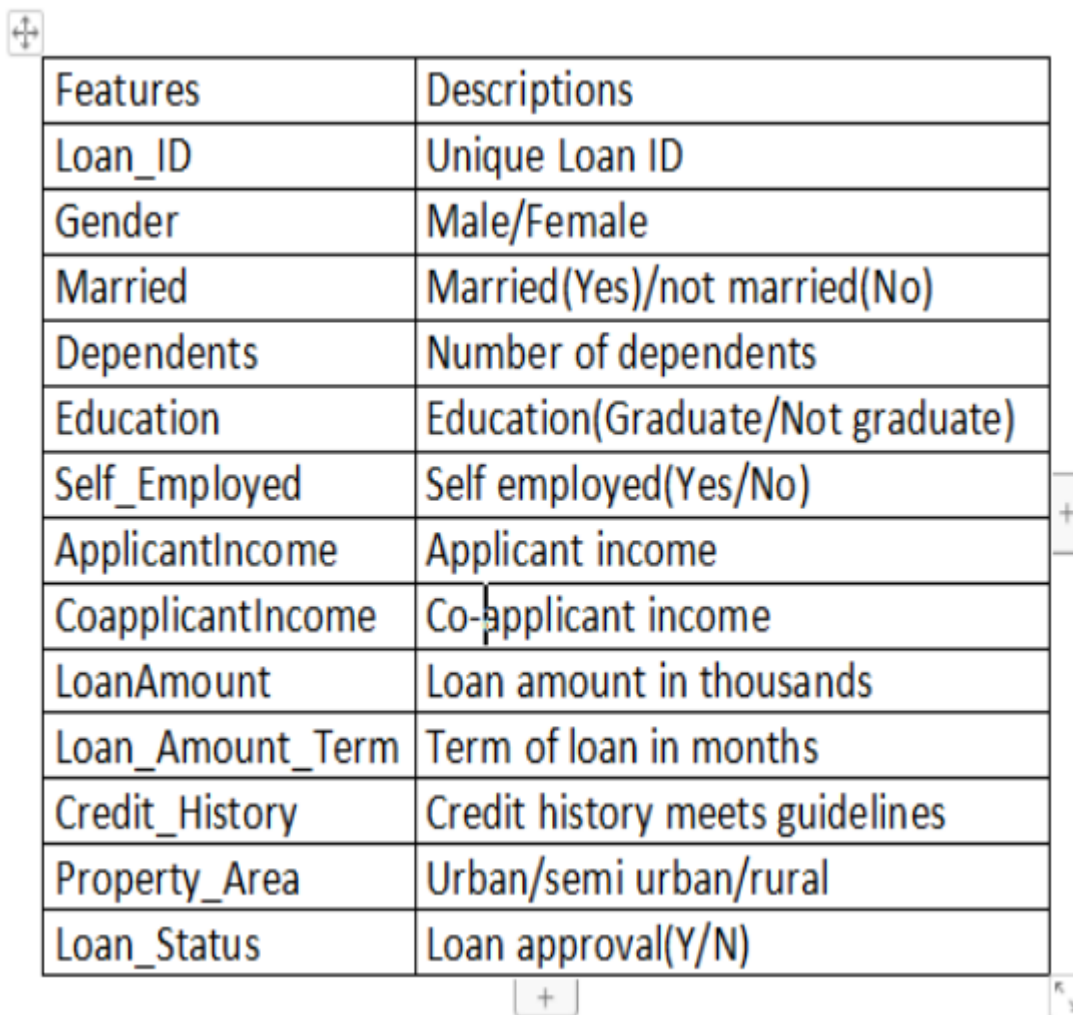
This can be solved with any of the algorithm listed below:

- i. Logistic regression
- ii. Decision tree
- iii. Random Forest

The above-listed algorithm is a few of the algorithms that can be used to solve this problem. What I mean is, the training of the data is not limited to the algorithm listed above.

DESCRIPTION OF COLUMNS

Two data sets are given, one is the training data set and the other is the testing data set. It is better to understand the data set and all the columns in it before trying to solve the problem, to avoid ending up confused. I'll be explaining the columns below;



Features	Descriptions
Loan_ID	Unique Loan ID
Gender	Male/Female
Married	Married(Yes)/not married(No)
Dependents	Number of dependents
Education	Education(Graduate/Not graduate)
Self_Employed	Self employed(Yes/No)
ApplicantIncome	Applicant income
CoapplicantIncome	Co-applicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	Credit history meets guidelines
Property_Area	Urban/semi urban/rural
Loan_Status	Loan approval(Y/N)

DESCRIPTION OF COLUMNS

Two data sets are given, one is the training data set and the other is the testing data set. It is better to understand the data set and all the columns in it before trying to solve the problem, to avoid ending up confused. I'll be explaining the columns below;

Gender:- This is the gender of the applicant, either male or female.

Married:- The marital status of the applicant, either married or not married. If the applicant is married, it is represented with "Yes" and if not, "Not".

Dependents:- Number of persons dependent on the applicant.

Education:- The level of education of the applicant is also a requirement for the approval of the loan. The category is either a graduate or not graduate.

Self_Employed:- This implies that he/she is their own employer i.e they are their boss. The category in the data set is either Yes(Self_employed) or No(not self_employed).

ApplicantIncome:- This is how much the applicant earns. The general assumption is the higher the income, the higher the chance of the applicant paying back.

CoapplicantIncome:- This is the amount the co-applicant earns.

LoanAmount:- This is the loan amount the applicant applied for in thousands. And the assumption is, the higher the amount lesser the chances to pay back.

Loan_Amount_Term:- This is the time take to pay back the loan. It is represented in months.

Credit_History:- This is the record of the applicant's previous loan and if he/she obeyed the rules of payment or not.

Loan_Status:- If the applicant is eligible for a loan, it's yes represented by Y else it's no represented by N.

DATA EXPLORATION

I'm going to work us through the codes and processes in predicting this data.

For this project, I used Jupyter Notebook as the Integrated Development Environment (IDE).

Firstly, all the packages needed to explore the data was imported, before loading the data as shown below:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
train = pd.read_csv("loan train.csv") #to read the train data set
test = pd.read_csv("loan test.csv") #to read the test data set
```

Then I explored the data by checking the first few columns and rows then checked the summary of the statistics, the missing values, number of rows and columns, and listed out all the columns. The summary of the statistics can be check by using:

```
train.describe() #this gives the summary of the statistics
```

```
In [5]: train.describe() #summary of the statistics
```

```
Out[5]:
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2677.500000	0.000000	100.000000	360.000000	1.000000
50%	3612.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

image of the statistics

From this statistics summary, it can be seen that there are some missing values. Explanation of the summary statistics row is below:

Metrics	Description
count	The number of rows in the <u>Dataframe</u>
mean	The mean value for a feature
std	The standard deviation for a feature
min	The minimum value in a feature
25%	The 25 th percentile of a feature
50%	The 50 th percentile of a feature. Note that this is identical to the median.
75%	The 75 th percentile of a feature
max	The maximum value of a feature

DATA CLEANING

After exploring the data set, found out there were some missing values and had to fill it using the median for numerical data and for categorical data used mode instead of mean. Median was chosen because outliers have a significant effect on the mean of data, if there are a lot of outliers it is safe to say that the mean is not the right way to go. It turned out there were a lot of outliers, so I used the median and mode to fill the missing values.

To check missing value

```
train.isnull().sum()
```

```
In [8]: train.isnull().sum()

Out[8]: Loan_ID          0
        Gender           13
        Married          3
        Dependents       15
        Education         0
        Self_Employed     32
        ApplicantIncome    0
        CoapplicantIncome  0
        LoanAmount        22
        Loan_Amount_Term   14
        Credit_History     50
        Property_Area      0
        Loan_Status        0
        dtype: int64
```

Snippet code to show how the missing value was filled

```
train["LoanAmount"].fillna((train["LoanAmount"].median()), inplace=True)
```

```
train["Self_Employed"].fillna((train["Self_Employed"].mode()[0]), inplace=True)
```

```
In [24]: train.isnull().sum()

Out[24]: Loan_ID          0
        Gender           0
        Married          0
        Dependents       0
        Education         0
        Self_Employed     0
        ApplicantIncome    0
        CoapplicantIncome  0
        LoanAmount        0
        Loan_Amount_Term   0
        Credit_History     0
        Property_Area      0
        Loan_Status        0
        dtype: int64
```

Now, it can be seen that the data is clean.

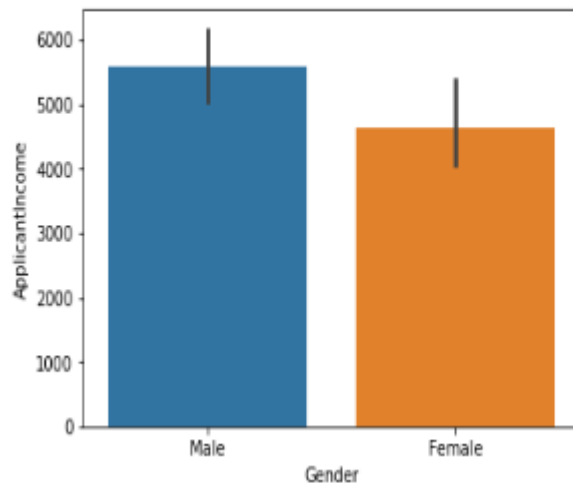
DATA VISUALIZATION

After cleaning the data set, I did the value count and then visualize the data.

```
train["Gender"].value_counts()
```

```
sns.barplot(x="Gender", y="ApplicantIncome", data=train)
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x2900e2ebbe0>
```



And my conclusion after visualizing the data is:

- We can infer that percentage of married people who have got their loan approved is higher compared to non- married people.
- The percentage of applicants with either 0 or 2 dependents have higher loan approval.
- The percentage of applicants who are graduates have higher loan approval than the ones who are not graduates.
- Even after the data analysis, there is still no unique factor to determine loan status.

FEATURE ENGINEERING

After which, columns with categorical data like gender, self_employed, etc were converted to numerical data as shown below:

```
cleanup_num = {"Loan_Status": {"Y": 1, "N": 0},
               "Gender": {"Male": 1, "Female": 0},
               "Married": {"Yes": 1, "No": 0},
               "Self_Employed": {"Yes": 1, "No": 0},
               "Education": {"Graduate": 0, "Not Graduate": 1} }train.replace(cleanup_num, inplace=True)
```

MODELLING

After doing the EDA(Exploratory data analysis), it's time to model/ train the data. Firstly, imported all the necessary packages for modeling the data, for this project three different algorithms(LogisticRegression, DecisionTreeClassifier, and RandomForestClassifier) were used.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=1)
```

After which, the model was defined using:

```
model=LogisticRegression()
```

Then save the model

```
pickle.dump(model,open('model.pkl','wb'))
```

Next, evaluate the model by checking the accuracy score.

```

scoring = "accuracy"

score = cross_val_score(model, train_data, target, cv=k_fold, n_jobs=1, scoring = scoring)

print(score)

round(np.mean(score)*100, 2)

```

After training the data, it was discovered that the LogisticRegression model was the best fit for the data because its accuracy was the highest(81.12%).

It was then used to fit the test data set and the prediction came out successful.

```

model.fit(train_data, target)

test_data = test.drop("Loan_ID", axis=1).copy()

prediction = model.predict(test_data)

```

In this loan model we are using logistic regression to predict this problem because logistic regression is used to calculate or predict the probability of binary event occurring. Generally it is used for prediction and classification problems. After training the data, we found logistic regression model was best fit for the data because its accuracy was the highest 80.14% compared to decision tree(69.23%) and Random forest(77.53%).

app.py:

```

from flask import Flask, request, jsonify, render_template
from flask_mysqldb import MySQL
import yaml
import random
import string
from mysql.connector import connect, Error

from mysql.connector.cursor import MySQLCursor
import numpy as np
import pickle; pickle.HIGHEST_PROTOCOL
app = Flask(__name__)
db = yaml.full_load(open('E:\CodeBase\Loan\db.yaml'))
#cnx = mysql.connector.connect(user=db['mysql_user'],
password=db['mysql_password'], host=db['mysql_host'], database=db['mysql_db'])
def create_connection():
    try:
        conn = connect(
            host=db['mysql_host'],
            user=db['mysql_user'],
            password=db['mysql_password'],

```

```

        database=db['mysql_db']
    )
    return conn
except Error as e:
    print(e)
    return None

def create_cursor(conn):
    try:
        cursor = conn.cursor()
        return cursor
    except Error as e:
        print(e)
        return None

def insert_data(cursor, id,amt, emimonth , status,email):
    try:
        sql = "INSERT INTO loan_data(id,amount,emimonths,status,user)
VALUES(%s,%s,%s,%s,%s)"
        val = (id,amt, emimonth, status,email)
        cursor.execute(sql, val)
    except Error as e:
        print(e)

def fetch_data(cursor,param):
    sql='SELECT * FROM loan_data where user = %s'
    val=[(param)]
    cursor.execute(sql,val)
    rows = cursor.fetchall()
    return rows

def commit_changes(conn):
    conn.commit()

def close_connection(cursor, conn):
    cursor.close()
    conn.close()

with open(r"E:\CodeBase\Loan\loanmodel.pkl", 'rb') as f:
    model = pickle.load(f)

@app.route("/predict/<param>", methods=["POST"])
def predict(param):
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    feature=np.array(int_features)
    prediction = model.predict(final_features)
    print(prediction);
    output = round(prediction[0], 2)
    #prediction_text = "{}".format(output)
    ide = ''.join([random.choice(string.ascii_letters + string.digits) for n
in range(32)])
    print(feature[3])
    print(prediction.item())
    if feature[3] == 240:
        month = 6
    elif feature[3] == 360:
        month = 12

```



```

    else:
        month = 0

    amt = feature[2].item()
    if prediction.item() == 1:
        stat = "ACCEPTED"
    elif prediction.item() == 2:
        stat = "REJECTED"
    else:
        stat = "NULL"

    conn = create_connection();
    cursor=create_cursor(conn);
    insert_data(cursor,ide,amt,month,stat,param)
    commit_changes(conn)
    close_connection(cursor,conn)
    return 'Sucess'

```

```

@app.route('/data/<param>', methods=['GET'])
def results(param):
    print(type(param))
    conn = create_connection()
    cursor = conn.cursor()
    rows = fetch_data(cursor,param)
    cursor.close()
    conn.close()
    return jsonify(rows)

```

```

if __name__ == "__main__":
    app.run(debug=True)

```

index.html

```

<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></scrip
t>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></sc
ript>
  <link rel="stylesheet" type="text/css" href="static/style.css">
  <title>Loan Predict</title>
</head>
<body>
<div class="container main">
  <div class="jumbotron" id="holder">
    <br>
    <center><b></b></center>

```

```

    <br>
    <br>
    <center><h1 class='main_heading'> Loan Approval Prediction</h1></center>
    <p>Machine Learning Algorithm to predict whether a loan would be approved or
not.</p>
    <p>Note that this Model This Web Based Application is processed by a is
81.12% accurate</p>
    <p><b>Please fill the form below</b></p>
    <br>
    <form class="form-horizontal" action="/predict"method="post">
        <div class="form-group">
            <div class="col-sm-10">
                <input class="form-control" type="text" name="ApplicantIncome"
placeholder="Enter your current income" required="required" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-sm-10">
                <input class="form-control" type="text" name="CoapplicantIncome"
placeholder="Enter Your guarantor's income" required="required" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-sm-10">
                <input class="form-control" type="text" name="LoanAmount"
placeholder="Enter the amount you wish to borrow" required="required" />
            </div>
        </div>

        <div class="form-group">
            <div class="col-sm-10">
                <select class="form-control" type="number" name="Loan_Amount_Term"
required="required">
                    <option value="0">Please choose a loan term</option>
                    <option value="240">6 months</option>
                    <option value="360">1 year</option>
                </select>
            </div>
        </div>
        <div class="form-group">
            <div class="col-sm-10">
                <select class="form-control" type="number" name="Credit_History"
required="required">
                    <option value="0">Do you have a credit history? </option>
                    <option value="1">yes</option>
                    <option value="0">no</option>
                </select>
            </div>
        </div>
        <div class="form-group">
            <div class="col-sm-10">
                <select class="form-control" type="number" name="Self_Employed"
required="required">
                    <option value="0">Self Employed</option>
                    <option value="1">yes</option>
                    <option value="0">no</option>
                </select>
            </div>
        </div>
    </div>

```

```

    <div class="form-group">
      <div class="col-sm-10">
        <select class="form-control" type="number" name="Property Area"
required="required">
          <option value="0">Please select your geographical area</option>
          <option value="2">urban</option>
          <option value="1">semiurban</option>
          <option value="0">rural</option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-10">
        <select class="form-control" type="number" name="Married"
required="required">
          <option value="0">Select your marital status</option>
          <option value="1">yes</option>
          <option value="0">no</option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-10">
        <select class="form-control" type="number" name="Education"
required="required">
          <option value="0">Please choose your level of
education</option>
          <option value="0">Graduate</option>
          <option value="1">not Graduate</option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-10">
        <select class="form-control" type="number" name="Gender"
required="required">
          <option value="0">Please choose your gender</option>
          <option value="1">male</option>
          <option value="0">female</option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="button btn btn-default">PREDICT</button>
      </div>
    </div>
  </form>
  <br>
  <br>
  {{ prediction_text }}
</div>
</div>
</body>
</html>

```

db.yaml

```

mysql_host: 'localhost'
mysql_user: 'root'
mysql_password: '4445'

```

```
mysql_db: 'loan'
```

Procfile

```
web: gunicorn app:app
```

Requirements.txt

```
Flask==1.1.1  
gunicorn==20.0.4  
itsdangerous==1.1.0  
Jinja2==2.10.1  
MarkupSafe==1.1.1  
Werkzeug==0.15.5  
numpy>=1.9.2  
scipy>=0.15.1  
scikit-learn>=0.18  
matplotlib>=1.4.3  
pandas>=0.19  
flask_mysqldb  
pyyaml~=6.0  
mysql-connector-python
```

Request.py

```
import request  
url = 'http://localhost:5000/results'  
r = request.post(url,json={'ApplicantIncome':4000, 'CoapplicantIncome':2000,  
                           'LoanAmount':600,  
                           'Loan Amount Term':360,'Credit History':1,  
                           'Self Employed':1, 'Property Area':1, 'Married':1,  
                           'Education':0,  
                           'Gender':1 })  
print(r.json())
```

REFERENCES

- [1] <https://codebun.com/loan-management-in-spring-boot-and-hibernate-with-source-code/>
- [2] <https://reactjs.org/>
- [3] <https://stackoverflow.com/questions/60746698/basic-create-react-app-jest-test-explained>
- [4] <https://stackoverflow.com/questions/65396568/react-js-npm-start-shows-failed-to-compile-web-vitals>

BIODATA



Name : Sumana Mupparapu
Mobile Number : 9393946464
Email : sumana.19bce7316@vitap.ac.in
Permanent Address : H-No:5-8-98,Opp:New Bustand,Beside
Raja Xerox,Kamareddy



Name : Kookutla Satwika
Mobile Number : 8688738711
E-mail :satwika.19bce7585@vitap.ac.in
Permanent Address :H-NO : 15-21/1 Vidyanagar , Near
Vijayalaxmi sarees, Kamareddy



Name :Karri Roshini Sri Vyshna Pranathi
Mobile Number :9347286799
E-mail :pranath.19bce7656@vitap.ac.in
Permanent Address:Pittalavemavaram, vadali, peravali,
west Godavari, Andhra Pradesh, 534324.