The Egyptian E Learning University

Faculty of Computer& Information Technology

Quick Loan

GRADUATION PROJECT

PREPARED BY

Beshoy Ayman	20-00378
Sandra Wadee	20-00851
Mena Habeeb	20-01333
Zahir Romany	20-00467
Magy Youhanna	20-00715
Kareem Waseef	20-00043
Fatma Mohamed Hashem	20-00387

SUPERVISED BY

Dr. Rodaina Abdel-salam

Eng. Rofida Abdel-hafez

A Graduation Project Report Submitted for the Partial Fulfillment of the Requirements of the B.SC. Degree

2023-2024

TABLE OF CONTENTS

Abstract

Abstract

With the enhancement of the technology, expand of businesses and thoughts more and more people are applying for the loans. Both for their personal use and for business use.

But due to the limited amount of assets the bank cannot grant loans to each and every person. Finding out those right people is a typical and time-consuming process.

Banks desire to deliver the loan to an individual who can be recompensate the loan on time and can afford maximum profit the bank. So there is a need of a system which could do this analysis and save the banks time and resources.

This can be done using Machine Learning. The objective of this paper is to create a more accurate loan prediction model using machine learning to reduce the risk behind selecting of appropriate people for the loan.

For this we'll mine the previous records of the people to whom the bank has granted loan. Using these records variables and bank loan rules we will train a machine learning model which will predict that a person is eligible for loan or not. We will use

sklearn for our model and train_test_split for splitting the data set into train dataset and test dataset.

Here we are going to use various models like Logistic Regression, Decision Tree(DT) , SVM, Naïve Bayes, Neural Network, xgboost, K Neighbors and RandomForest(RF) so as to fetch more accurate results as the given problem is a supervised classification problem.

Acknowledgement

First and foremost, we thank God Almighty for the blessing of reaching this advanced stage in our education to achieve this result of success.

We put effort into this project. However, this would not have been possible without the support and assistance of many individuals and organizations. I would like to express my sincere thanks to all of them.

Thank you to the Egyptian University for E-Learning and especially the Asyut Center for helping us reach this level of awareness.

We would like to express our gratitude to our Project Supervisor Dr. Rodaina Abdelsalam for giving us the opportunity to lead us and provide invaluable guidance during this project. It was a great honor and privilege for us to work under her supervision.

We are especially indebted to express our heartfelt thanks to Eng. Rofida Abdel -hafez for the guidance, constant supervision, patience, friendship and great sense of humor for all his efforts with us.

Finally, we would like to express our gratitude to everyone who helped us during the graduation project.

Introduction

1. Introduction

Loan Prediction is very helpful for employee of banks as well as for the applicant also.

The aim of this Paper is to Provide quick, immediate and easy way to choose the deserving applicants.

Customer first apply for loan After that company or bank validates the customer Eligibility for loan.

Company or bank wants to automate the loan eligibility process (real time) based on customer Details provided while filling application form. These Details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and Other.

This project has taken the data of previous Customers of various banks to whom on a set of Parameters loan were approved. So the machine learning Model is trained on that record to get accurate results.

Our Main objective of this project is to predict the safety of Loan.

To predict loan safety, the Logistic Regression, Decision Tree(DT) , SVM, Naïve Bayes, Neural Network, xgboost, K Neighbors and RandomForest(RF)Algorithm are used.

1.1. History

Loans are a vital tool for financial transactions, but they come with their own complexities and costs.
Obtaining a loan involves more than just securing funds; it requires careful consideration of terms, interest rates, and repayment schedules.
Experts have been grappling with these challenges for nearly a century.

Loan Origination Processes (LOPs) are procedures through which individuals or businesses apply for and receive loans from financial institutions.
They encompass various stages, from initial application to final approval and disbursement of funds.
LOPs are crucial for streamlining the loan application process, reducing paperwork, and expediting access to credit.

The history of loan origination processes can be traced back to the early 20th century when traditional banks relied heavily on manual paperwork and face-to-face interactions.
As the demand for credit grew, so did the need for more efficient and scalable lending practices.

In the 1970s, with the advent of computers and data processing technologies, banks began to explore automated loan origination systems.
These systems allowed for faster application processing, improved risk assessment, and greater consistency in lending decisions.

The widespread adoption of the internet in the 1990s further revolutionized loan origination processes.
Online applications and digital documentation made it possible for borrowers to apply for loans remotely, reducing the need for physical branch visits and paperwork.

Today, loan origination processes continue to evolve with advancements in artificial intelligence, machine learning, and big data analytics.

These technologies enable financial institutions to analyze vast amounts of data, assess creditworthiness more accurately, and offer personalized loan products tailored to individual needs.

Despite these advancements, challenges remain, such as ensuring data security, maintaining regulatory compliance, and providing a seamless user experience.
Nevertheless, the ongoing innovation in loan origination processes promises to make access to credit more convenient, efficient, and inclusive for borrowers around the world.

1.2. Motivation

Loan approval is a very important process for banking organizations.
The system approved or reject the loan applications.
Recovery of loans is a major contributing parameter in the financial statements of a bank.
It is very difficult to predict the possibility of payment of loan by the customer.
Using Machine learning we predict the loan approval.

1.3. Problem statement

Any financial institution would like to automate the loan eligibility process (in real time) based on the customer details provided while filling the online application form.
These details are gender, marital status, number of educational dependents, income, loan amount, credit history, etc. To automate this process, they may go through the trouble of identifying customer segments, those who are eligible for the loan amount so that they can target those customers specifically.
Here we have provided a partial dataset that the project seeks to screen loan applicants based on these factors.
Automation with training and testing datasets improves efficiency.
A machine learning model trained on the former predicts loan eligibility based on the latter.

1.4. Problem solutions

- **Data Exploration**

  Examine the provided partial data set and understand the data distribution for each feature Identify any missing values, outliers, or patterns in the data.

- **Data preprocessing**

  Clean the data by handling missing values, encoding categorical variables

- **Model selection**

  Choose the appropriate machine learning algorithm for binary classification.
  Experiment with different models to find the one that performs best for your specific data set

- **Model training**

  Split the data set into training and test sets.
  Train the selected machine learning model on the training set, using loan eligibility as the target variable.

- **Model evaluation**

  Evaluate the model's performance on the test set using appropriate metrics such as precision , precision , recall , F1 score , and area under the ROC curve.

1.5.Project Phases

- Research
- Analysis
- Prototyping
- Implementation
- Development

Overall Description

2. Overall description

This section will give you an overview about the website.
This website makes it easy for users to take out a quick loan without human intervention and saves them time and effort. Once their data is entered, they are responded to, whether by accepting their request for the loan if they meet the criteria, or by refusal it if one of the conditions set for taking the loan is not found.

2.1. website perspective

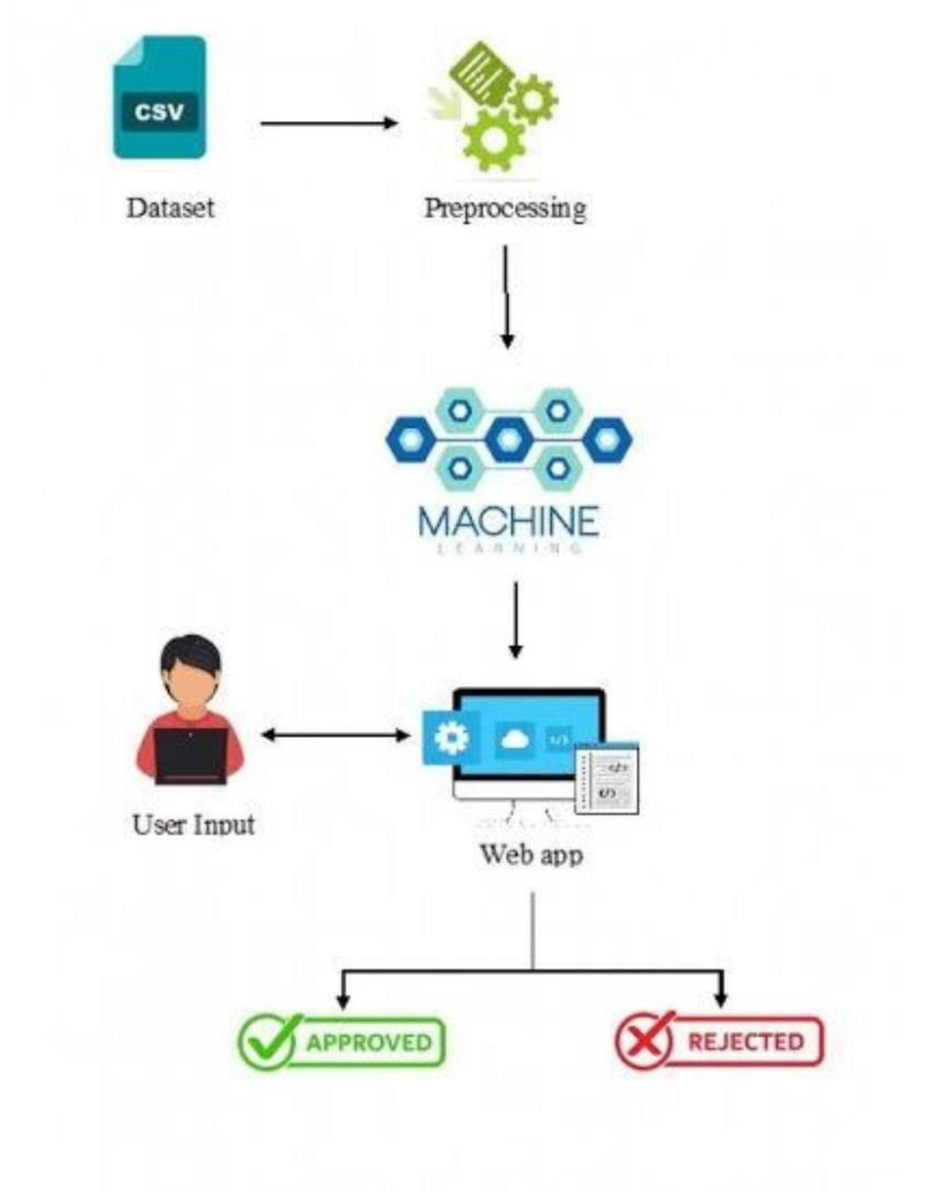
The website is easy for the user to use and makes it easy for him to get a loan quickly from his place.
First The user performs a search by opening the web browser and writing the bank address in the web browser bar.
Once you access the bank's website, the bank's home page will appear, which contains information about the bank
and loan details.
The user then requests an application for the loan he needs and enters his data, such as name, date of birth, nationality, current address, monthly income, marital status, city, and mobile number to obtain the loan.

Then The Ai analyzes the data, compares it to the data set , compares it to the most accurate algorithm, and then approves or rejects it.

After training, testing, and confirming the data entry.

After that, if the loan is approved, the loan will be calculated and its installments will be determined over a
specific period of time.

2.2. website functions



2.3. User characteristics

1. The customer:

Opens the application, uses the application to choose an operation he/she wants to do and then the application will perform it.

2. He/she can be:

a. Minimum age: 21 years, maximum age: 60 years
b. You have a bank account
c. Has a certificate of deposit
d. The loan holder must have a commercial register

2.4. Use Cases

Nowadays everyone uses the mobile typically all the time.

1. Required documents: You may be asked to upload documents such as a valid ID photo, proof of income (such as an account statement or tax statement), and other documents related to the type of loan requested.
2. Submission of the necessary documents: During the bank, the customer can upload the documents required to process the personal loan request, such as personal ID, income certificates, etc.
3. Required documents: You may be asked to upload documents such as a valid ID photo, proof of income (such as an account statement or tax statement), and other documents related to the type of loan requested.
4. Customers can apply for loans online anywhere and anytime, without the need to visit the bank branch in person. This saves customers time and effort.
5. Employing artificial intelligence to make decisions: We use artificial intelligence techniques to analyze loan

applicants and make quick decisions regarding approval or rejection. This can be useful for customers who are looking for a quick answer to their questions.

It is important that customers understand the terms and conditions of necessities and benefits clearly before attending, and must ensure that they are coordinated by reliable and licensed banks.

Therefore, user can use this website to:

1. Currently, artificial intelligence is employed in personal loans to provide a better and more effective experience for users and to improve the decision-making process regarding loan approval.
2. Artificial intelligence can analyze the financial data provided by the previous applicants and estimate their ability to repay the loan. This analysis can be based on information such as income, debt, credit history and other financial variables to provide an accurate estimate of the borrower's ability to repay the loan.
3. By using artificial intelligence, it is possible to speed up the processing of loan applicants. The intelligent system can quickly analyze the information and decide on the approval or rejection based on the pre-defined criteria,

which reduces the time required for the users to get a response on their requests.

4. Artificial intelligence can improve the user experience by providing a smoother application process and precise guidance to users during the application process. The intelligent system can provide instructions and advice to users about the required documents and necessary information, which helps them to complete the application correctly and effectively.
5. By using artificial intelligence, it is possible to reduce human errors that may occur during the process of analyzing data and making decisions. The intelligent system can work with high precision and overcome possible human factors such as prejudice, stress or wrong decisions

2.5. Assumptions and dependencies

When employing artificial intelligence to accept and reject requests without human intervention, you may face some of the following potential problems:

a) Error in analysis: The system makes incorrect decisions based on an inaccurate analysis of the statements, which leads to the rejection of eligible requests or the approval of inappropriate requests.

b) deficiency in data: The lack of data available to the system leads to inaccurate or unbalanced decisions.

c) Data exposure: It exposes personal and financial data that are used in decision-making processes to the risk of leakage or intrusion, which puts the privacy of customers at risk.

d) Difficulty in communication: It is difficult for customers to communicate with the bank or the party that uses artificial intelligence to deal with any problems or inquiries they have.

e) Corruption and bribery: The occurrence of falsification or falsification of documents, or the fact that employees take bribes from clients or companies to speed up the loan approval process, which is a type of corruption.

f) Miscalculation of ability to pay: providing loans to customers who do not have the ability to pay, which leads to financial problems for both the bank and the customer.

It is necessary to consider these problems and take the necessary measures to reduce them and ensure that the system works actively and fairly without endangering the privacy of customers.

Pages Description

3. Pages description

You can use the personal loan website in the following ways:

Submitting a loan application: You will be asked to provide some required personal and financial information, such as your name, address, income, and other financial obligations.

Determine the amount and terms: You determine the amount of money you want to borrow and the loan repayment period. The site will also show you the specific loan terms, such as interest rate, monthly payments, and other terms.

Submission and Approval: After completing and submitting your application, the site will evaluate the application and information provided. If your application is approved, you will receive an approval notification and loan details, such as the amount borrowed and repayment terms.

Loan repayment: You will be obligated to repay the loan according to the specified terms. Premiums are typically charged through monthly payments on a set schedule.

You must read and understand the terms and conditions of the loan before agreeing to it. Make sure you understand the total amount of the loan, including interest and associated fees, and ensure you are able to repay the loan on the scheduled dates.

You should be careful and responsible when using a personal loan website, and make sure to choose a loan option that suits your needs and ability to repay.

3.1. Logo

The "QuickLoan" logo appears at the top left of the page.

3.2. Web site name

The site name "QuickLoan" appears next to the logo.

3.3. Navigation bar

The site contains a navigation bar at the top of the page that includes the following sections:

Home: This page leads to the main page of the site.

About Us: This page provides information about QuickLoan and its services.

Business Loans: This page provides information about business loans offered by QuickLoan.

Calculator: This tool allows you to calculate the cost of your loan.

Contact Us: This page provides information on how to contact QuickLoan.

Apply Now: This page leads to a loan application form.

3.4. Page title

The page title “Start or Grow Your Own Business” appears below the navigation bar.

3.5. Promotional photo

A large promotional image appears in the middle of the page showing a man and woman working in their office.

3.6. Promo text

A promotional text appears below the promotional image that says:

Get what matters

Start or grow your own business

Get your dream in simple steps and without any guarantees!

Get a loan from your income

Apply now

3.7. Business loan calculator

The Business Loan Calculator appears on the right of the page. This calculator allows you to calculate the cost of your loan based on the amount you need, the term of the loan, and the interest rate.

3.8. Apply now button

The Apply Now button appears below the Business Loan Calculator. This button leads to a loan application form.

3.9. Contact information

QuickLoan contact information appears at the bottom of the page.

Comments

- The website is simple and easy to use.
- The site focuses on providing business loans.
- The website is available in Arabic.

Conclusion

QuickLoan is a website that offers business loans to small and medium-sized businesses. The website is easy to use and provides clear information about the business loans offered by the company.

Related Work

4. Related work

4.1. Compare apps

4.1.1. Bankrate website

www.bankrate.com: Bankrate is a comprehensive financial resource website that provides tools and information for various financial products, including loans.It offers loan calculators, rate comparisons, and educational articles.

4.1.2. SoFi

www.sofi.com: SoFi is an online lending platform that specializes in student loan refinancing, personal loans, mortgages, and other financial products. It also offers career services and financial planning tools.

4.2. Bank loan approval prediction using machine learning techniques

4.2.1. Theoneste Ndayisenga Registration Number: 215042343

The dissertation by Theoneste Ndayisenga explores the use of machine learning techniques to predict bank loan approval and default, with a focus on the Bank of Kigali's data. The research concludes that Gradient Boosting is the most effective model for predicting loan default, followed by Xgboosting, while Decision Trees, Random Forest, and Logistic Regression perform poorly. The author recommends financial institutions to adopt machine learning for loan prediction, highlighting the significance of credit scores in assessing creditworthiness. The document includes sections such as a declaration, approval sheet, dedication, acknowledgment, abstract, table of contents, list of figures and tables, and a conclusion with recommendations for further study.

Ashwini S. Kadam, Shraddha R. Nikam, Ankita A. Aher, Gayatri V. Shelke, Amar S. Chandgude

The research paper titled "Prediction for Loan Approval Using Machine Learning Algorithm" underscores the importance of predicting loan approval and the challenges in forecasting customer loan repayment. It advocates for the use of machine learning algorithms, particularly emphasizing the Naïve Bayes model for loan forecasting and reducing non-performing assets. The document covers data collection, cleaning, and performance evaluation as crucial steps in loan prediction, referencing SVM and Naïve Bayes algorithms for safety prediction. The paper discusses related works, citing research papers like "Loan Approval Prediction Based on Machine Learning Approach" and "Exploring the Loan Sanctioning Process." Additionally, it addresses the loan approval process across various financial institutions and highlights the aim of analyzing customer credibility to determine loan approval.

4.2.2. Ashwini S. Kadam, Shraddha R. Nikam, Ankita A. Aher

Amruta S. Aphale Prof. Dr. Sandeep. R. Shinde

The academic paper, authored by Amruta S. Aphale and Prof. Dr. Sandeep. R. Shinde, discusses the application of machine learning algorithms in predicting loan approval for cooperative banks. It introduces two main categories of machine learning: supervised learning, which involves predicting features using a target variable, and unsupervised learning, used for clustering entities without a target variable. The paper aims to extract crucial information from loan data using various machine learning algorithms, facilitating informed decisions on loan requests. Additionally, it explores the use of classification algorithms, such as neural networks and Naive Bayes, to predict loan defaulters and improve credit risk assessment in the banking sector

Yash Diwate, Prashant Rana, Pratik Chavan (Students)

The document is a research paper titled "Loan Approval Prediction Using Machine Learning," authored by Yash Diwate, Prashant Rana, and Pratik Chavan, published in the International Research Journal of Engineering and Technology (IRJET) in May 2021. The study focuses on predicting loan defaulters to reduce Non-Performing Assets (NPA) in banks. The authors employ a logistic regression model, utilizing Kaggle data and considering variables like personal attributes and checking account information. The paper is structured into sections covering data collection, comparison of AI models, training the framework, and testing, with keywords including Support Vector Machine (SVM), Machine Learning, and Loan Prediction.

4.2.3. Amruta S. Aphale Prof. Dr. Sandeep. R. Shinde

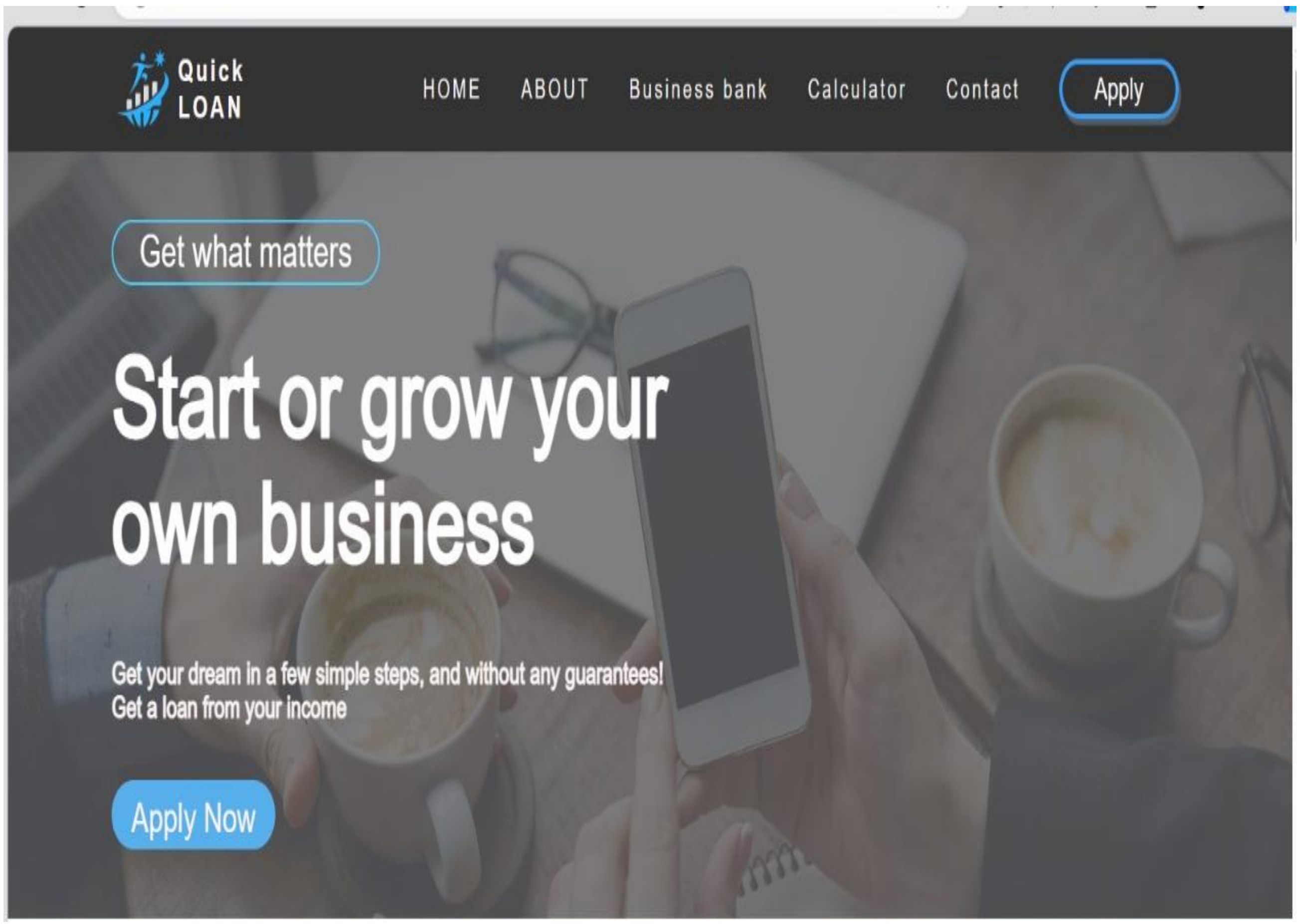
R. Shinde discusses the application of machine learning algorithms in predicting loan approval for cooperative banks. It offers two main categories of machine learning: supervised learning, which involves predicting features using a target variable, and unsupervised learning, used to cluster entities without a target variable. The paper aims to extract important information from loan data using various machine learning algorithms, thus facilitating informed decisions on loan applications. In addition, it explores the use of classification algorithms, such as neural networks and Naive Bayes, to predict loan defaulters and improve credit risk assessment in the banking sector.

4.2.4. Yash Diwate, Prashant Rana, Pratik Chavan (Students)

The document is a research paper titled “Loan Approval Prediction Using Machine Learning”, the study focuses on predicting loan defaulters to reduce non-performing assets (NPA) in banks. The authors use a logistic regression model, using Kaggle data and looking at variables such as personality traits and checking account information. The paper is organized into sections covering data collection, AI model comparison, framework training, and testing, with keywords including support vector machine (SVM), machine learning, and loan prediction.

Design

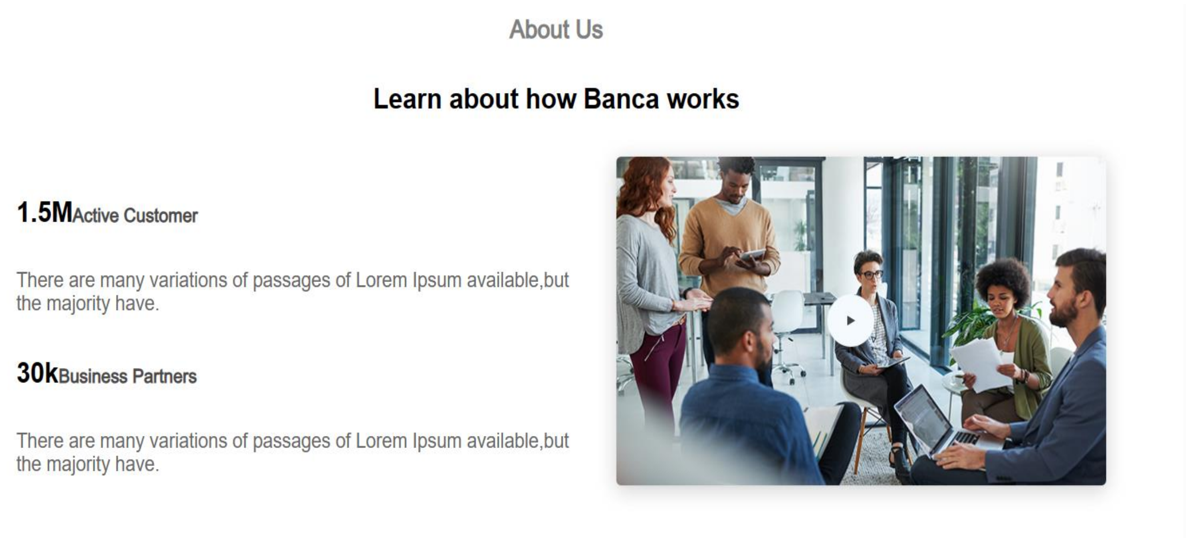
5.1. Home screen



Start or grow your own business

Get your dream in a few simple steps, and without any guarantees!
Get a loan from your income

5.2.About Us



About Us

Learn about how Banca works

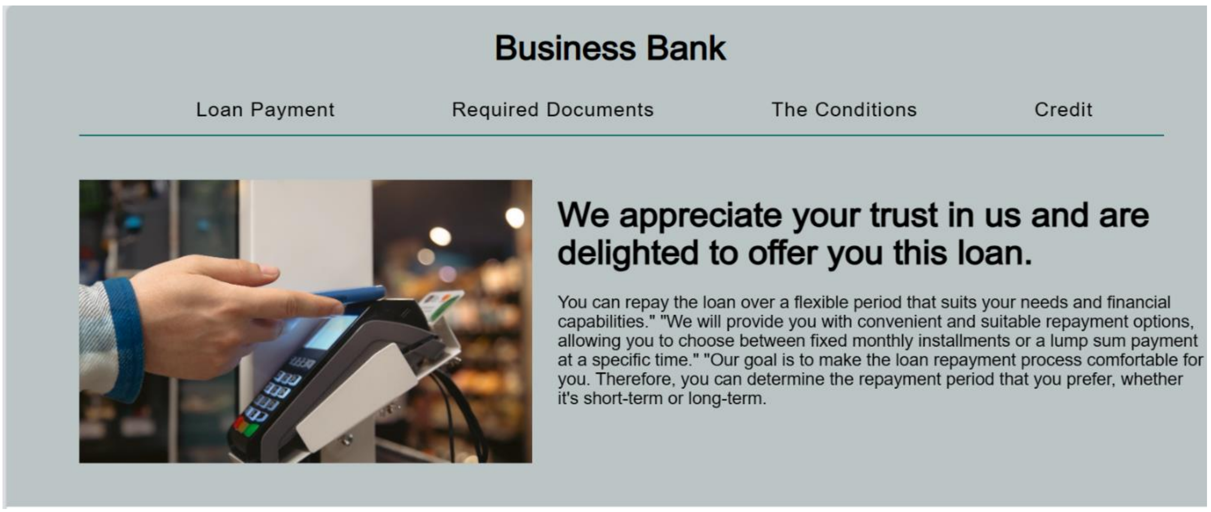
1.5MActive Customer

There are many variations of passages of Lorem Ipsum available,but the majority have.

30kBusiness Partners

There are many variations of passages of Lorem Ipsum available,but the majority have.

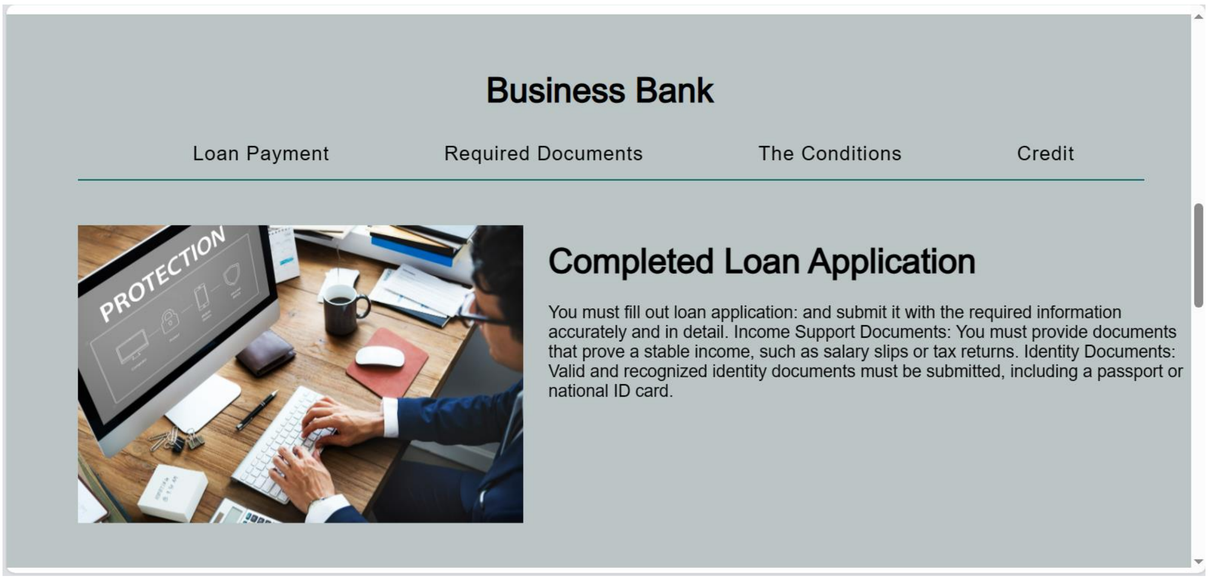
5.3.1 Business Bank



We appreciate your trust in us and are delighted to offer you this loan.

You can repay the loan over a flexible period that suits your needs and financial capabilities." "We will provide you with convenient and suitable repayment options, allowing you to choose between fixed monthly installments or a lump sum payment at a specific time." "Our goal is to make the loan repayment process comfortable for you. Therefore, you can determine the repayment period that you prefer, whether it's short-term or long-term.

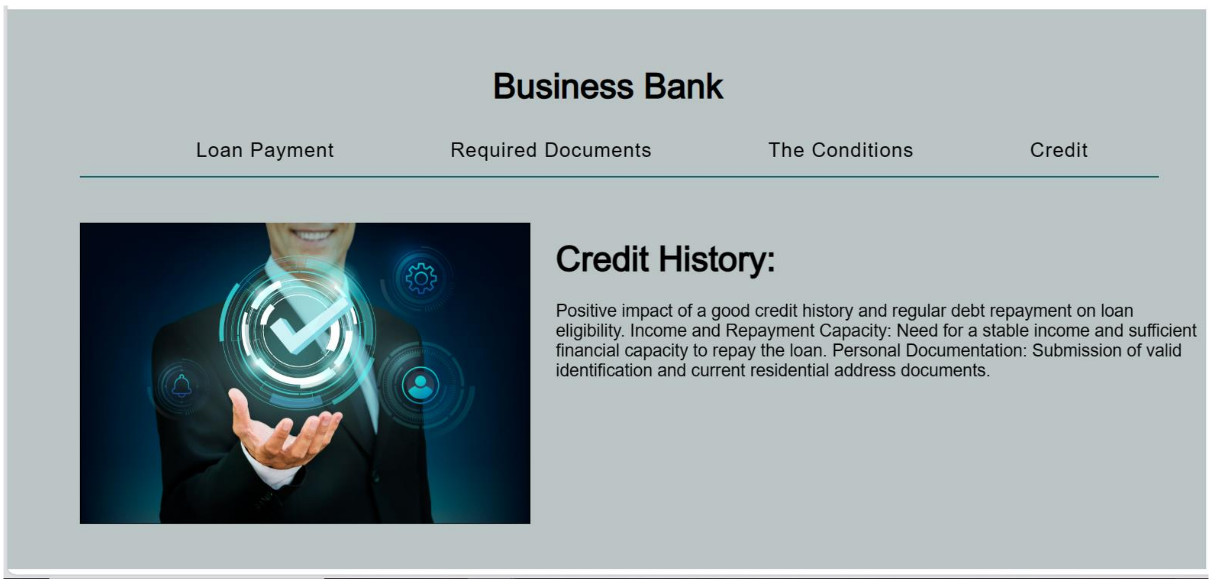
5.3.2 Required Documents



Completed Loan Application

You must fill out loan application: and submit it with the required information accurately and in detail. Income Support Documents: You must provide documents that prove a stable income, such as salary slips or tax returns. Identity Documents: Valid and recognized identity documents must be submitted, including a passport or national ID card.

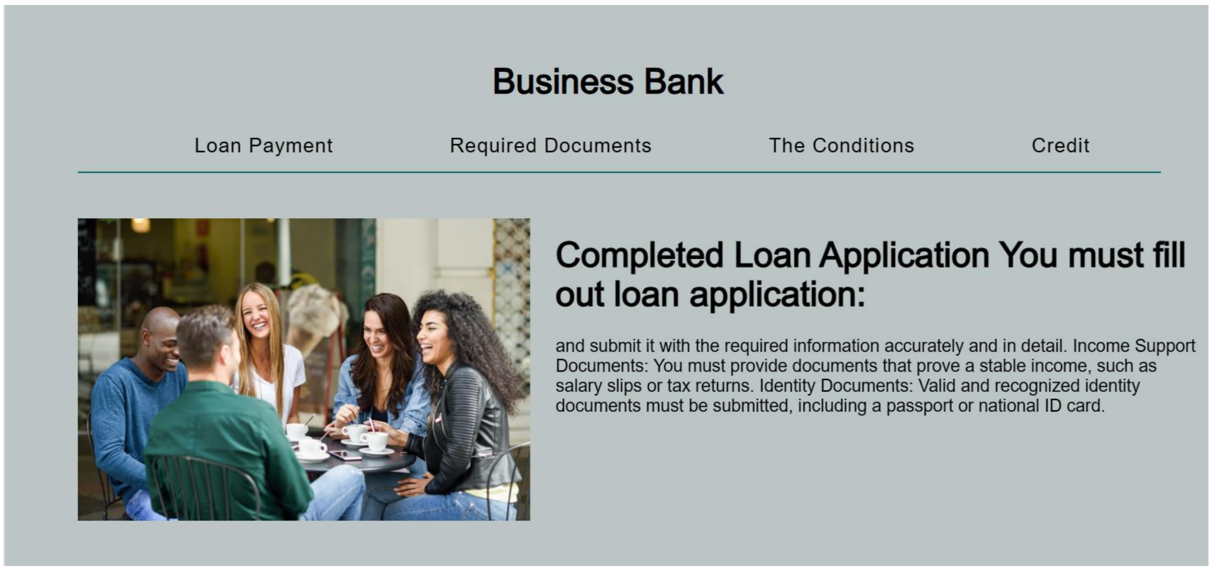
5.3.3. The Conditions



Credit History:

Positive impact of a good credit history and regular debt repayment on loan eligibility. Income and Repayment Capacity: Need for a stable income and sufficient financial capacity to repay the loan. Personal Documentation: Submission of valid identification and current residential address documents.

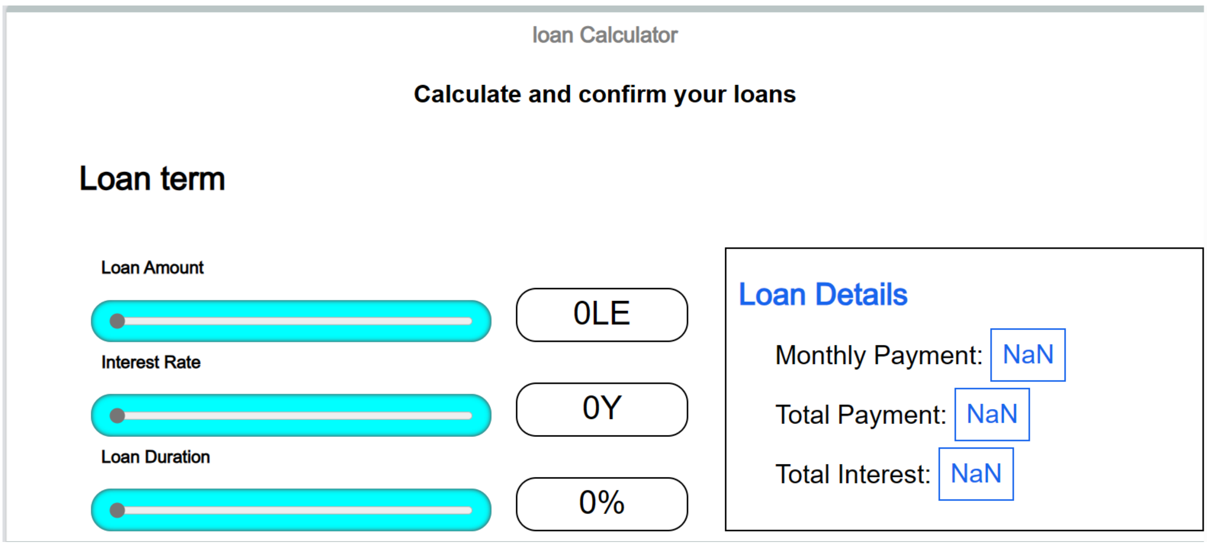
5.3.4. Credit



Completed Loan Application You must fill out loan application:

and submit it with the required information accurately and in detail. Income Support Documents: You must provide documents that prove a stable income, such as salary slips or tax returns. Identity Documents: Valid and recognized identity documents must be submitted, including a passport or national ID card.

5.4. loan Calculator


loan Calculator

Calculate and confirm your loans

Loan term

Loan Amount

0LE

Interest Rate

0Y

Loan Duration

0%

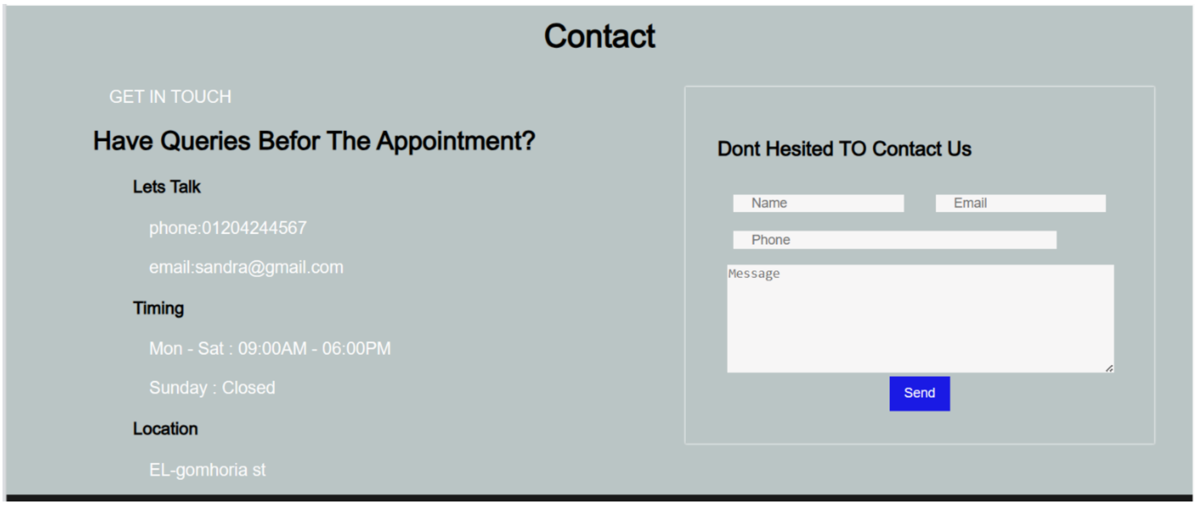
Loan Details

Monthly Payment: NaN

Total Payment: NaN

Total Interest: NaN

5.5.Contact


Contact

GET IN TOUCH

Have Queries Befor The Appointment?

Lets Talk

phone:01204244567

email:sandra@gmail.com

Timing

Mon - Sat : 09:00AM - 06:00PM

Sunday : Closed

Location

EL-gomhoria st

Dont Hesited TO Contact Us

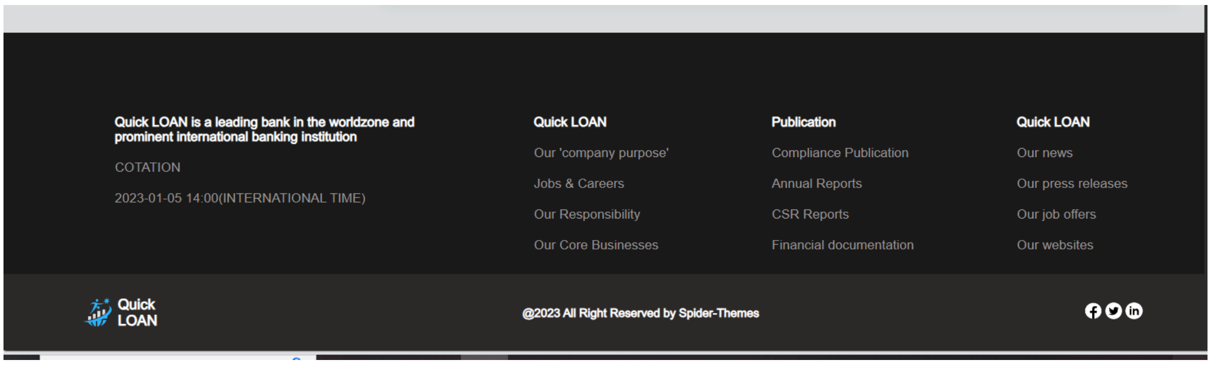
Name | Email

Phone

Message

Send

5.6.Footer



Quick LOAN is a leading bank in the worldzone and prominent international banking institution

COTATION

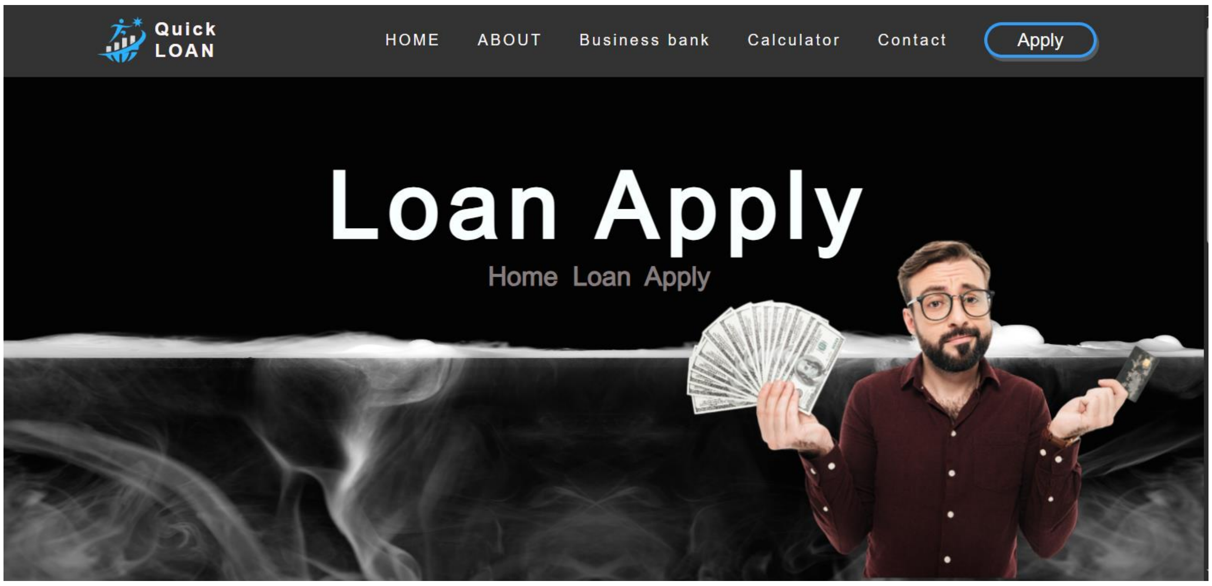
2023-01-05 14:00(INTERNATIONAL TIME)

Quick LOAN	Publication	Quick LOAN
Our 'company purpose'	Compliance Publication	Our news
Jobs & Careers	Annual Reports	Our press releases
Our Responsibility	CSR Reports	Our job offers
Our Core Businesses	Financial documentation	Our websites

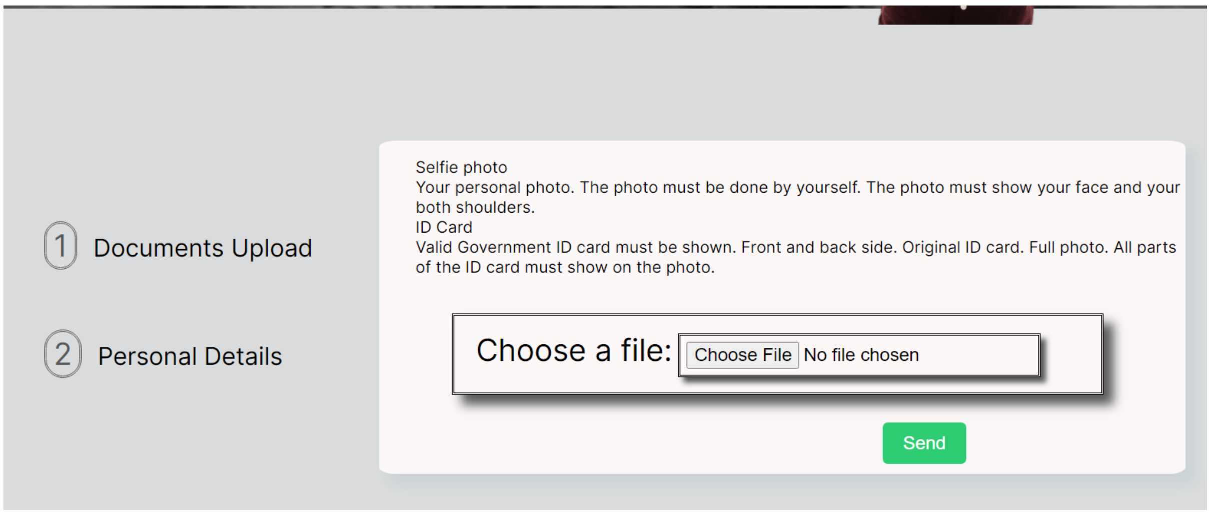
Quick LOAN

@2023 All Right Reserved by Spider-Themes

5.7. Apply



5.8. Apply

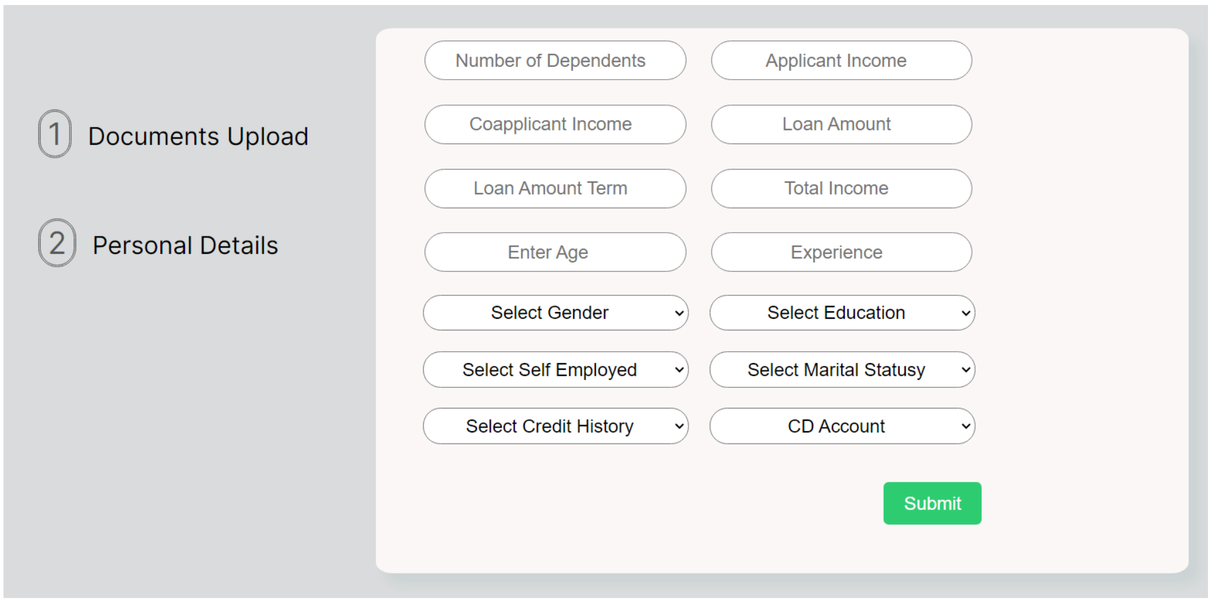

1 Documents Upload

2 Personal Details

Selfie photo
Your personal photo. The photo must be done by yourself. The photo must show your face and your both shoulders.
ID Card
Valid Government ID card must be shown. Front and back side. Original ID card. Full photo. All parts of the ID card must show on the photo.

Choose a file: Choose File No file chosen

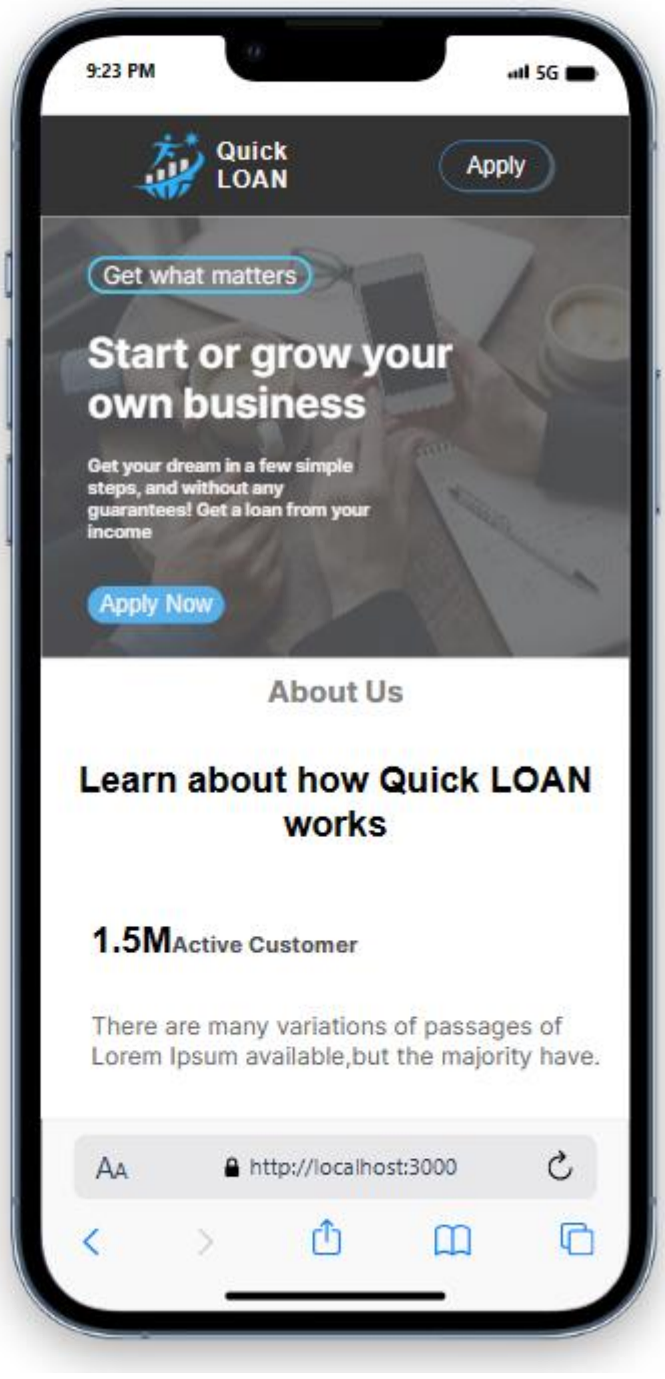
5.9 Apply


1 Documents Upload

2 Personal Details

Number of Dependents	Applicant Income
Coapplicant Income	Loan Amount
Loan Amount Term	Total Income
Enter Age	Experience
Select Gender	Select Education
Select Self Employed	Select Marital Statusy
Select Credit History	CD Account

5.10. Responsive


9:23 PM

5G

Quick LOAN

Apply

Get what matters

Start or grow your own business

Get your dream in a few simple steps, and without any guarantees! Get a loan from your income

Apply Now

About Us

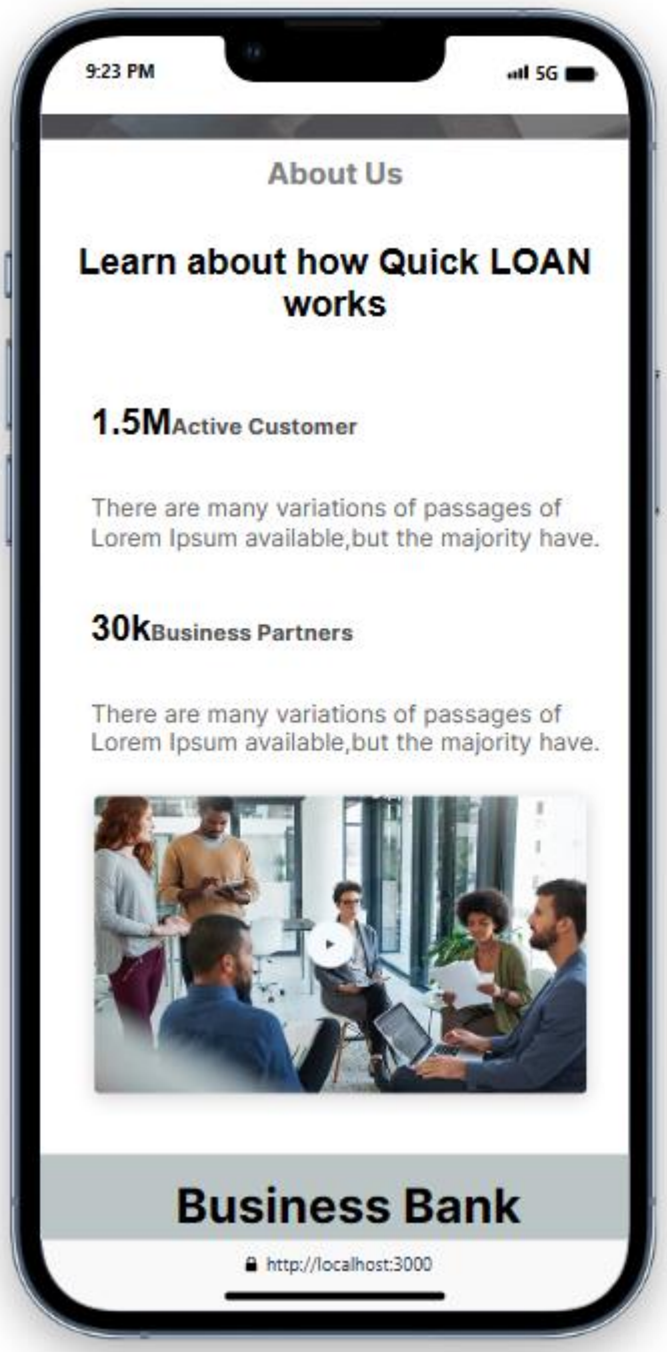
Learn about how Quick LOAN works

1.5MActive Customer

There are many variations of passages of Lorem Ipsum available,but the majority have.

AA

http://localhost:3000



About Us

Learn about how Quick LOAN works

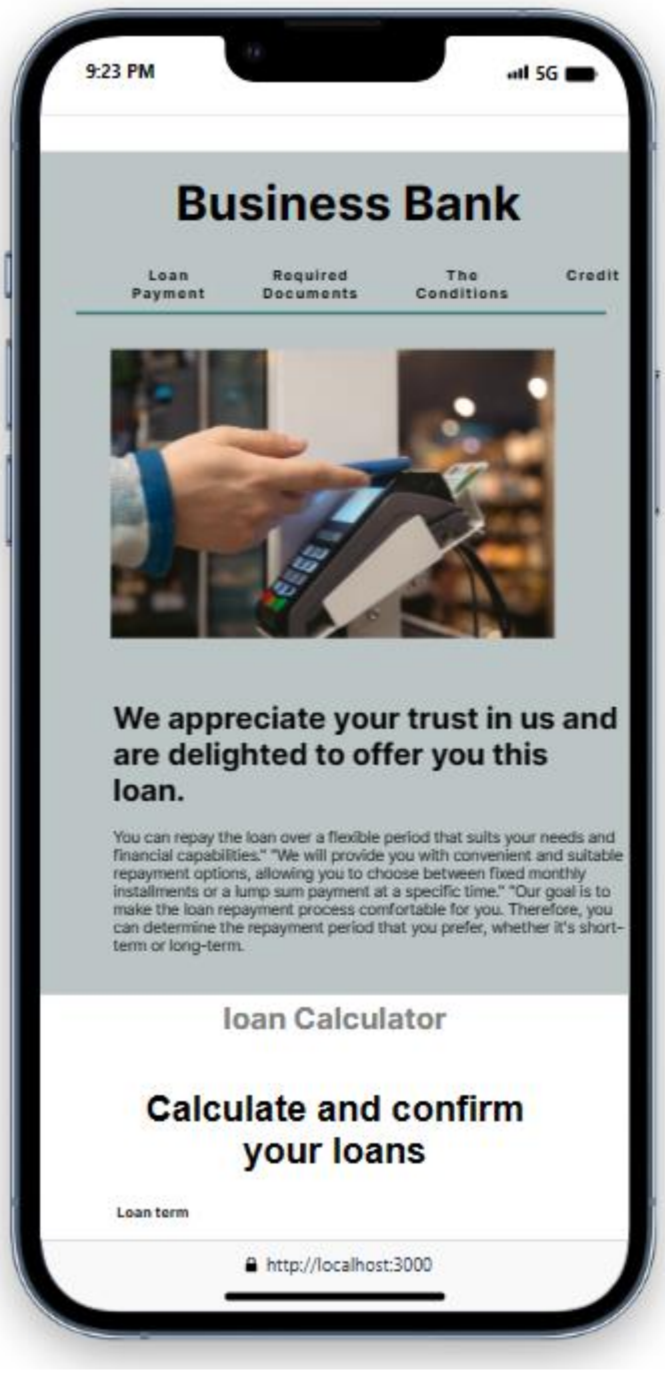
1.5MActive Customer

There are many variations of passages of Lorem Ipsum available,but the majority have.

30kBusiness Partners

There are many variations of passages of Lorem Ipsum available,but the majority have.

Business Bank



9:23 PM

5G

Business Bank

Loan Payment | Required Documents | The Conditions | Credit

We appreciate your trust in us and are delighted to offer you this loan.

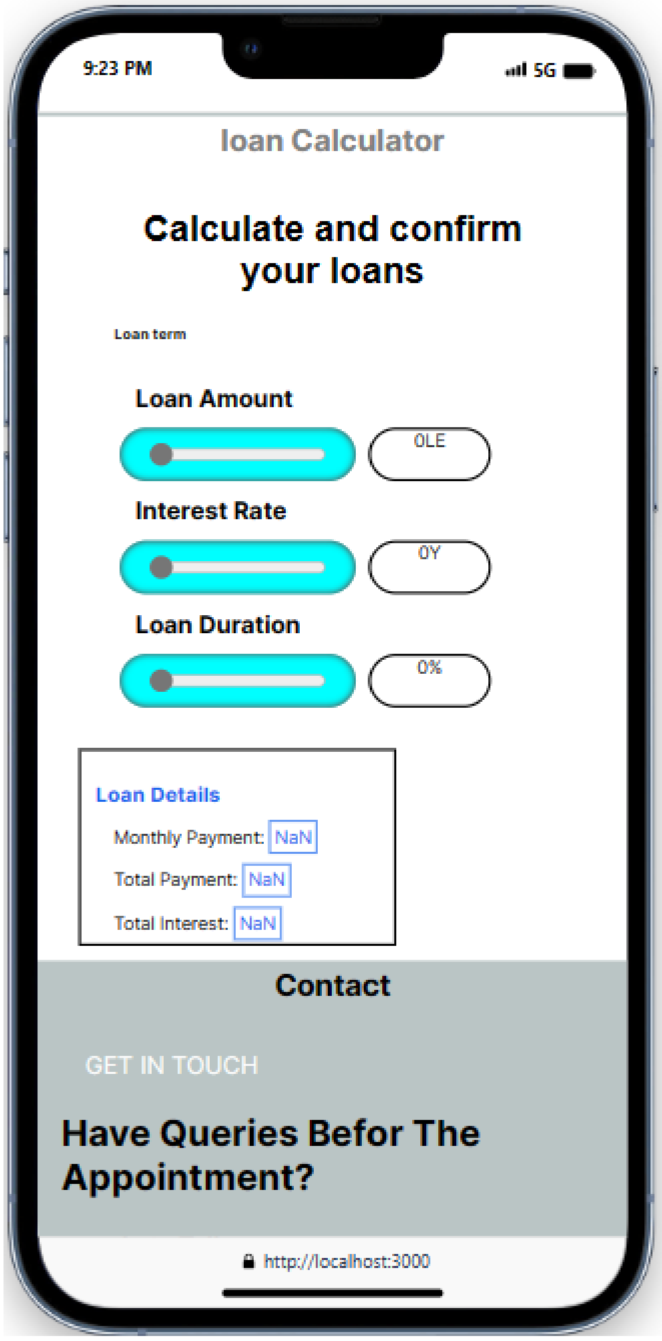
You can repay the loan over a flexible period that suits your needs and financial capabilities." "We will provide you with convenient and suitable repayment options, allowing you to choose between fixed monthly installments or a lump sum payment at a specific time." "Our goal is to make the loan repayment process comfortable for you. Therefore, you can determine the repayment period that you prefer, whether it's short-term or long-term.

loan Calculator

Calculate and confirm your loans

Loan term

http://localhost:3000



9:23 PM

5G

loan Calculator

Calculate and confirm your loans

Loan term

Loan Amount

0LE

Interest Rate

0Y

Loan Duration

0%

Loan Details

Monthly Payment: NaN

Total Payment: NaN

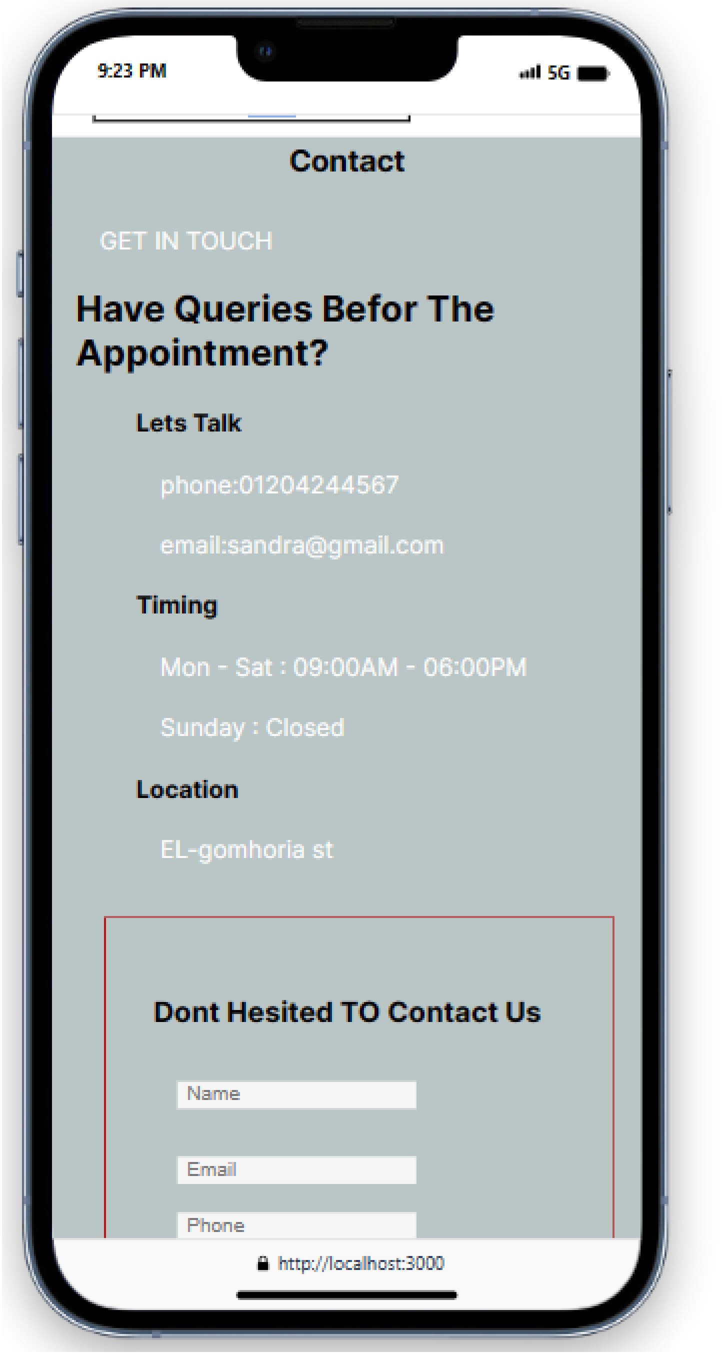
Total Interest: NaN

Contact

GET IN TOUCH

Have Queries Befor The Appointment?

http://localhost:3000



9:23 PM

5G

Contact

GET IN TOUCH

Have Queries Befor The Appointment?

Lets Talk

phone:01204244567

email:sandra@gmail.com

Timing

Mon - Sat : 09:00AM - 06:00PM

Sunday : Closed

Location

EL-gomhoria st

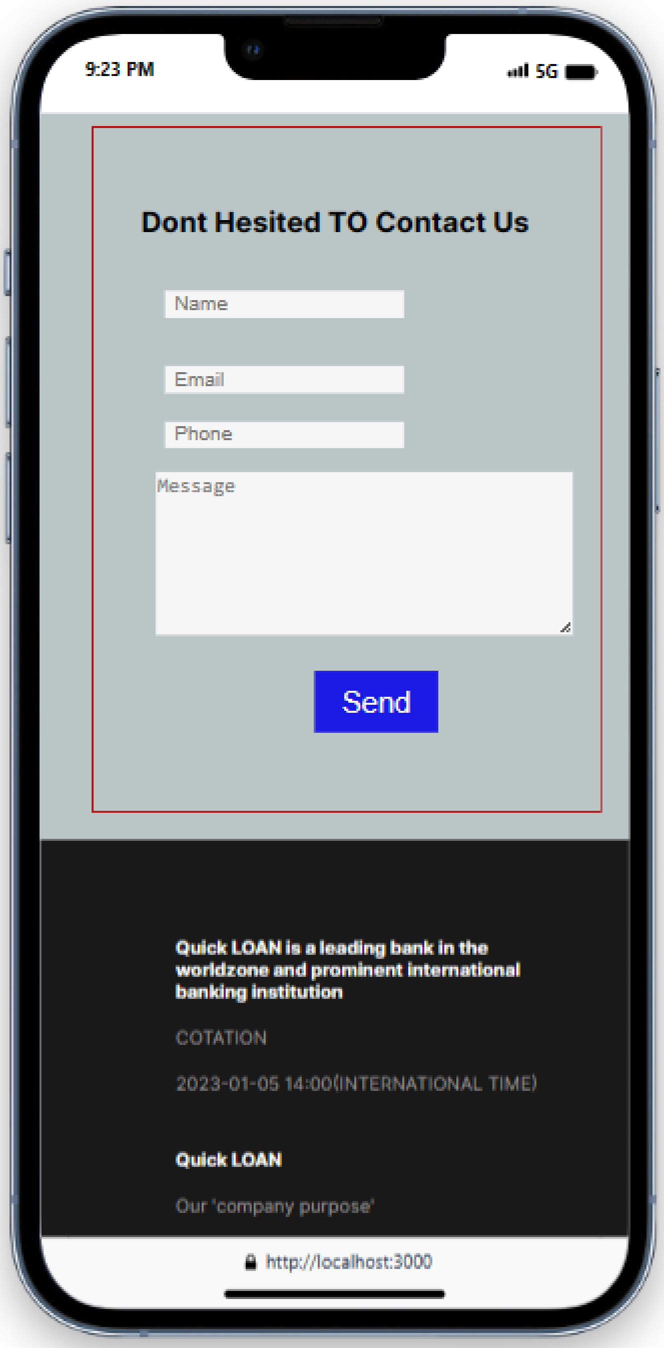
Dont Hesited TO Contact Us

Name

Email

Phone

http://localhost:3000



9:23 PM

5G

Dont Hesited TO Contact Us

Name

Email

Phone

Message

Send

Quick LOAN is a leading bank in the worldzone and prominent international banking institution

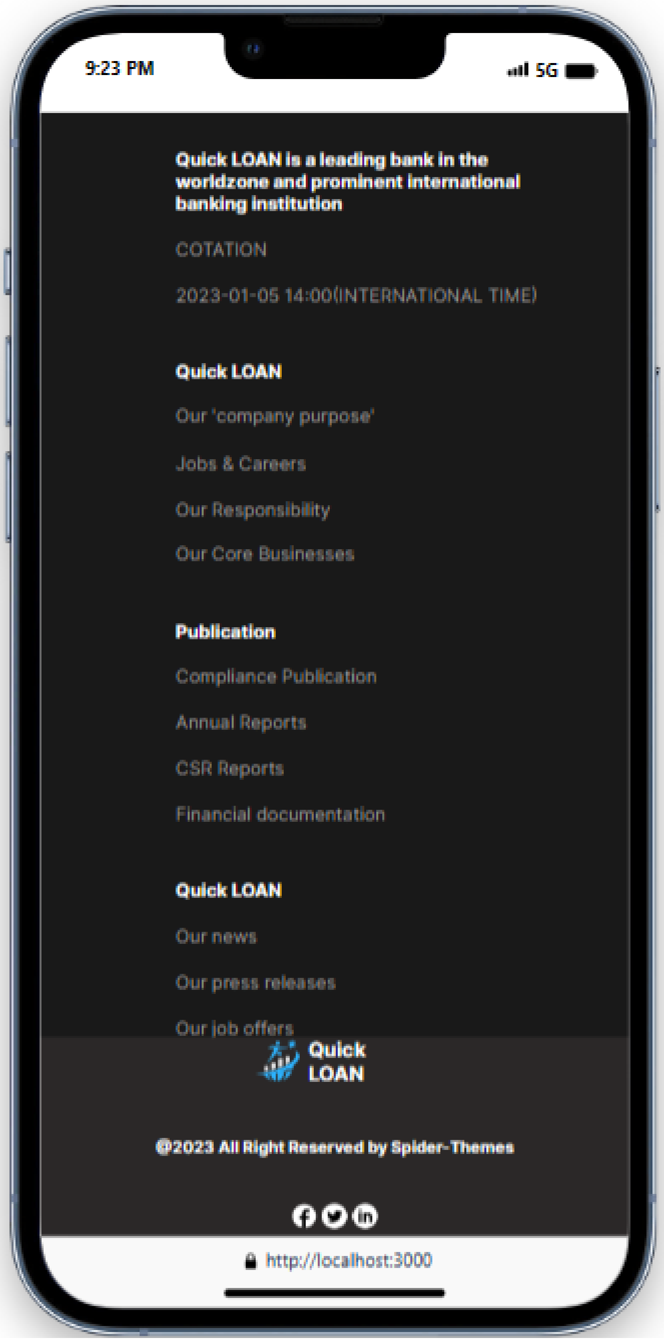
COTATION

2023-01-05 14:00(INTERNATIONAL TIME)

Quick LOAN

Our 'company purpose'

http://localhost:3000



9:23 PM

5G

Quick LOAN is a leading bank in the worldzone and prominent international banking institution

COTATION

2023-01-05 14:00(INTERNATIONAL TIME)

Quick LOAN

Our 'company purpose'

Jobs & Careers

Our Responsibility

Our Core Businesses

Publication

Compliance Publication

Annual Reports

CSR Reports

Financial documentation

Quick LOAN

Our news

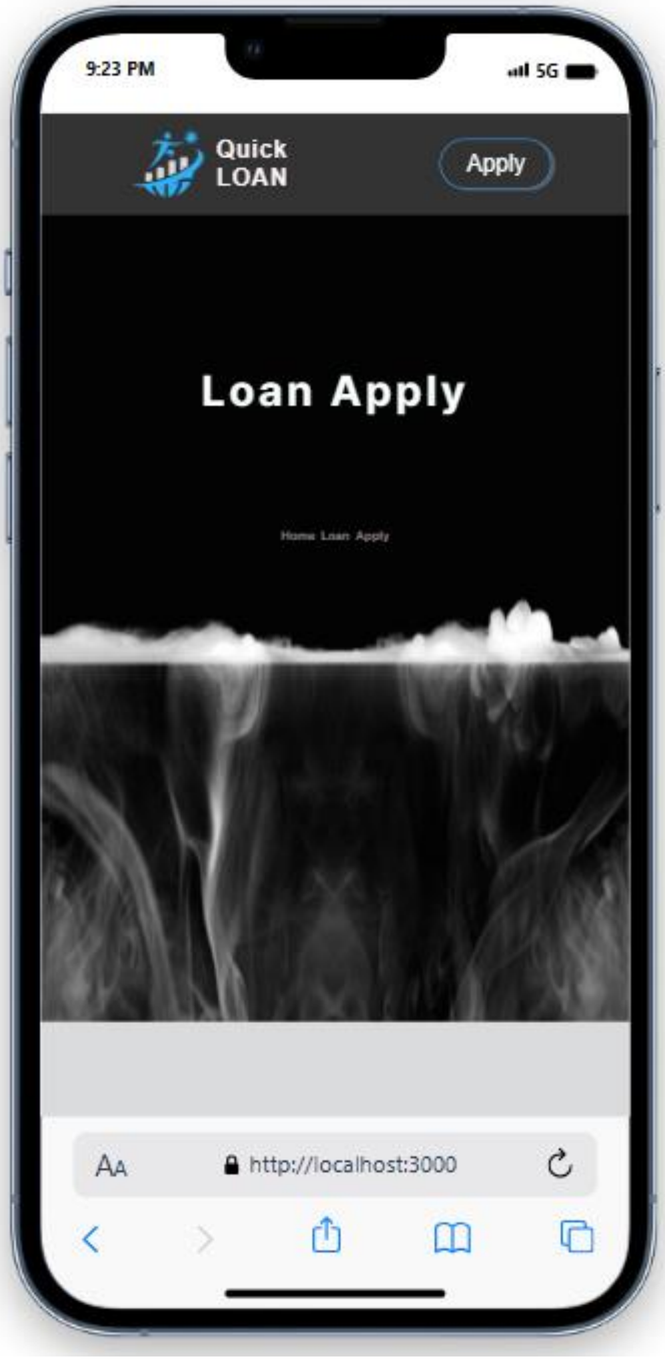
Our press releases

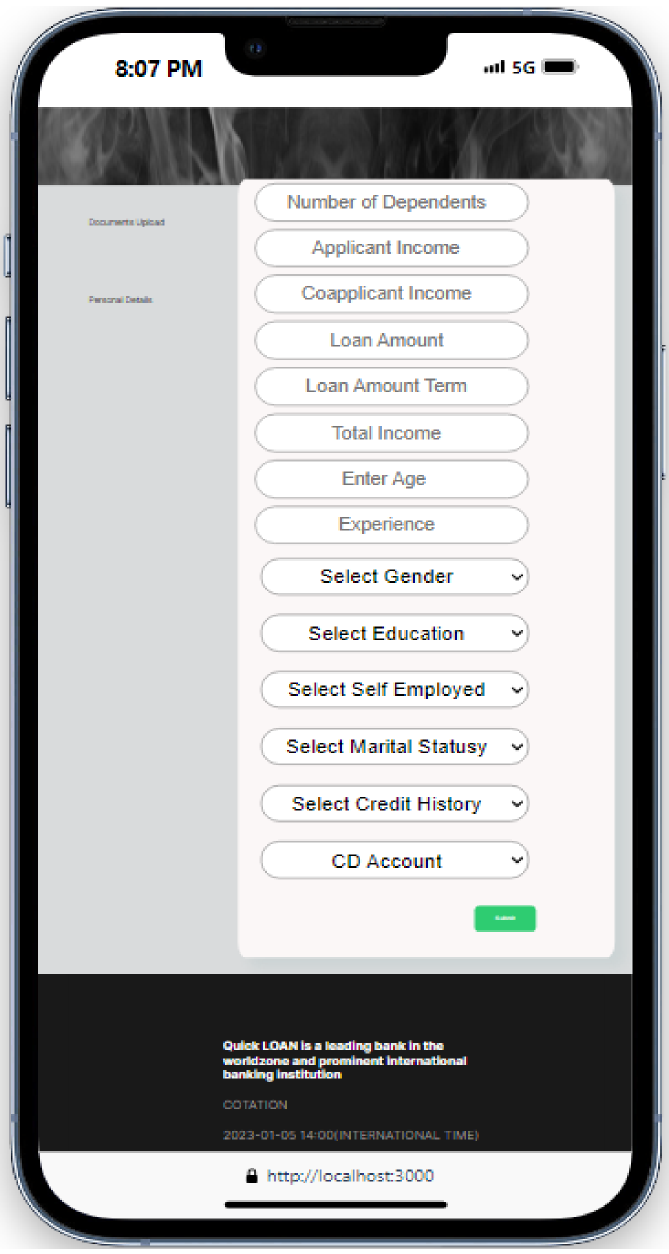
Our job offers

Quick LOAN

@2023 All Right Reserved by Spider-Themes

http://localhost:3000


9:23 PM
5G
Quick LOAN
Apply
Loan Apply
Home Loan Apply
http://localhost:3000



8:07 PM 5G

Documents Upload

Personal Details

Number of Dependents

Applicant Income

Coapplicant Income

Loan Amount

Loan Amount Term

Total Income

Enter Age

Experience

Select Gender

Select Education

Select Self Employed

Select Marital Statusy

Select Credit History

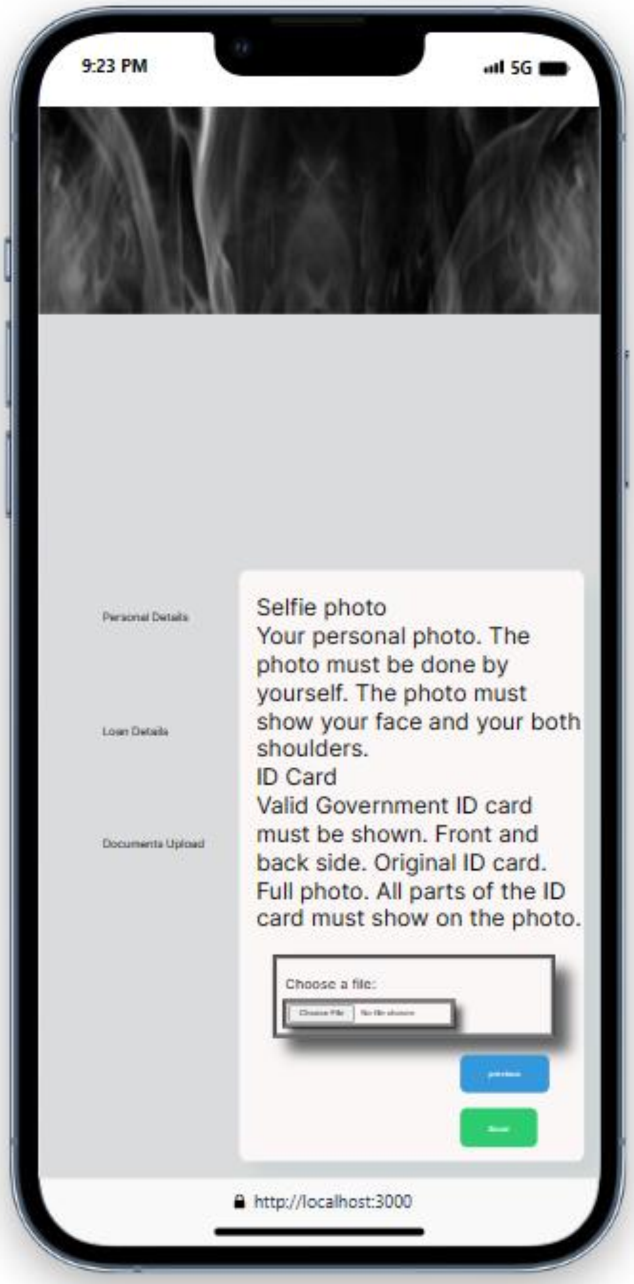
CD Account

Quick LOAN is a leading bank in the worldzone and prominent international banking institution

COTATION

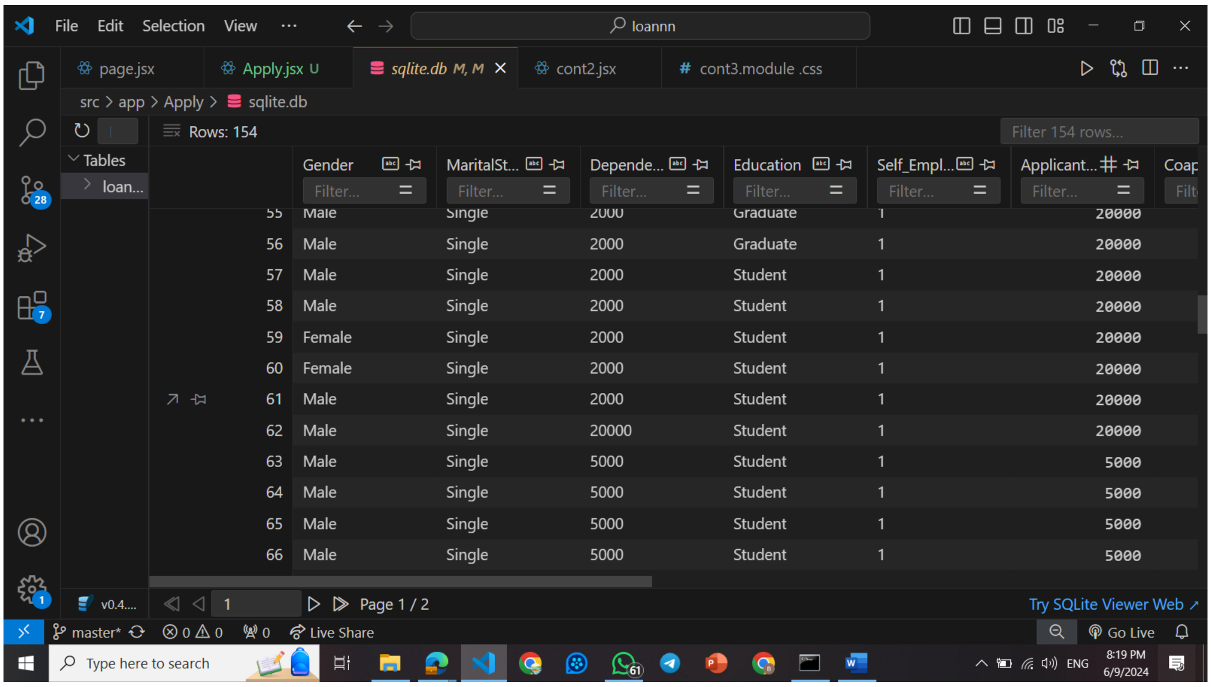
2023-01-05 14:00(INTERNATIONAL TIME)

http://localhost:3000



Selfie photo
Your personal photo. The photo must be done by yourself. The photo must show your face and your both shoulders.
ID Card
Valid Government ID card must be shown. Front and back side. Original ID card. Full photo. All parts of the ID card must show on the photo.

5.11.Backend



Rows: 154

	Gender	MaritalSt...	Depende...	Education	Self_Empl...	Applicant...
55	Male	Single	2000	Graduate	1	20000
56	Male	Single	2000	Graduate	1	20000
57	Male	Single	2000	Student	1	20000
58	Male	Single	2000	Student	1	20000
59	Female	Single	2000	Student	1	20000
60	Female	Single	2000	Student	1	20000
61	Male	Single	2000	Student	1	20000
62	Male	Single	20000	Student	1	20000
63	Male	Single	5000	Student	1	5000
64	Male	Single	5000	Student	1	5000
65	Male	Single	5000	Student	1	5000
66	Male	Single	5000	Student	1	5000

Page 1 / 2

Implementation

6.Implementation

6.1. Main page

page.jsx

```
import Image from 'next/image'

import styles from './page.module.css'

import Nav from './nav/Nav'

import Homee from './Slider/Home'

import About from './About/About'

import Apply from "./Apply/Apply"

import Footer from './Footer/Footer'

import BusinessBank from './BusinessBank/BusinessBank'

import Calculator from './Calculator/Calculator'

import Slider from './Calculator/Slider'

import Contact from './Contact/Contact'

export default function Home() {

  return (

    <main  >

  <Nav />
```

```
      <Apply />

      <Homee />

      <About />

      <BusinessBank />

      <Calculator />

      <Slider />

      <Contact />

      <Footer />

    </main>

  )

}
```

6.1.1.Nav.jsx

The "QuickLoan" logo appears at the top left of the page.

The site name "QuickLoan" appears next to the logo.

The site contains a navigation bar at the top of the page that includes the following sections:

Home: This page leads to the main page of the site.

About Us: This page provides information about QuickLoan and its services.

Business Loans : This page provides information about business loans offered by QuickLoan.

Calculator :This tool allows you to calculate the cost of your loan.

Contact Us: This page provides information on how to contact QuickLoan.

Apply Now: This page leads to a loan application form.

6.1.2.Nav.jsx

```
"use client"

// import React, { useState } from 'react';

import { Link } from 'react-router-dom';
import styles from './Nav.module.css'


function Nav() {


  return (

    <>

      <nav className={styles.nav1}>



        <ul className={styles.ul_nav} id="ul_nav">

          <img className={styles.logo} src='../../nav/icon4.png'
alt="My " />

          <h1 className={styles.h1_nav}> Quick LOAN</h1>

          <li className={styles.li_nav}> <a
className={styles.a_nav}

            href="#home">HOME</a></li>

          <li className={styles.li_nav}> <a
className={styles.a_nav}

            href="#About">ABOUT</a> </li>

          <li className={styles.li_nav}> <a
className={styles.a_nav}
```

```
          href="#Business_bank">Business bank</a></li>

         <li className={styles.li_nav}><a
className={styles.a_nav}

          href="#Calculator">Calculator</a> </li>

         <li className={styles.li_nav}> <a
className={styles.a_nav}

          href="#Contact">Contact</a></li>

         <button type="button" className={styles.but_Apply}>

          <a className={styles.a_Apply} href="" >

           Apply </a>

         </button>


        </ul>

        <div className={styles.icons}>


        </div>


        <div className={styles.icons}>
```

```
          <img id='open' src='../../nav/icon1.png' alt="My " />
</div>
        {/* <img id='colos' src='../../nav/icon2.png' alt="My "
/>  */}


      </nav>


    </>
  )



}
export default Nav
```

6.1.3.Nav.module.css

```
.nav1 {
    background-color: #333;
    display: flex;
    padding-left: 5vw;
```

```
}

.ul_nav {
   font-family: 'Lato', sans-serif;
   list-style-type: none;
   display: flex;
   align-items: center;
   letter-spacing: .2VW;
   margin: 0px;

}

.h1_nav {
   float: left;
   padding-right: 10vw;
   color: #fff7f7;
   width:70px;
   font-size: 1.5vw;
```

```
}
.logo{
   margin-right: 7px;
   width: 50px;
   height: 50px;
   }

.li_nav {
   margin-left: 3vw;
}

.a_nav {
   text-decoration: none;
   color: rgb(253, 249, 249);
   font-size: 1.25vw;
   position: relative;
}

.a_nav::after {
   content: '';
   display: block;
   width: 0;
   height: .4vh;
   background-color: rgb(20, 106, 106);
   position: absolute;
   bottom: -.6vh;
   left: 50%;
   transform: translateX(-50%);
   opacity: 0;
   transition: opacity 0.2s ease-in-out, width 0.5s ease-in-out;
}

.a_nav:hover::after {
   width: 100%;
   opacity: 1;
}

.but_Apply {
   background-color: transparent;

   border: .3vw solid #379cf4;

   margin-left: 3vw;

   padding: .7Vh 1.2vw;

   box-shadow: .4vh .3vw #585a5b;

   border-radius: 10vw;


   transition:

       background-color 0.3s ease-in-out,

       border-radius 0.2s ease-in-out,

       box-shadow 0.3s ease-in-out,

       border-color 0.5s ease-in-out;


}

.but_Apply:hover {

   background-color: rgb(0, 191, 255);

   border-radius: 10vw;

   box-shadow: .4vh .3vw #585a5b;

   border-color: #333;
```

```
}


.a_Apply {

   text-decoration: none;

   color: white;

   font-size: 1.5vw;

   padding: 1.25vw;


}


.icons {




   display:none;


}


@media screen and (max-width: 664px) {
```

```
.nav1 {
    height: 10vh;
    align-items: center;


}


.h1_nav {


    color: #fff7f7;
    font-size: 4vw;


}


.but_Apply {
    margin-left: 10vw;
    padding: 1Vh 3vw;


}
```

```
    .a_Apply {
        font-size: 4vw;


    }


    .li_nav {
        display:none;
    }


    .icons {
        display: none   ;


    }
}
```

2.1.Home

1. Logo:

   The "QuickLoan" logo appears at the top left of the page.

2. Web site name:

The site name "QuickLoan" appears next to the logo.

3. Navigation bar:

The site contains a navigation bar at the top of the page that includes the following sections:

Home:This page leads to the main page of the site.

About Us:This page provides information about QuickLoan and its services.

Business Loans: This page provides information about business loans offered by QuickLoan.

Calculator: This tool allows you to calculate the cost of your loan.

Contact Us:This page provides information on how to contact QuickLoan.

Apply Now: This page leads to a loan application form.

4. Page title:

The page title “Start or Grow Your Own Business” appears below the navigation bar.

6. Promotional photo:

A large promotional image appears in the middle of the page showing a man and woman working in their office.

6. Promo text:

A promotional text appears below the promotional image that says:

> Get what matters

> Start or grow your own business

> Get your dream in simple steps and without any guarantees!

> Get a loan from your income

> Apply now

2.2.Home.jsx

```
import styles from './Slider.module.css'


function Home() {


  return (

    <div className={styles.container}id='Home'>
```

```
          <div className={styles.cont}>
          <p className={styles.text_home}   >Get what matters
</p>
          <h1 className={styles.text2_home}>
            Start or grow your own business</h1>
          <h4 className={styles.text3_home}>Get your dream in
a few
            simple steps, and without any guarantees!
            Get a loan from your income
          </h4>
          <button type="button"
className={styles.but_Apply_home}>
            <a className={styles.a_Apply_home}
href="forms.html">
              Apply Now</a>


          </button>
          </div>
        </div>
    )
```

```
}


export default Home
```

2.2.3.Slider.module.css

```
.container {
   background-image: url("../../../public/img10.jpg");
   color: white;
   background-repeat: no-repeat;
   background-size: cover;
   padding-left: 8vw;
   padding-right: 8vw;
   /* height: 80vh; */
   opacity: 0.7;
   ;
   margin-top: 0;
   position: relative;
```

```
}

.text_home {
   color: rgb(255, 252, 252);
   border-radius: 20px;
   margin-top: 7vh;
   padding: .2vw 2vw;
   display: inline-block;
   border: 2px solid rgb(0, 191, 255);
   font-size: 2vw;
   opacity: 0; /* جعل النص غير مرئي في البداية */
  animation: slideInFromRight 1s ease-in-out 1s forwards;
  transform: translateX(-50%);

}
@keyframes slideInFromRight {
   to {
     opacity: 1;
     transform: translateX(0%);
```

```
    }
  }

.text2_home {
    color: rgb(253, 253, 253);
    font-size: 5vw;
    width: 50vw;
    margin-top: 1vh;
    transform: translateX(50%);
    opacity: 0;
    animation: slideInFromRight  2s ease-in-out 2s forwards;
}
@keyframes slideInFromRight  {
    to {
      opacity: 1;
      transform: translateX(0%);
    }
  }
```

```
.text3_home {
   color: rgb(247, 239, 239);
   width: 45vw;
   opacity: 0;
   animation: slideInFromTop  3s ease-in-out 2s forwards;
   transform: translateY(-100%);
}
@keyframes slideInFromTop  {
   to {
      transform: translateY(0%);
     opacity: 1;
   }
 }

.but_Apply_home {
   margin-top: 2vh;
   margin-bottom: 7vh;
   padding: .5vh 1.5vw;
   background-color: rgba(16, 152, 243, 0.927);
```

```
    font-size: 2.5vw;

    border-radius: 20vw;

    border: none;

    opacity: 0;

    animation: slideInFromTop  3s ease-in-out 1s forwards;


    transform: translateY(100%);

}

@keyframes slideInFromTop  {

    to {

        transform: translateY(0%);

      opacity: 1;

    }

  }


.a_Apply_home {

    text-decoration: none;

    color: rgb(255, 239, 239);

    font-size: 2vw;
    position: relative;
}
@media screen and (max-width: 664px) {
    .container {
        height: 100%;
    }
    .text_home {
        margin-top: 4vh;
        display: inline-block;
        border: 3px solid rgb(0, 191, 255);
        font-size: 4vw;
    }
    .text2_home {
        font-size: 7vw;
        width: 70vw;
        margin-top: 1vh;
    }

    .text3_home {
        color: rgb(247, 239, 239);

        width: 50vw;

        font-size: 3vw;

    }

    .but_Apply_home {

       margin-bottom: 7%;

        padding: .5vh 2vw;

        font-size: 4vw;

        border-radius: 20vw;

    }


    .a_Apply_home {

        font-size: 4vw;

    }


}
```

3.1.About

Company logo: 1.5 million active customers and 30,000 business partners

About the company:

A leading company in the field of quick loans.

Offers quick and easy personal loans.

It has an easy-to-use website for requesting loans online.

It has a good reputation for providing fair loans at competitive interest rates.

How Quick LOAN works:

1. Fill out the loan application online.
2. You will receive an instant decision.
3. If your application is approved, you will receive the funds in your bank account within hours.

Quick LOAN Advantages:

SPEED: Quick LOAN offers fast and easy loans that can be obtained within hours.

Ease of use: Loans can be easily requested online through the Quick LOAN website

Competitiveness: Quick LOAN offers competitive interest rates and flexible terms.

Reliability: Quick LOAN has a reputation for providing fair loans at competitive interest rates.

3.2.About.tsx

```
import styles from './About.module.css'

function About() {

  return (

    <section className={styles.AboutUs} id='About'>

      <div className={styles.titl}>

        <h5>About Us</h5>

        <h1>Learn about how Banca works</h1>

      </div>

      <div className={styles.about_item}>

        <div className={styles.about_text}>

          <div className={styles.text1}>
```

```
                    <div>
                        <h1>1.5M</h1>
                        <h5>Active Customer</h5>
                    </div>
                    <p>There are many variations of passages of
Lorem Ipsum available,but the majority have.</p>
                </div>
                <div className={styles.text1}>
                    <div>
                        <h1>30k</h1>
                        <h5>Business Partners</h5>
                    </div>
                    <p>There are many variations of passages of
Lorem Ipsum available,but the majority have.</p>
                </div>
            </div>
            <div >
            <img className={styles.about_imeag}
src='../../About/icon1.png' alt="My2 " />
```

```
                </div>

            </div>

        </section>

    )

}

export default About
```

3.3.About.module.css

```
.AboutUs {

    margin: 0;

    box-sizing: border-box;




}



.titl h5 {

    color: gray;

    text-align: center;
    margin-top: 10px;

    font-size: 20px;

}


.titl h1 {

    text-align: center;

    font-weight: bolder;

    font-family: Arial, Helvetica, sans-serif;

    margin-top: 10px;

}


.text1 {

    padding-left: 6vw;

    margin: 10px;

}


.text1 p {

    margin-top: 10px;

    color: rgb(105, 104, 104);
```

```
}

.text1 h1 {
   display: inline-block;
   font-weight: bolder;
   font-family: Arial, Helvetica, sans-serif
}

.text1 h5 {
   color: rgb(76, 75, 75);
   display: inline-block;
   font-size: 15px;
}

.about_item {
   margin-top: 20px;
   display: flex;
   justify-content: space-between;
   align-items: center;
```

```
    /* padding: 0 20px ; */

}

.about_imeag{

    width: 90%;

    height: auto;

}

@media screen and (max-width: 1210px) {

    .about_imeag {

        flex-direction: column;

        margin: auto;

    }

}


@media screen and (max-width: 664px) {


    .about_item {

        flex-direction: column;

        margin: auto;

    }
    .about_imeag {
        padding-left: 6vw;


    }



}
```

4.2.BusinessBank

The photo shows a person using a credit card reader to pay a loan.

Text:

Business Bank(Business Bank)

Loan installment

The required documents

the conditions

Credit

The photo shows a person using a credit card reader to pay a loan. The "Business Bank" logo can be seen on the card reader. It is noteworthy that this service is available to customers who have obtained a loan from the bank.

The required documents:

ID card

Bank statement

Salary slip

Any other documents required by the bank

the conditions:

The customer must fulfill all other bank requirements

Credit:

Business Bank offers loans at competitive interest rates

Customers can choose the loan repayment period that suits their needs

The bank offers flexible payment options, including fixed monthly installments or lump sum payment

4.2.BusinessBank.jsx

```
"use client"
import styles from './BusinessBank.module.css';
import React from "react"
export default function BusinessBank() {

  const [isVisible1, setIsVisible1] = React.useState(true);
  const [isVisible2, setIsVisible2] = React.useState(false);
  const [isVisible3, setIsVisible3] = React.useState(false);
  const [isVisible4, setIsVisible4] = React.useState(false);

  const handleButtonClick1 = () => {
    setIsVisible1(!isVisible1);
    setIsVisible1(true);
    setIsVisible2(false);
    setIsVisible3(false);
    setIsVisible4(false);
};
const handleButtonClick2 = () => {
    setIsVisible2(!isVisible2);
    setIsVisible2(true);
    setIsVisible1(false);
    setIsVisible3(false);
    setIsVisible4(false);
};
const handleButtonClick3 = () => {
    setIsVisible3(!isVisible3);
    setIsVisible3(true);
    setIsVisible2(false);
    setIsVisible1(false);
    setIsVisible4(false);
};
const handleButtonClick4 = () => {
    setIsVisible4(!isVisible4);
```

```
        setIsVisible4(true);

        setIsVisible2(false);

        setIsVisible3(false);

        setIsVisible1(false);

    };

    return (

        <div className={styles.all} id='Business_bank'>

            <h1 className={styles.business}> Business Bank</h1>

            <div className={styles.all_cont}>

                <div className={styles.tabs}>

                    <div className={styles.tab}
onClick={handleButtonClick1} > Loan Payment</div>

                    <div className={styles.tab}
onClick={handleButtonClick2}> Required Documents</div>

                    <div className={styles.tab}
onClick={handleButtonClick3}> The Conditions</div>

                    <div className={styles.tab}
onClick={handleButtonClick4}> Credit</div>



                </div>
            {/* isVisible1  */}
            <div id="div1" style={{ display: isVisible1 ? 'block' :
'none' } }>

              <div id="tab1" className={styles.tab_content}>
                <div className={styles.tap1}>
                  <div className={styles.tab1_item}>

                    <img className={styles.img}
src="./img18.jpg" alt='11' />

                    <div className={styles.content}>

                      <h1> We appreciate your trust in us and
are delighted to offer you this loan.</h1>
                      <p>You can repay the loan over a
                        flexible period
                        that suits your needs and financial
capabilities."
```

```
                    "We will provide you with convenient
and suitable repayment options, allowing you to choose
between
                    fixed monthly
                    installments or a lump sum payment at a
specific time."
                    "Our goal is to make the loan repayment
process comfortable for you. Therefore, you can determine the
                    repayment period
                    that you prefer, whether it's short-term
or long-term.</p>


                </div>
              </div>


            </div>


          </div>
        </div>
        {/* isVisible2  */}
```

```
            <div id="test" style={{ display: isVisible2 ? 'block' :
'none' }}>

             {/*   محتوب 2*/}

             <div id="tab2" className={styles.tab_content}>

                <div className={styles.tap1}>

                   <div className={styles.tab1_item}>

                      <img className={styles.img}
src="./img14.jpg" />


                      <div className={styles.content}>


                         <h1> Completed Loan Application</h1>

                         <p> You must fill out  loan application:

                            and submit it with the

                            required

                            information accurately and in detail.

                            Income Support Documents: You must
provide documents that prove a stable income, such as salary
slips or

                            tax returns.
```

```
                                        Identity Documents: Valid and
recognized identity documents must be submitted, including a
passport or
                                        national ID card.</p>


                                    </div>


                                  </div>
                              </div>


                          </div>
                      </div>
                      {/* isVisible3  */}
                      <div id="test" style={{ display: isVisible3 ? 'block' :
'none' }}>
                          <div id="tab3" className={styles.tab_content}>
                              <div className={styles.tap1}>
                                  <div className={styles.tab1_item}>
                                      <img className={styles.img}
src="./img19.jpg" />
```

```
                    <div className={styles.content}>


                      <h1> Credit History:</h1>
                      <p> Positive impact of a good credit history
and regular debt repayment on loan eligibility.


                        Income and Repayment Capacity: Need
for a stable income and sufficient financial capacity to repay
the
                        loan.
                        Personal Documentation: Submission of
valid identification and current residential address documents.
                      </p>
                    </div>
                  </div>


                </div>
              </div>
            </div>
            {/* isVisible4  */}
            <div id="test" style={{ display: isVisible4 ? 'block' :
'none' }}>
              <div id="tab4" className={styles.tab_content}>
                <div className={styles.tap1}>
                  <div className={styles.tab1_item}>
                    <img className={styles.img}
src="./img21.jpg" />

                    <div className={styles.content}>

                      <h1> Completed Loan Application
                        You must fill out  loan application:</h1>
                      <p>  and submit it with the
                        required
                        information accurately and in detail.
                        Income Support Documents: You must
provide documents that prove a stable income, such as salary
slips or
                        tax returns.
                    Identity Documents: Valid and
recognized identity documents must be submitted, including a
passport or
                    national ID card.</p>

                </div>
              </div>

            </div>
          </div>
        </div>

      </div>
    </div>
  )
};
```

4.3.BusinessBank.module.css

```
.all {

  background-color: rgb(186, 197, 197);



}



.business {

  padding-left: 40vw;

  padding-top: 10vh;

  padding-bottom: 1vw;



}



.tabs {

  margin-left: 6vw;

  margin-right: 6vw;

  text-align: center;
  display: flex;

  font-size: 1.5vw;

  position: relative;

  width: 90%;

  padding-bottom: 1vw;

  border-bottom: 2px solid rgb(20, 106, 106);

}


.tab {

  letter-spacing: 1px;

  margin-left: 4%;

  text-align: center;

  opacity: 1;

  padding-left: 6vw;

  color: #000;

  cursor: pointer;

}


.tab::after {
  content: '';

  display: flex;

  width: 0;

  height: .4vh;

  background-color: rgb(20, 106, 106);

  bottom: -.6vh;

  transform: translateX(-50%);

  opacity: 0;

  transition: opacity 0.2s ease-in-out, width 0.5s ease-in-out;

}


.tab:hover::after {

  color: red;

  width: 100%;

  margin-left: 50%;

  opacity: 1;

}
```

```
.tap1 {

  margin-left: 6vw;

  display: flex;



}

.tab1_item {
  display: contents;
  align-items: center;
  margin: auto;
  height: 70px;
}

.img {
  margin-top: 4%;
  margin-bottom: 4%;
  width: 40%;
  height: 100%;
  transform: translateX(-50%);
  opacity: 0;
  animation: slideInFromRight .5s ease-in-out .5s forwards;

}

@keyframes slideInFromRight {
  to {
    opacity: 1;
    transform: translateX(0%);
  }
```

```
}

.content {
  margin: 40px 4px 4px 25px;
   transform: translateX(50%);

  opacity: 0;
  animation: slideInFromRight .5s ease-in-out .75s forwards;

}

@keyframes slideInFromRight {
  to {
    opacity: 1;
    transform: translateX(0%);
  }
}

/* Hide the tab content by default */
```

```
@media screen and (max-width: 664px) {
 .all{
  margin-bottom: -6vw;
 }



  .tabs{
font-size: 2.5vw;
font-weight: 700;


 }
 .business{
```

```
 padding-left: 23vw;

 padding-top: 2vh;

 padding-bottom: 0vw;

}

 .img{

   display: flex;

   width: 80%;

   margin: 6vw;

}

.tap1{

 display: flow;

 font-size: x-small;

 padding-bottom: 4vw

}




}
```

5.1.LoanCalculator

The image shows a calculator that can be used to calculate and confirm loans. The calculator consists of several fields for entering information, including:

Loan Amount: The amount you borrow from the lender.

Interest Rate: The percentage of the loan amount that is charged as interest over the life of the loan.

Loan Term:The length of time you will repay the loan, usually in years or months.

The calculator also shows the following results:

Monthly payment: The amount you will pay each month to repay the loan, including interest and principal.

Total Paid: The total amount you will pay to repay the loan over the life of the loan.

Total Interest: The total amount you will pay in interest over the life of the loan.

Explanation of fields:

Loan amount:

The loan amount is the amount you borrow from the lender. This amount is used to calculate the interest rate, monthly payments, total paid, and total interest.

interest rate

The interest rate is the percentage of the loan amount that is charged as interest over the life of the loan. The interest rate is used to calculate the monthly payments, total paid, and total interest.

Loan term:

The loan term is the length of time you will repay the loan, usually in years or months. The loan term is used to calculate the monthly payments, total paid, and total interest.

Monthly payment:

The monthly payment is the amount you will pay each month to repay the loan, including interest and principal. The monthly payment is calculated using the following formula:

PMT = (PV * r) / (1 - (1 + r)^(-n))

where:

PMT = Monthly Payment

PV = loan amount

r = monthly interest rate (annual interest rate / 12)

n = number of payments (loan term * 12)

Total Paid:

The total payment is the total amount you will pay to repay the loan over the life of the loan. The total paid is calculated using the following formula:

Total Payment = PMT * n

Total interest:

Total interest is the total amount you will pay in interest over the life of the loan. The total interest is calculated using the following formula:

Total Interest = Total Payment - PV

A loan calculator can be used to calculate the total costs of a loan before borrowing. They can also be used to compare loan offers from different lenders

It can help you make informed decisions about borrowing.

Can help you save money on interest.

It can help you avoid borrowing more than you can afford.

Tips for using the loan calculator:

Read all loan terms and conditions before borrowing.

Compare loan offers from different lenders.

Make sure you understand all costs associated with the loan, including interest, insurance fees, and processing fees.

Don't borrow more than you can afford to repay.

5.2.LoanCalculator.jsx

```
"use client";
import styles from './Calculator.module.css';
import React, { useEffect, useState } from "react";
function LoanCalculator() {
  const [loanAmount, setLoanAmount] = useState(0);
  const [loanTerm, setLoanTerm] = useState(0);
  const [percentage, setPercentage] = useState(0);

  const [monthlyPayment, setMonthlyPayment] = useState(0);
  const [totalPayment, setTotalPayment] = useState(0);
  const [totalInterest, setTotalInterest] = useState(0);
```

```
useEffect(() => {
  calculateLoan();
  updateTooltip("sliderLoanAmount", "tooltipLoanAmount");
  updateTooltip("sliderLoanTerm", "tooltipLoanTerm");
  updateTooltip("sliderPercentage", "tooltipPercentage");
}, [loanAmount, loanTerm, percentage]);

const updateTooltip = (sliderId, tooltipId) => {
  const slider = document.getElementById(sliderId);
  const tooltip = document.getElementById(tooltipId);

  if (slider && tooltip) {
    const maxValue = slider.getAttribute("max");
    const val = slider.value;
    const center = (val / maxValue) * 100 + "%";
    tooltip.style.left = center;
  }
};
```

```
const calculateLoan = () => {
  // Calculation logic for loan
  const principal = parseFloat(loanAmount);
  const annualInterest = parseFloat(percentage) / 100;
  const numberOfPayments = parseFloat(loanTerm) * 12;

  const x = Math.pow(1 + annualInterest, numberOfPayments);
  const monthly = (principal * x * annualInterest) / (x - 1);

  const total = monthly * numberOfPayments;
  const totalInterest = total - principal;

  setMonthlyPayment(monthly.toFixed(2));
  setTotalPayment(total.toFixed(2));
  setTotalInterest(totalInterest.toFixed(2));
};

return (
<>


 <section className={styles.Calculator} id='Calculator'>

  <div className={styles.titl}>

   <h5> loan Calculator</h5>

   <h1> Calculate and confirm your loans</h1>

  </div>

  <div className={styles.Calculator_item}>

   <div className={styles.Calculator_text}>

    <div className={styles.text1}>

     {/* <h1>loan Calculator</h1> */}

     <h5 className={styles.term}>Loan term</h5>

    </div>


   </div>

  </div>


 </section >
```

```
      <div className={styles.tatal}>


       <section className={styles.Calculator} id='Calculator'>

        <h4 className={styles.h6}>Loan Amount</h4>


        <div className={styles.totalcont}>

         <div className={styles.test1}>

          <input

           className={styles.test}

           aria-valuemin={0}

           type="range"

           min="5000"

           max="1000000"

           value={loanAmount}

           id="sliderLoanAmount"

           onChange={({ target: { value } }) =>
setLoanAmount(value)}

          />

         </div>
```

```
  <div id="progress"></div>
  <div
    id="tooltipLoanAmount"
    className={styles.cont}
  >
    {loanAmount}
    LE
  </div>
</div>
<h4 className={styles.h6}>Interest Rate</h4>
<div className={styles.totalcont}>
  <div className={styles.test1}>
    <input
      className={styles.test}
      aria-valuemin={1}
      type="range"
      min="1"
      max="12"
```

```
              value={loanTerm}
              id="sliderLoanTerm"
              onChange={({ target: { value } }) =>
setLoanTerm(value)}
            />
          </div>

          <div id="progress"></div>

          <div
            id="tooltipLoanTerm"
            className={styles.cont}
          >
            {loanTerm}
            Y
            {/* {loanTerm !== 1 && "y"} */}
          </div>
        </div>
        <h4 className={styles.h6}>Loan Duration</h4>
```

```
          <div className={styles.totalcont}>
            <div className={styles.test1}>
             <input
               className={styles.test}
               aria-valuemin={0}
               type="range"
               min="0"
               max="100"
               value={percentage}
               id="sliderPercentage"
               onChange={({ target: { value } }) =>
setPercentage(value)}
              />
            </div>
            <div id="progress"></div>
            <div
              id="tooltipPercentage"
              className={styles.cont}
            >
          {percentage}%
         </div>
        </div>
       </section>

       <div className={styles.result}>
        <h3 className={styles.Details} >Loan Details</h3>
        <div>
         <span className={styles.Payment}>Monthly
Payment:</span> <span
className={styles.totalPayment}>{monthlyPayment}</span>

        </div>
        <br />
        <div>

         <span className={styles.Payment}>Total
Payment:</span> <span
className={styles.totalPayment}>{totalPayment}</span>
        </div>
```

```
          <br />


          <div>

            <span className={styles.Payment}>Total
Interest:</span> <span
className={styles.totalPayment}>{totalInterest}</span>

          </div>


        </div>

      </div>

    </>

  );

}


export default LoanCalculator;
```

5.3.LoanCalculator.module.css

```
.Calculator {
    margin: 0;
    box-sizing: border-box;


}

.titl h5 {
    color: gray;
    text-align: center;
    margin-top: 10px;
    font-size: 20px;
}

.titl h1 {
    text-align: center;
    font-weight: bolder;
    font-family: Arial, Helvetica, sans-serif;
}

.text1 {
    padding-left: 6vw;

}

.Calculator_item {
    margin: 0px;

    padding: 0vw;
}

.term {
    font-size: 2.5vw;
    padding-top: 0vw;
}
```

```
.Calculatorr {
    position: relative;
    display: block;
    font-size: 1vw;



}

.totalcont {
    display: flex;
}



.test1 {
    margin-left: 7vw;
    outline: none;
    width: 33vw;
    height: 3vw;
    background-color: aqua;
    border-radius: 20px;

    box-shadow: inset 0 0 5px rgb(115, 19, 19);

    padding: .5vw 0vw 0vw 0vw;

    margin-top: 12px;

    text-align: center;

    line-height: 3vw;



}


.cont {

    margin-left: 2vw;

    width: 14vw;

    font-size: 2.5vw;

    border-radius: 20px;

    border: 2px solid black;

    text-align: center;

    line-height: 4vw;
```

```
}


.h6 {

    padding-left: 7vw;

    margin: 10px;

    padding-bottom: 0vw;


}


.test {

    width: 30vw;

    display: inline-block;

}


.tatal {

    display: flex;

}


@media only screen and (max-width: 393px) {
    .tatal {
        display: block;
        padding: 4vw;
    }
}

.result {
    border: 2px solid black;


    width: 41vw;
    font-size: 2vw;
    margin-left: 3vw;
}

.Details {
    color: #1765ed;
    padding: 0vw 0vw 0vw 1vw;
}
```

```
.Payment {
    padding-left: 4vw;


}


.totalPayment {
    border: 2px solid #1765ed;
    color: #1765ed;
    padding: 10px;
}
```

6.1.Contact

The page contains a contact form that requires the user to enter the following information:

Name: Full user name.

Email: The user's email address.

Phone number: The user's phone number.

Message: A message from the user stating his inquiry or request.

The image also contains the following information on the left side

Phone number 01288456585

Email sandora.wadee@yahoo.com

Working hours: Monday to Saturday from 9Am-6Pm, with Sunday off

User's location: The user's current location (Assiut, Assiut Governorate, Egypt).

Explanation of the form's functions:

Name: This field helps identify the user.

Email: This field helps to communicate with the user directly.

Phone number: This field helps to communicate with the user directly if email is not available.

Message: This field allows the user to write his inquiry or request in detail.

This page aims to facilitate and speed up communication between the bank and its customers or those interested in it

6.2.Contact.jsx

```
import styles from './Contact.module.css';
```

```
function Contact() {
    return (
        <section className={styles.Contactt} id='Contact'>


            <div className={styles.contact}>


                <h1 className={styles.h1}>Contact</h1>
                <div className={styles.contact_child}>
                    <div>
                        <span className={styles.span}>GET IN
TOUCH</span>
                        <h2>Have Queries Befor The Appointment?</h2>
                        <ul className={styles.ul}>
                            <li className={styles.li}>
                                <h4 className={styles.part1}>Lets Talk</h4>
                                {/* <br /> */}
                                <span
className={styles.span}>phone:01204244567</span>
                                <br />
                    <br />
                    <span
className={styles.span}>email:sandra@gmail.com</span>
                  </li>
                  <li>
                    <h4>Timing</h4>
                    <span className={styles.span}>Mon - Sat :
09:00AM - 06:00PM</span>
                    <br />
                    <br />
                    <span className={styles.span}>Sunday :
Closed</span>
                  </li>
                  <li>
                    <h4>Location</h4>
                    <span className={styles.span}>EL-gomhoria
st</span>
                  </li>
                </ul>
              </div>
```

```
          <div className={styles.contact_child1}>
            <h3>Dont Hesited TO Contact Us</h3>
            <div>
              <input className={styles.input} type="text"
placeholder="Name" />
              <input className={styles.input} type="text"
placeholder="Email" />
            </div>
            <div>
              <input className={styles.phone}
type="number" placeholder="Phone" />
            </div>
            <textarea className={styles.textarea} cols="55"
rows="7" placeholder="Message"></textarea>
            <div className={styles.buttonn}>
              <button
className={styles.button}>Send</button>
            </div>
          </div>
        </div>
```

```
           </div >
           </section>
              )
}
export default Contact
```

6.3.Contact.module.css

```
.Contactt {
   background-color: rgb(186, 197, 197);
   width: 100%;


}


.h1 {
   padding-top: 1vw;
   color: rgb(1, 1, 1);
   text-align: center;
   margin-top: 10px;
```

```
    font-size: 2.5vw;
}

@media only screen and (max-width: 393px) {
    .h1 {
        font-size: 5vw;


    }
}

.part1 {
    padding-left: 0px;
    /* background-color: aqua; */


}

.contact_child {
    display: flex;
    justify-content: space-around;
    margin-top: 2rem;

    flex-wrap: wrap;

    padding-left: 4vw;


}


.contact_child1 {

    box-shadow: 0px 0px 2px 0.5px rgb(255, 255, 255);

    padding: 2rem;

    overflow: hidden;

    margin-left: 6vw;

    height: 60%;


}


.h3 {

    width: 55%;

    /* margin: auto; */

    margin-bottom: 0.5rem;
```

```
}

.input {
   width: 150px;
   background-color: rgb(247, 246, 246);
   border: none;
   padding-left: 1.5vw;
   margin: 1rem;
}

.phone {
   width: 75%;
   background-color: rgb(247, 246, 246);
   border: none;
   padding-left: 1.5vw;
   margin: 0rem 1rem 1rem;
}

.textarea {
   background-color: rgb(247, 246, 246);

   border: none;

   padding: 1px;

   width: 24rem;

   margin-left: 10px;

}


.buttonn {



   width: 15%;

   margin: auto;

}


.button {

   cursor: pointer;

   width: 100%;

   padding: 0.6rem;

   border: none;

   background-color: rgb(26, 26, 228);
   color: white;

}


.ul {


   list-style: none;

}



.span {

   padding: 0.5rem;

   margin: 0.5rem;

   color: rgb(253, 253, 253);

}


@media only screen and (max-width: 393px) {

   .buttonn {

       width: 30%;

   }
```

```
}
```

7.1.Footer

....

7.2.Footer.jsx

```
import styles from './Footer.module.css';

function Footer() {

  return (


    <footer className={styles.footerr}>

      <div className={styles.footer}>

        <div className={styles.footer_item}>

          <ul className={styles.ul1}>

            <li>

              <h4>Banca is a leading bank in the worldzone
and  prominent international banking institution</h4>

            </li>
  <li>COTATION</li>
  <li>2023-01-05 14:00(INTERNATIONAL TIME)</li>
</ul>
<ul>
  <li>
    <h4>Banca At A Glance</h4>
  </li>
  <li>Our 'company purpose'</li>
  <li>Jobs & Careers</li>
  <li>Our Responsibility</li>
  <li>Our Core Businesses</li>
</ul>
<ul>
  <li>
    <h4>Publication</h4>
  </li>
  <li>Compliance Publication</li>
  <li>Annual Reports</li>
  <li>CSR Reports</li>
```

```
           <li>Financial documentation</li>
          </ul>
          <ul>
           <li>
            <h4>Banca At A Glance</h4>
           </li>
           <li>Our news</li>
           <li>Our press releases</li>
           <li>Our job offers</li>
           <li>Our websites</li>
          </ul>
         </div>
        </div>
        <div className={styles.social_footer}>
         <div className={styles.social_footeritem}>
          <div className={styles.logoo}>
           <h5>
            <img className={styles.logo} src='../../nav/icon4.png'
alt="My " />
```

```
          Quick LOAN
        </h5>
      </div>
      <h6>@2023 All Right Reserved by Spider-Themes</h6>
      <div  >
        <a href="https://www.facebook.com">
          <img className={styles.icons}
src='../../Footer/icon1.png' alt="My " />
        </a>

        <a href="https://www.twitter.com/">
          <img className={styles.icons}
src='../../Footer/icon2.png' alt="My " />
        </a>

        <a href="https://www.facebook.com">
          <img className={styles.icons}
src='../../Footer/icon3.png' alt="My2 " />
        </a>
```

```
          </div>








         </div>

        </div>



      </footer>

      // <script src="./all.min.js"></script>



    )

  }

  export default Footer
```

Footer.module.css

```
  .footerr {

      background-color: rgb(25, 25, 25);
```

```
    color: white;

    position: relative;


}


.footer {

    border: 1px solid rgb(25, 25, 25);;

    width: 90%;

    margin: auto;

    padding-bottom: 3rem;


}


.footer_item {

    display: flex;

    justify-content: space-between;

    padding: 1rem;

    margin-top: 2rem;

    margin-bottom: 1rem;
```

```
}

.footer_item ul {
   list-style: none;
}

.footer_item ul li {
   font-size: 12px;
   margin-top: 1rem;
   color: rgb(154, 149, 149);
}

.footer_item ul li h3 {
   color: white;
}

.footer_item ul li h4 {
   color: white;
   font-size: 12px;
```

```
}

.social_footer {
    background-color: rgb(43, 40, 40);
    position: absolute;
    width: 100%;
    bottom: 0%;
}

.social_footeritem {
    width: 90%;
    margin: auto;
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.icon_social svg {
    font-size: 15px;
    margin: 0 10px;

}


.ul1 {

    width: 30%;



}

.logoo h5{

    padding-left: 2vw;

    display: flex;

    width: 50%;



}

.logo{



    width: 30px;

    padding-right: 5px ;



}
```

```
@media screen and (max-width: 664px) {
  .footer_item {
    flex-direction: column;


  }


  .footer_item ul {
    margin: auto;
    width: 250px;
    font-size: 15px;
    margin-bottom: 20px;
  }


  .social_footeritem {
    flex-direction: column;
    gap: 5px;
  }


  .about_imeag {
    width: 300px;
    margin: 0%;
  }


  .text1 {
    width: 300px;
  }


.logoo h5{
  margin: 0px;
  bottom: 0px;
  display: flex;
  /* width: 50%; */


}
.logo{
  width: 30px;

  padding-bottom: 5px ;


}


}


.icons {

  width: 20px;

  height: 20px;

  /* /-right: 2px; */

}
```

Apply

7.1.Apply

7.2. Apply.jsx

```
"use client"

import styles from './Apply.module.css';

import React from "react"

// import Nav from './nav/Nav'

import Form from './Form'


// import React, { useEffect } from "react";


export default function Apply() {

    const [isVisible1, setIsVisible1] = React.useState(true);

    const [isVisible2, setIsVisible2] = React.useState(false);

    const [isVisible3, setIsVisible3] = React.useState(false);


    const handleButtonClick1 = () => {

        setIsVisible1(!isVisible1);

        setIsVisible1(true);
    setIsVisible2(false);

    setIsVisible3(false);

};

const handleButtonClick2 = () => {

    setIsVisible2(!isVisible2);

    setIsVisible2(true);

    setIsVisible1(false);

    setIsVisible3(false);

};

const handleButtonClick3 = () => {

    setIsVisible3(!isVisible3);

    setIsVisible3(true);

    setIsVisible2(false);

    setIsVisible1(false);

};

return (

    <div id='Applyll'>


        {/* home */}
```

```
      <div className={styles.apply}>
        <div className={styles.prg}>
          <h1 className={styles.h_apply}> Loan Apply</h1>
          <h5 className={styles.h_apply2}>Home  Loan
Apply</h5>
          <img className={styles.img_apply} src="./img26.png"
alt="mm" />
        </div>
      </div>

      {/* <!-- ==========box-left====== --> */}
      <div className={styles.boxLeft}>
        <ul className={styles.boxLeftUl}>
          <li className={styles.boxLeftLi} data-cont=".one">
            <span className={styles.icon}></span>
            <span className={styles.boxLeftLiTitle}
onClick={handleButtonClick1}>Documents Upload</span>
          </li>
          {/* <li className={styles.boxLeftLi} data-cont=".two">
```

```
                    <span className={styles.icon}></span>

                    <span className={styles.boxLeftLiTitle}
onClick={handleButtonClick2}>Loan Details</span>

                </li> */}

                <li className={styles.boxLeftLi} data-cont=".three">

                    <span className={styles.icon}></span>

                    <span className={styles.boxLeftLiTitle}
onClick={handleButtonClick3}>Personal Details</span>

                </li>

            </ul>


            {/* <!-- =========box- right======== --> */}

            <div className={styles.boxRight}>




                <div className={styles.content}>

                    <div className={styles.three} style={{ display: isVisible3
? 'block' : 'none' }}>

                        {/* <FormComponent /> */}
                    <Form />
                    <div className={styles.buttonContainer}>
                      {/* <button className={styles.beforeButton}
onClick={handleButtonClick2}>previous</button> */}
</div>


                    {/* <div className={styles.buttonContainer}>
                      <button
className={styles.beforeButton}>previous</button>
                    //   {/* <button
className={styles.sendButton}>Send</button> }
                      <button className={styles.sendButton}
onClick={handleButtonClick2}>Send</button>


                      </div> */}
                  </div>
                  {/* <div className={styles.two} style={{ display:
isVisible2 ? 'block' : 'none' }}> */}
                    {/* <FormComponent2 /> */}
                    {/* <div className={styles.buttonContainer}>
                      <button className={styles.beforeButton}
onClick={handleButtonClick1}>previous</button>
                      <button className={styles.sendButton}
onClick={handleButtonClick3}>Send</button>
                    </div> */}
                  {/* </div> */}


                  {/* ========content3===== */}
                  <div className={styles.one} style={{ display: isVisible1
? 'block' : 'none' }}>
                    <p>
                      Selfie photo
                      <br />
                      Your personal photo. The photo must be done by
yourself.
                      The photo must show your face and your both
shoulders.
                      <br />
                      ID Card
                      <br />
```

```
                        Valid Government ID card must be shown.

                        Front and back side. Original ID card. Full photo.

                        All parts of the ID card must show on the photo.

                    </p>

                    <form className={styles.fileUploadForm}
id="fileUploadForm">

                        <label htmlFor="fileInput">Choose a file:</label>

                        <input type="file" id="fileInput" name="fileInput"
accept=".txt, .pdf, .docx, .jpg" />

                    </form>

                    <div className={styles.buttonContainer}>

                        {/* <button
className={styles.beforeButton}>previous</button> */}


                        <button className={styles.sendButton}
onClick={handleButtonClick3}>Send</button>

                    </div>

                  </div>

                </div>

              </div>
```

```
            </div>

          </div >

    )

  };
```

Apply.module.css

```
.apply {
  background-image: url("../Apply/img/img23.jpg");
  background-repeat: repeat-x;
  background-size: cover;

}



.prg {
  /* background-image: url("../Apply/img/img23.jpg"); */
  width: 100%;
  height: 100%;

}

/*
```

```
@media screen and (max-width: 1210px) {
  .apply {
    background-size: 100% 20%;
  }


}

@media screen and (max-width: 664px) {
  .apply {
    background-size: 100% 20%;
  }



  .img_apply {
    display: none;


  }
} */
```

```
.h_apply {
  color: rgb(249, 255, 255);
  position: absolute;
  top: 25%;
  left: 50%;
  transform: translate(-50%, -50%);
  font-size: 7vw;
  letter-spacing: 0.1em;

}

.h_apply2 {
  position: absolute;
  top: 40%;
  left: 50%;
  transform: translate(-50%, -50%);
  text-align: center;
  word-spacing: 0.3em;
  font-size: 2vw;

  color: rgb(134, 126, 126);


}


.img_apply {


  width: 40%;

  height: auto;

  position: absolute;

  bottom: 1%;

  right: 1vw;

  margin: 0;

  animation: slide-in 2s ease 2s forwards;

  opacity: 0;


}


@media screen and (min-width: 1210px) {
  .img_apply {
    display: none;
  }
}

@media screen and (max-width: 664px) {
  .img_apply {
    display: none;
  }
}

@keyframes slide-in {
  0% {
    opacity: 0;
    right: 0;
  }

  1% {
    opacity: 1;
  }

  100% {
    right: 3%;
    opacity: 1;
  }
}

/* body */
/* =======box-left============== */

.boxLeft {
  padding-top: 100VH;
  background-image: url("../Apply/img/img23.jpg");
  background-repeat: no-repeat;
  background-size: 100% 49%;
  background-color: #d9dbdc;
  width: 100%;
  height: 100%;
  display: flex;

}

/* background-color: #d9dbdc; */


.boxLeftUl {
  width: 40%;
  height: 80%;
  margin: 0%;
  padding-top: 10%;
  list-style-type: none;
  counter-reset: my-counter;
  /* إعادة تعيين العداد */
}

.boxLeftLi {
  font-size: 2vw;

  position: relative;

  /* padding-left:1vw; */

  margin-bottom: 70px;

}


/* .boxLeftLi a {

 color: rgb(237, 13, 13);

} */


.boxLeftLi:before {

  content: counter(my-counter);

  counter-increment: my-counter;

  font-size: 2.5vw;

  color: #585a5b;

  padding: 3px 7px;

  border-radius: 50px;

  border-style: double;

}
```

```
}

@media screen and (min-width: 1210px) {
  .boxLeftLi:before {
    display: none;
  }
}

@media screen and (max-width: 664px) {
  .boxLeftLi:before {
    display: none;

  }

}

.boxLeftLiTitle {
  padding-left: 5%;
  font-size: 2vw;
}
```

```
/* .boxLeftLiActive  */
.boxLeftLi :hover {
  border-radius: 0% 0% -27%;
  padding: 5%;
  width: 100%;
  background-color: #fbf7f7;

}

.boxLeftLiActive:before {
  content: counter(my-counter);
  counter-increment: my-counter;
  font-size: 2.5vw;
  color: #585a5b;
  padding: 3px 7px;
  border-radius: 50px;
  border-style: double;
}
```

```
.boxLeftLiActive:hover {
  border-radius: 0% 0% -27%;
  padding: 5%;
  width: 100%;
  background-color: #fbf7f7;
}




/* =======box-right============== */

.boxRight {
  background-color: #fbf7f7;
  width: 100%;
  margin: 3% 3% 3% 1%;
  border-radius: 2%;
  box-shadow: 10px 10px 10px rgba(92, 147, 147, 0.1);
}

.boxRightUl {
  list-style-type: none;
  display: flex;
  margin-right: 4%;
}

.boxRightli {
  background-color: #d9dbdc;
  margin-left: 2%;
  margin-right: 2%;
  padding: 3%;
  opacity: 0;
  transition: opacity 0.5s ease-in-out;
  box-shadow: 5px 5PX 5PX #585a5b;
}

/* animation right  */
```

```
.boxRightli:nth-child(1) {
  animation: fadeIn 2s ease-in-out 0.5s forwards;
}

.boxRightli:nth-child(2) {
  animation: fadeIn 2s ease-in-out 1s forwards;
}

.boxRightli:nth-child(3) {
  animation: fadeIn 2s ease-in-out 1.5s forwards;
}

@keyframes fadeIn {
  to {
    opacity: 1;
  }
}

.boxRightliImg {
  padding-left: 30%;

  max-width: 6vw;

  max-height: 7vh;

}


.boxRightIiTitle {

  padding-left: 15%;

  font-size: 2vw;

}


.content {

  margin-left: 3VW;



}


.fileUploadForm {

  margin: 3VW 3vw 2vw;

  width: 80%;

  padding: 2%;
  /* padding: 20%; */
  /* background-color: #379cf4; */
  box-shadow: 10px 10PX 10PX #585a5b;
  border-style: double;

}

.fileUploadForm label {
  /* margin-left: 2VW; */
  width: 100%;
  padding: 1%;
  color: black;
  font-size: 2.5vw;


}

.fileUploadForm input {
  padding: 1%;
  font-size: 1.5vw;

  box-shadow: 5px 5PX 5PX #585a5b;

  border-style: double;


}


.buttonContainer {

  /* text-align: center; */

  margin-left: 60%;

  margin-top: 5%;



}


.buttonContainer button {

  padding: 10px 20px;

  font-size: 1.5vw;

  border: none;
  border-radius: 5px;
  margin: -20px 20px 10px 5px;
}

.beforeButton {
  background-color: #3498db;
  color: #fff;
}

.sendButton {
  background-color: #2ecc71;
  color: #fff;
}
```

Form.jsx

```
"use client"

import React, { useState } from 'react'

import styles from './formStyle.css';

// import Swal from 'sweetalert2';

import toastr from 'toastr';

import 'toastr/build/toastr.css';

import Swal from 'sweetalert2';


export const Form = () => {

  const [Gender, setGender] = useState("")

  const [MaritalStatus, setMaritalStatus] = useState("")

  const [Dependents, setDependents] = useState("")

  const [Education, setEducation] = useState("")

  const [Self_Employed, setSelf_Employed] = useState("")

  const [ApplicantIncome, setApplicantIncome] = useState("")

  const [CoapplicantIncome, setCoapplicantIncome] = useState("")
  const [LoanAmount, setLoanAmount] = useState("")
  const [Loan_Amount_Term, setLoan_Amount_Term] =
useState("")
  const [Credit_History, setCredit_History] = useState("")
  const [Total_Income, setTotal_Income] = useState("")
  const [Age, setAge] = useState("")
  const [Experience, setExperience] = useState("")
  const [CD_Account  , setCD_Account ] = useState("")

  function displayResponse(data) {

    console.log(data);
    const message = data.message;

    let icon;
    if (message.includes('Congratulations')) {
      icon = 'success';
    } else {
      icon = 'error';
    }


    Swal.fire({
      icon: icon,
      title: message,
      showConfirmButton: false,
      // timer: 2000 // تعديل المدة الزمنية هنا (بالمللي ثانية)
    });
      }
```

```
//create function to send data to database
  function sendData() {
        let data = { Gender, MaritalStatus, Dependents, Age,
Education, Experience, LoanAmount, Loan_Amount_Term,
CD_Account, CoapplicantIncome, Credit_History, Total_Income,
ApplicantIncome, Self_Employed, }
        console.log(data)
        fetch('http://127.0.0.1:5000/predict', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify(data),
        })
          .then(response => {
            if (!response.ok) {
              throw new Error('Network response was not ok');
            }
            return response.json();
          })
          .then(data => {
                displayResponse(data);
            })
            .catch(error => {
                console.error('There was a problem with your fetch
operation:', error);
                alert('Failed to send data. Please try again later. ');
            });
    }

  return (
    <div className="container">
      <div id="form">

            {/* <input type="text" id="gender" class="amount"
name="gender" placeholder=" Gender" onChange={(e) => {
setGender(e.target.value) }} /> */}
            <input type="number" id="dependents" class="amount"
name="dependents" placeholder=" Number of Dependents"
onChange={(e) => { setDependents(e.target.value) }} />
```

```
            {/* <input type="text" id="education" class="amount"
name="education" placeholder=" Education" onChange={(e) => {
setEducation(e.target.value) }} /> */}

            {/* <input type="text" id="selfEmployed" class="amount"
name="selfEmployed" placeholder="Enter Self Employed"
onChange={(e) => { setSelf_Employed(e.target.value) }} /> */}

            <input type="number" id="applicantIncome"
class="amount" name="applicantIncome" placeholder=" Applicant
Income" onChange={(e) => { setApplicantIncome(e.target.value) }} />

            <input type="number" id="coapplicantIncome"
class="amount" name="coapplicantIncome" placeholder="
Coapplicant Income" onChange={(e) => {
setCoapplicantIncome(e.target.value) }} />

            <input type="number" id="loanAmount" class="amount"
name="loanAmount" placeholder=" Loan Amount" onChange={(e) =>
{ setLoanAmount(e.target.value) }} />

            <input type="number" id="loanAmountTerm"
class="amount" name="loanAmountTerm" placeholder=" Loan
Amount Term" onChange={(e) => {
setLoan_Amount_Term(e.target.value) }} />
```

```
            {/* <input type="number" id="creditHistory"
class="amount" name="creditHistory" placeholder=" Credit History"
onChange={(e) => { setCredit_History(e.target.value) }} /> */}

            <input type="number" id="totalIncome" class="amount"
name="totalIncome" placeholder=" Total Income" onChange={(e) =>
{ setTotal_Income(e.target.value) }} />

            <input type="number" id="age" name="age"
class="amount" placeholder="Enter Age" onChange={(e) => {
setAge(e.target.value) }} />

            <input type="number" id="experience" class="amount"
name="experience" placeholder=" Experience" onChange={(e) => {
setExperience(e.target.value) }} />

            <br />

            {/* ..... */}

            <>

              {/* <label htmlFor="Marital">Gender</label><br /> */}

              <select

                className="select"

                id="setGender"

                onChange={(e) => setGender(e.target.value)}
```

```
                    value={Gender}
                  >
                    <option value="">Select Gender</option>


                    <option value="Male">Male</option>
                    <option value="Female">Female</option>
                    {Gender === "Male" ? "Male" : Gender === "Female" ?
"Female" : "None"}
                  </select>


                </>
                <>
                  {/* <label htmlFor="setEducation"> Education</label><br
/> */}
                  <select className="select" id="setEducation"
onChange={(e) => { setEducation(e.target.value) }}>
                    <option value="">Select Education</option>
                    <option value="Student">Student</option>
                    <option value="Graduate">Graduate</option>
                    {Education === "Student" ? "Student" : Gender ===
"Graduate" ? "Graduate" : "None"}


                  </select>
              </>
              <>
                  {/* <label htmlFor="Marital"> Self Employed</label><br
/> */}
                  <select className="select" id="selfEmployed"
onChange={(e) => { setSelf_Employed(e.target.value) }}>
                    <option value="">Select Self Employed</option>


                    <option value="1">Yas</option>
                    <option value="0">No</option>
                    {Self_Employed === "1" ? "Yas" : Gender === "0" ? "No"
: "None"}


                  </select>
              </>
              <>
```

```
                {/* <label htmlFor="MaritalStatus"> Marital
Status</label><br /> */}
                <select className="select" id="maritalStatus"
onChange={(e) => { setMaritalStatus(e.target.value) }}>
                    <option value="">Select Marital Statusy</option>


                    <option value="Single">Single</option>
                    <option value="Married">Married</option>
                    {MaritalStatus === "Single" ? "Single" : Gender ===
"Married" ? "Married" : "None"}


                </select>
            </>
            <>
                {/* <label htmlFor="Credit_History">Credit
History</label><br /> */}
                <select className="select" id="Credit_History"
onChange={(e) => { setCredit_History(e.target.value)
}}  value={Credit_History}>
                    <option value="">Select Credit History</option>
                    <option value="1">Good</option>

                    <option value="0">Bad</option>

                    {Credit_History === "1" ? "Good" : Gender === "0" ?
"Bad" : "None"}

                  </select>

                </>

                <>

                  {/* <label for="cdAccount" >CD Account:</label> */}

                  {/* <br/> */}

                  <select id="CD_Account" className="select"
name="cdAccount" onChange={(e) => { setCD_Account
(e.target.value) }}value={CD_Account }>

                    <option value=""> CD Account</option>

                    <option value="1">Yes</option>

                    <option value="0">No</option>

                    {CD_Account  === "1" ? "Yes" : Gender === "0" ? "No" :
"None"}


                  </select>

                </>
                <div className="buttonContainer">

                    <button className="sendButton" onClick={() => {
sendData() }}>Submit</button>

                </div>  </div>

            <div id="response"></div>

        </div>

    )

}

export default Form;
```

formStyle.css

```
.flex1 {
   display: flex;
}

@media screen and (min-width: 1210px) {}

@media screen and (max-width: 664px) {
   .flex1 {
      display: block;
   }
}

.flex1 div {
   margin-top: 1vw;
   margin-right: 80px;
   font-size: 1.55vw;
```

```
    font-weight: bold;
}

.select {
    font-size: 1.5vw;
    width: 250px;
    text-align: center;
    margin: 10px 30px 0px 0px;
    padding: 6px 2px;
    border-radius: 17px;

}

.buttonContainer {
    /* text-align: center; */
    margin-left: 60%;
    margin-top: 5%;
```

```
 }



.select1 {



   margin: 30px 0px 15px 0px;

   padding: 6px 2px;

   border-radius: 17px;


}

.select {

   font-size: 1.5vw;

   width: 260px;

   text-align: center;

   margin: 10px 10px 10px 10px;

   padding: 6px 2px;

   border-radius: 17px;
```

```
}


@media screen and (max-width: 664px) {

    .select {

        font-size: 5vw;

        width: 250px;

    }



}

.responseBox {

    width: 20vw;

    height: 15vw;

    background-color: #3498db

    color: white;

    font-weight: bold;

    padding: 20px;

    border: 5px solid #2980b9
    box-shadow: 0px 0px 10px 5px rgba(0, 0, 0, 0.5);



  .buttonContainer button {
    padding: 10px 20px;
    font-size: 1.5vw;
    border: none;
    border-radius: 5px;
    margin: -20px 20px 10px 5px;
  }

.sendButton {
    background-color: #2ecc71;
    color: #fff;
  }
.select1{
    font-size: 1.5vw;
display:flex;
padding: 20px 12px;
}

.Education {
   display: inline-block;
   margin-bottom: 8px;
   font-weight: bold;
   margin-right: 5px;

}

.Education2 {
   margin-right: 15px;

}

.amount,
.Dependents,
.ApplicantIncome,
.coapplicantIncome,
.Loan {


   width: 250px;

   height: 35px;

   border-radius: 50px;

   color: rgb(94, 91, 91);

   border: 1px solid gray;

   text-align: center;

   font-size: large;

   margin: 1vw;


}


.totalInput {

   display: flex;


}


@media screen and (max-width: 664px) {
    .totalInput {
        display: block;
        width: 50%;


    }
}





.duration {
    font-size: 1.75vw;
    font-weight: bold;
    margin: 2% 0% 0%;
}

.rido {
    display: flex;
    color: rgb(67, 64, 64);
```

```
}


@media screen and (max-width: 664px) {
    .rido {
        display: block;
    }
}


.rido div {
    margin-right: 50px;
}
```

main.py

```
        from flask import Flask, request, jsonify
```

```
import joblib

import pandas as pd


app = Flask(__name__)

model = joblib.load('final_neural_network_model.pkl')


from flask_cors import CORS

CORS(app)  # Enable CORS for all origins


# Importing the required libraries


import sqlite3


import hashlib


import jwt


app.config['SECRET_KEY'] =
'0A1S2AD1S3A54C5ASC144SA65F41AS65F4'
```

```
# Generate JWT token
def generate_tokens(user_email):
    access_token = jwt.encode({'email': user_email},
app.config['SECRET_KEY'], algorithm='HS256')
    refresh_token = jwt.encode({'email': user_email},
app.config['SECRET_KEY'], algorithm='HS256')
    return access_token, refresh_token


# Verify JWT token
def verify_token(token):
    try:
        decoded_token = jwt.decode(token, app.config['SECRET_KEY'],
algorithms=['HS256'])
        return decoded_token['email']
    except jwt.ExpiredSignatureError:
        return None  # Token expired
    except jwt.InvalidTokenError:
        return None  # Invalid token



## Sign Up API using sqlite3 database creating a user table with
email, phone, password, name
@app.route('/signup', methods=['POST'])
def signup():
    data = request.get_json()

    if not data:
        return jsonify({'message': 'Invalid request'}), 400

    name = data.get('name')
    email = data.get('email')
    phone = data.get('phone')
    password = data.get('password')

    # hash the password
    password = password.encode('utf-8')
    password = hashlib.sha256(password).hexdigest()


    conn = sqlite3.connect('sqlite.db')
    cursor = conn.cursor()


    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            email TEXT PRIMARY KEY NOT NULL UNIQUE,
            phone TEXT NOT NULL UNIQUE,
            password TEXT,
            name TEXT
        )
    ''')


    cursor.execute("INSERT INTO users (email, phone, password,
name) VALUES (?, ?, ?, ?)", (email, phone, password, name))
    conn.commit()


    return jsonify({'message': 'User registered successfully'})
```

```
## Login API using sqlite3 database


@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()


    if not data:
        return jsonify({'message': 'Invalid request'}), 400


    email = data.get('email')
    password = data.get('password')


    # hash the password
    password = password.encode('utf-8')
    password = hashlib.sha256(password).hexdigest()
    conn = sqlite3.connect('sqlite.db')

    cursor = conn.cursor()




    cursor.execute("SELECT * FROM users WHERE email=? AND
password=?", (email, password))

    user = cursor.fetchone()




    if not user:

        return jsonify({'message': 'Invalid credentials'}), 401



    # Generate JWT token

    access_token, refresh_token = generate_tokens(user[0])


    return jsonify(
        {
        'message': 'Login successful',
        'email': user[0],
        'phone': user[1],
        'name': user[3],

        "access_token": access_token,
        "refresh_token": refresh_token,
        }
    )




# Define function to encode categorical variables
def encode_categorical(data):

    # Define a dictionary to map categorical variables to binary values
    binary_mapping = {
        "Gender": {"Male": 1, "Female": 0},
        "MaritalStatus": {"single": 1, "married": 0},
        "Education": {"Graduate": 1, "Not Graduate": 0},
        "Self_Employed": {"Yes": 1, "No": 0}

    }
    for key, value in data.items():
        if key in binary_mapping:
            data[key] = binary_mapping[key].get(value, 0)  # Convert
value to 0 or 1 if applicable
    return data

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()  # Get data from the request

    # Extract data from the request
    gender = data.get('Gender')
    marital_status = data.get('MaritalStatus')
```

```
dependents = data.get('Dependents')

education = data.get('Education')

self_employed = data.get('Self_Employed')

applicant_income = data.get('ApplicantIncome')

coapplicant_income = data.get('CoapplicantIncome')

loan_amount = data.get('LoanAmount')

loan_amount_term = data.get('Loan_Amount_Term')

credit_history = data.get('Credit_History')

total_income = data.get('Total_Income')

age = data.get('Age')

experience = data.get('Experience')

cd_account = data.get('CD Account')


# Connect to the database

conn = sqlite3.connect('sqlite.db')


# create the table if it does not exist
```

```
cursor = conn.cursor()


cursor.execute(
  '''
  CREATE TABLE IF NOT EXISTS loan_data
  (
    Gender TEXT NULL,
    MaritalStatus TEXT NULL,
    Dependents TEXT NULL,
    Education TEXT NULL,
    Self_Employed TEXT NULL,
    ApplicantIncome REAL NULL,
    CoapplicantIncome REAL NULL,
    LoanAmount REAL NULL,
    Loan_Amount_Term REAL NULL,
    Credit_History REAL NULL,
    Total_Income REAL NULL,
    Age REAL NULL,
    Experience REAL NULL,
            CD_Account REAL NULL
        )
        '''
    )

    conn.commit()

    try:
        # Insert data into the database
        cursor.execute(
            """
            INSERT INTO loan_data (
                Gender, MaritalStatus, Dependents, Education,
Self_Employed,
                ApplicantIncome, CoapplicantIncome, LoanAmount,
Loan_Amount_Term,
                Credit_History, Total_Income, Age, Experience, CD_Account
            )
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            """,
            (
                gender, marital_status, dependents, education,
self_employed, applicant_income,
                coapplicant_income, loan_amount, loan_amount_term,
credit_history, total_income,
                age, experience, cd_account
            )
        )

        conn.commit()

    except Exception as e:
        print(e)
        return jsonify({'message': 'An error occurred while saving data to
the database'}), 500

    finally:
        conn.close()
    # Encode categorical variables
    data = encode_categorical(data)

    # Convert all values to numeric types
    data = {key: float(value) if isinstance(value, int) or isinstance(value,
float) else value for key, value in data.items()}

    # Convert the dictionary into a DataFrame
    input_df = pd.DataFrame([data])

    # Make predictions using the loaded model
    prediction = model.predict(input_df)

    if prediction[0] == 0:
        response_message = 'Sorry, Rejected'



    else:
        response_message = 'Congratulations! Approved'

    return jsonify(message=response_message)




@app.route('/', methods=['GET'])
def Hello():
    return jsonify({'message': 'Hello World'})
app.run(host='0.0.0.0', port=5000)
if __name__ == '__main__':
    app.run(debug=True)
```

Machine Learning

8. Machine learning

8.1. Introduction

The machine learning models used in this project are:

1. Logistic Regression
2. K-Nearest Neighbour (KNN)
3. Support Vector Machine (SVM)
4. Naive Bayes
5. Decision Tree
6. Random Forest
7. Xgboost Boost
8. Neural Network

There are **16 variables** in this data set:

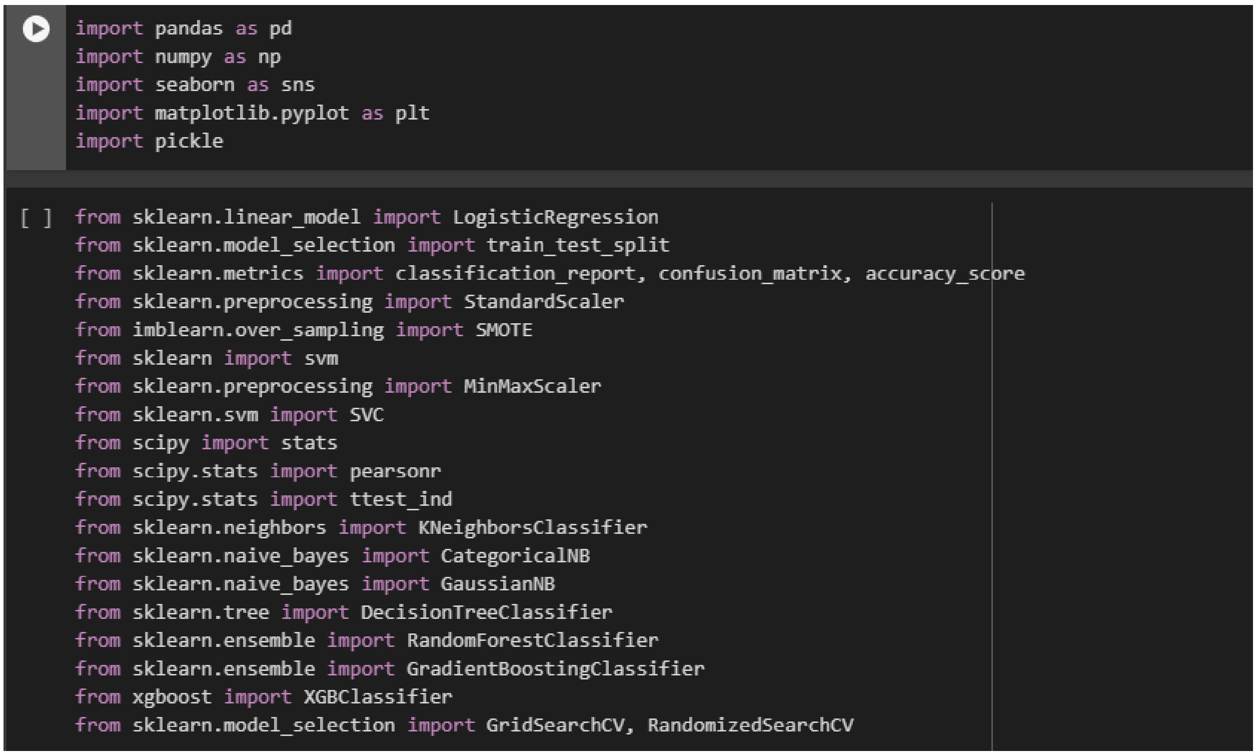
- **11 categorical** variables,
- **4 continuous** variables, and
- **1** variable to accommodate the loan ID.

The following is the **structure of the data set**.

Variable Name	Description
Loan_ID	Loan reference number (unique ID)
Gender	Applicant gender (Male or Female)
MaritalStatus	Applicant marital status (single or Married)
Dependents	Number of family members
Education	Applicant education/qualification (student or graduate)
Self_Employed	Applicant employment status (yes for self-employed, no for employed/others)
ApplicantIncome	Applicant's monthly salary/income
CoapplicantIncome	Additional applicant's monthly salary/income
LoanAmount	Loan amount

Variable Name	Description
Loan_Amount_Term	**The loan's repayment period (in days)**
Credit_History	**Records of previous credit history (bad credit history, good credit history**
Age	**Age of the person**
Experience	**The number of years of work experience of the person**
Total_Income	**Applicant's monthly salary/income + Additional applicant's monthly salary/income**
CD Account	**Does he have a certificate of deposit or not ? (have , No)**
Loan_Status	**Status of loan (Y: accepted, N: not accepted)**

8.2. Importing Libraries



```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pickle
```

```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
    from sklearn.preprocessing import StandardScaler
    from imblearn.over_sampling import SMOTE
    from sklearn import svm
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.svm import SVC
    from scipy import stats
    from scipy.stats import pearsonr
    from scipy.stats import ttest_ind
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.naive_bayes import CategoricalNB
    from sklearn.naive_bayes import GaussianNB
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.ensemble import GradientBoostingClassifier
    from xgboost import XGBClassifier
    from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

pandas: A library for data manipulation and analysis.

numpy: A library for numerical operations and array manipulation.

seaborn: A library for statistical data visualization.

matplotlib.pyplot: A library for creating static, animated, and interactive visualizations in Python.

pickle: A module for serializing and deserializing Python objects.

sklearn.linear_model.LogisticRegression: A class for logistic regression models.

sklearn.model_selection.train_test_split: A function for splitting data into training and testing sets.

sklearn.metrics.classification_report: A function for generating a classification report of a model's performance.

sklearn.metrics.confusion_matrix: A function for creating a confusion matrix to evaluate the performance of a classification model.

sklearn.metrics.accuracy_score: A function for calculating the accuracy of a classification model.

sklearn.preprocessing.StandardScaler: A class for standardizing features by removing the mean and scaling to unit variance.

imblearn.over_sampling.SMOTE: A class for oversampling the minority class in imbalanced datasets.

sklearn.svm: A module for Support Vector Machine algorithms.

sklearn.preprocessing.MinMaxScaler: A class for scaling features to a specified range.

scipy.stats: A module for statistical functions and tests.

sklearn.neighbors.KNeighborsClassifier: A class for k-nearest neighbors classification models.

sklearn.naive_bayes.CategoricalNB: A class for categorical naive Bayes models.

sklearn.naive_bayes.GaussianNB: A class for Gaussian naive Bayes models.

sklearn.tree.DecisionTreeClassifier: A class for decision tree classification models.

sklearn.ensemble.RandomForestClassifier: A class for random forest classification models.

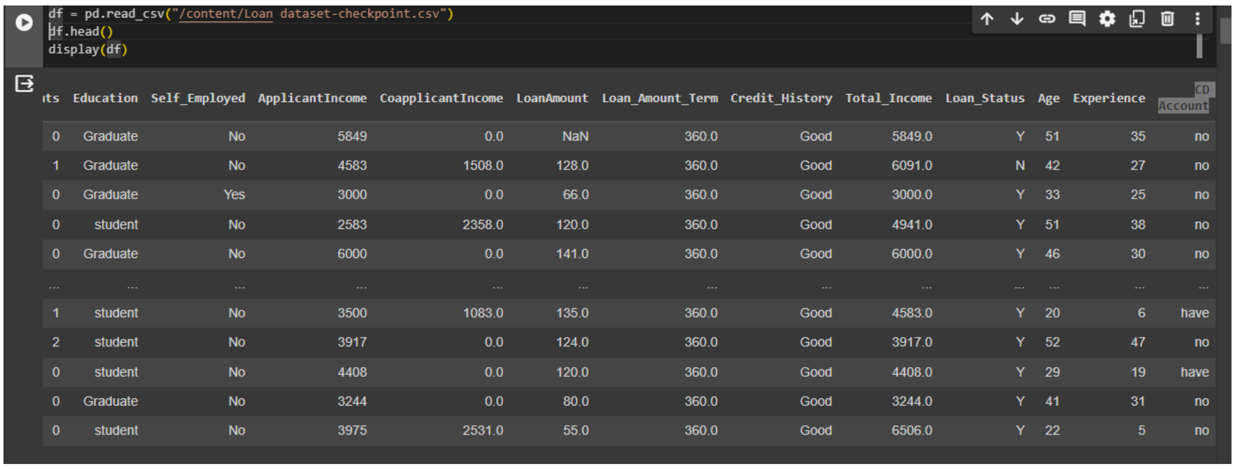
sklearn.ensemble.GradientBoostingClassifier: A class for gradient boosting classification models.

xgboost.XGBClassifier: A class for extreme gradient boosting classification models.

sklearn.model_selection.GridSearchCV: A class for performing grid search cross-validation to find the best hyperparameters for a model.

sklearn.model_selection.RandomizedSearchCV: A class for performing randomized search cross-validation to find the best hyperparameters for a model

8.3. Reading Data Set

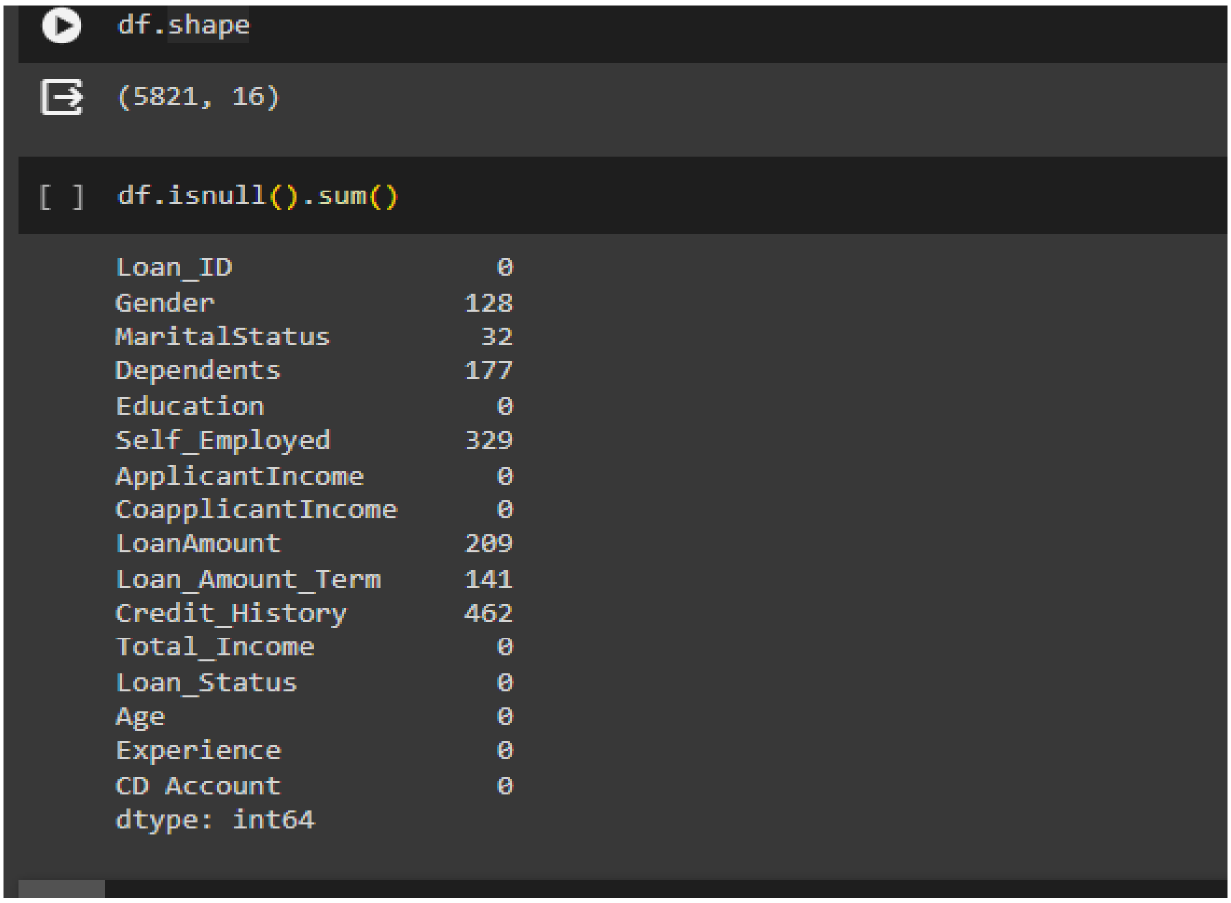


```
df = pd.read_csv("/content/Loan dataset-checkpoint.csv")
df.head()
display(df)
```

its	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Total_Income	Loan_Status	Age	Experience	CD Account
0	Graduate	No	5849	0.0	NaN	360.0	Good	5849.0	Y	51	35	no
1	Graduate	No	4583	1508.0	128.0	360.0	Good	6091.0	N	42	27	no
0	Graduate	Yes	3000	0.0	66.0	360.0	Good	3000.0	Y	33	25	no
0	student	No	2583	2358.0	120.0	360.0	Good	4941.0	Y	51	38	no
0	Graduate	No	6000	0.0	141.0	360.0	Good	6000.0	Y	46	30	no
...	...	...	...	...	...	...	...	...	...	...	...	...
1	student	No	3500	1083.0	135.0	360.0	Good	4583.0	Y	20	6	have
2	student	No	3917	0.0	124.0	360.0	Good	3917.0	Y	52	47	no
0	student	No	4408	0.0	120.0	360.0	Good	4408.0	Y	29	19	have
0	Graduate	No	3244	0.0	80.0	360.0	Good	3244.0	Y	41	31	no
0	student	No	3975	2531.0	55.0	360.0	Good	6506.0	Y	22	5	no

8.4. Data Cleaning

Data cleaning is a process to identify and correct errors in the dataset that may negatively impact our predictive model. We'll find the "null" values of every column as an initial step to data clean



```
df.shape
```

```
(5821, 16)
```

```
[ ] df.isnull().sum()
```

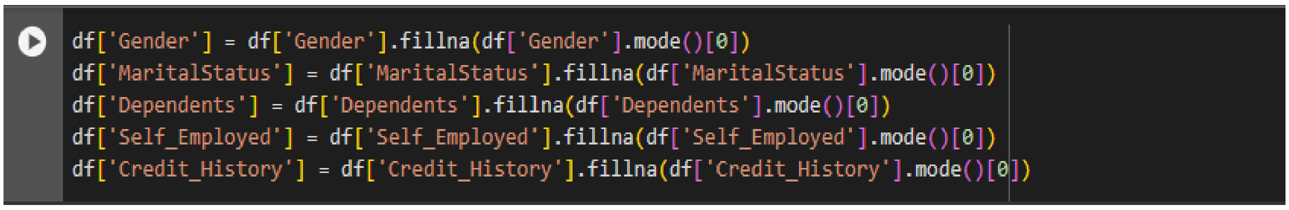
```
Loan_ID                0
Gender               128
MaritalStatus         32
Dependents           177
Education              0
Self_Employed        329
ApplicantIncome        0
CoapplicantIncome      0
LoanAmount           209
Loan_Amount_Term     141
Credit_History       462
Total_Income           0
Loan_Status            0
Age                    0
Experience             0
CD Account             0
dtype: int64
```

We observe that there are “128” missing values in “Gender”, “32” in “Marital status”, “177” in “Dependents”, “329” in “Self_Employed”, “209” in “Loan_Amount”, “141” in “Loan_Amount_Term” and “462” in “Credit_History”.

The missing values of the numerical and categorical features are “missing at random (MAR)” i.e. the data is not missing in all the observations but only within sub-samples of the data.

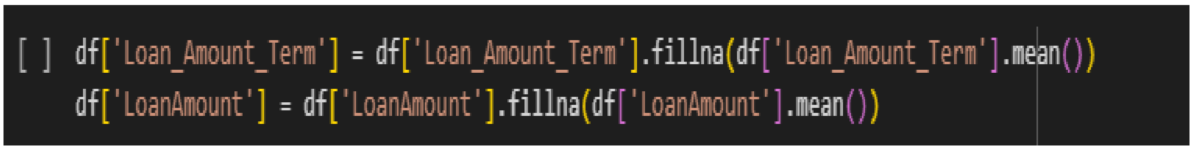
So the missing values of the numerical features should be filled with “mean” and the categorical features with “mode” i.e. the most frequently occurring values. We use Pandas “fillna()” function for imputing the missing values as the estimate of “mean” gives us the central tendency without the extreme values and “mode” is not affected by extreme values; moreover both provide neutral output.

fill the missing values for categorical terms-mode



```
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['MaritalStatus'] = df['MaritalStatus'].fillna(df['MaritalStatus'].mode()[0])
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mode()[0])
```

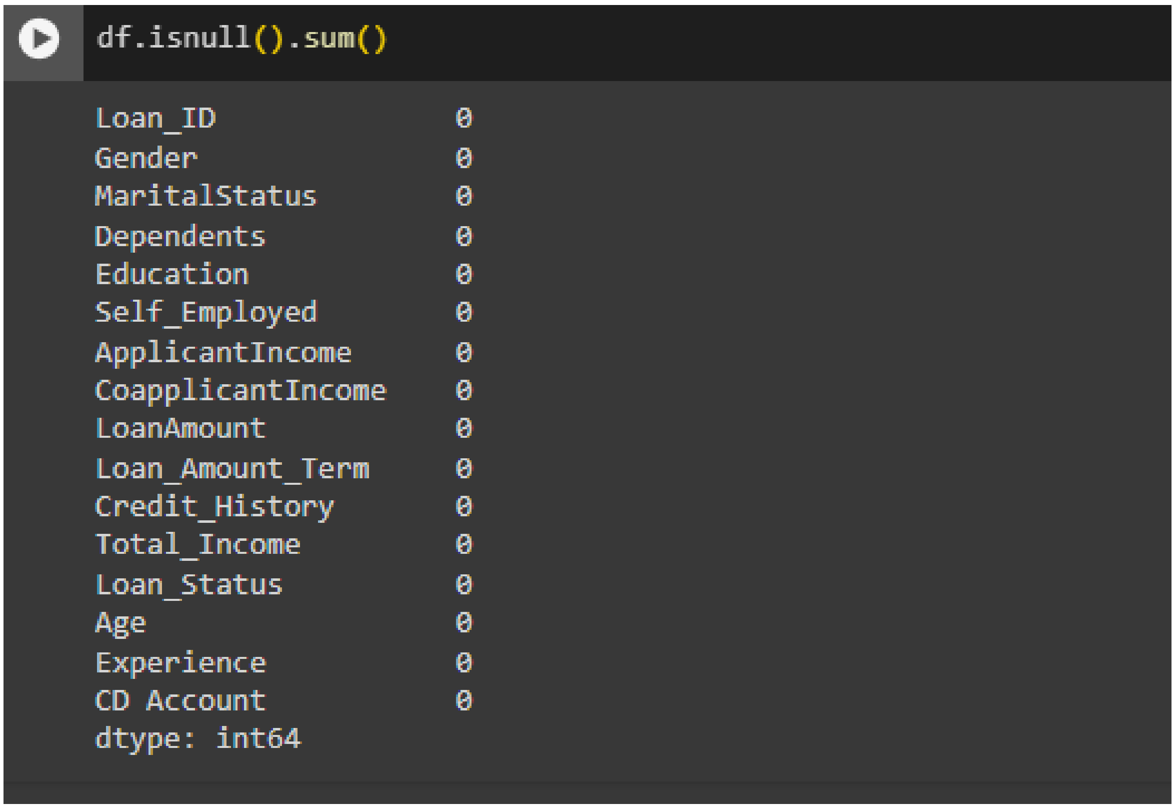
fill the missing values for numerical terms-mean



```
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
```

Let’s check the “null” values again to ensure that there are no missing values as it will lead us to incorrect results.

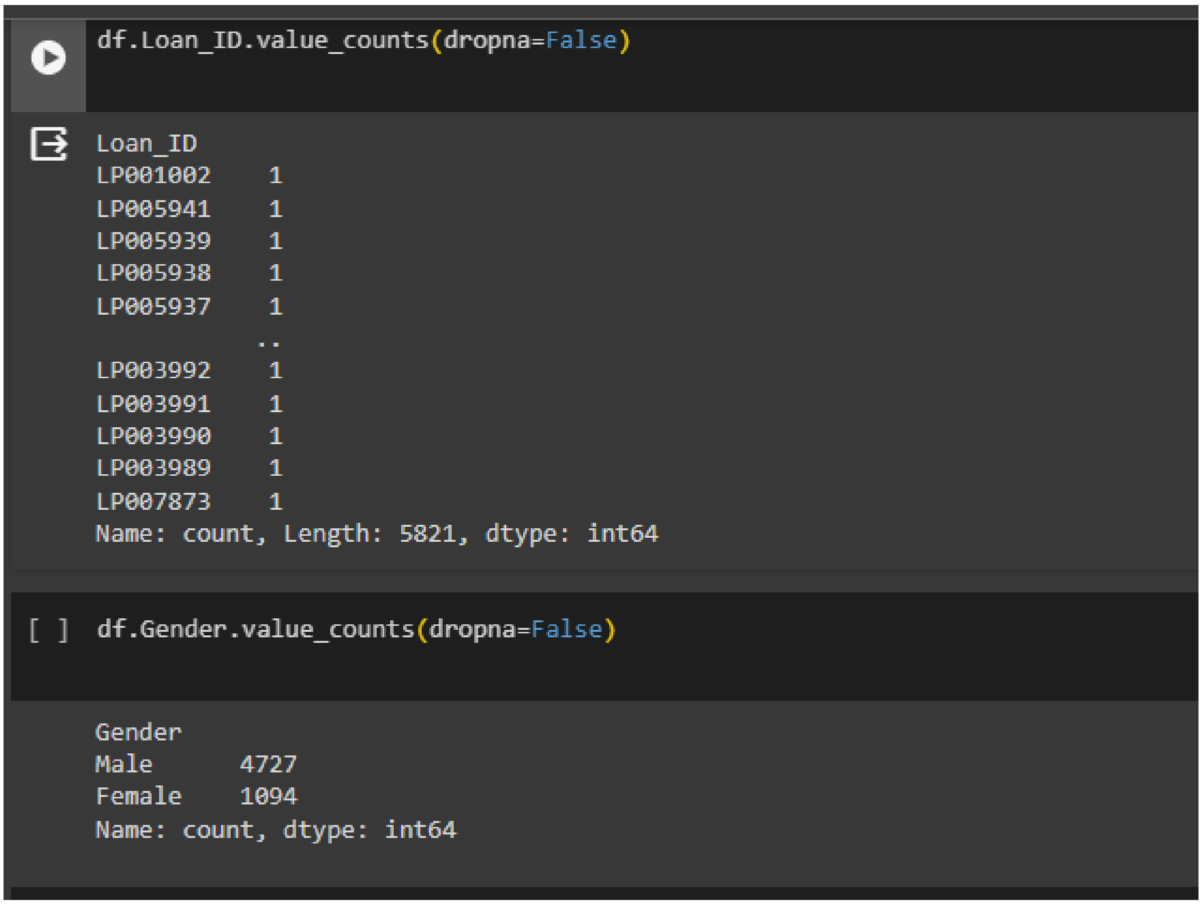
Note : loan_ID,Education,Applicant Income,coapplican income,total income , Age,Experience ,CD Account,loan_status don't have miss value so don't need to clean



```
df.isnull().sum()
```

```
Loan_ID              0
Gender               0
MaritalStatus        0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Total_Income         0
Loan_Status          0
Age                  0
Experience           0
CD Account           0
dtype: int64
```

This section will perform data exploration of "raw" data set that has been imported.The first type of variable that I will explore is categorical variable



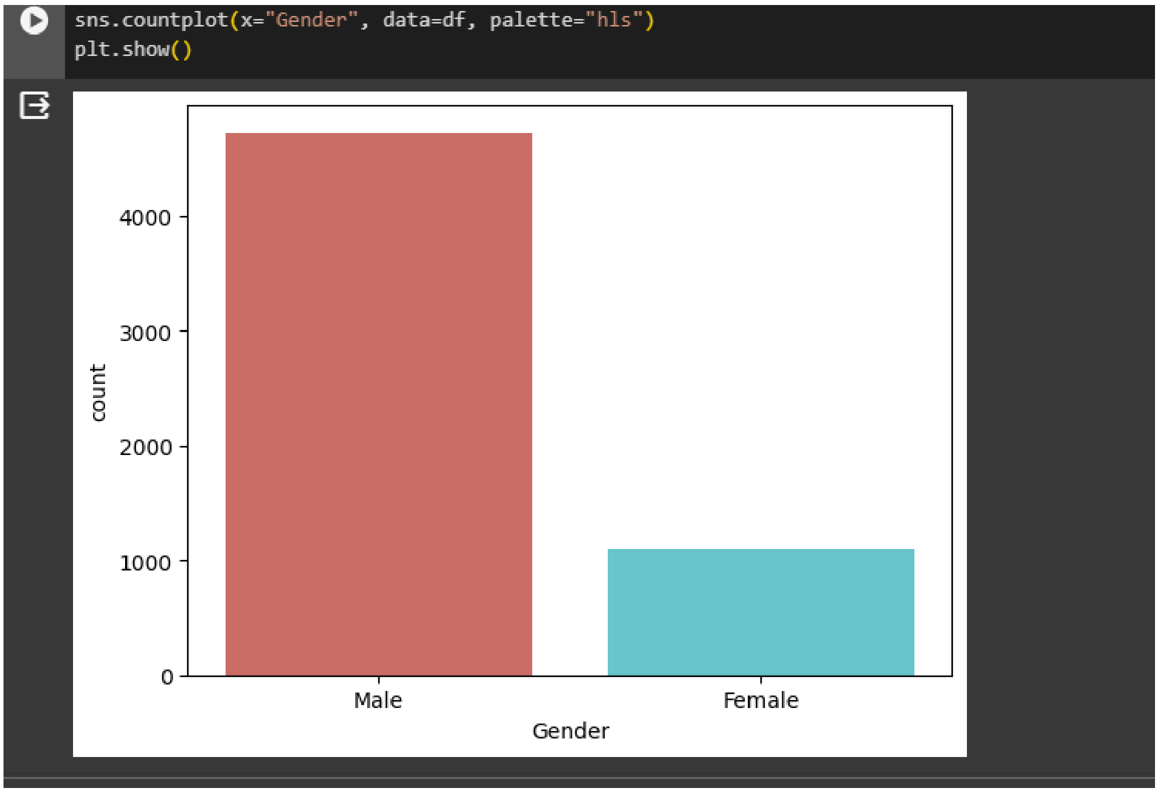
```
df.Loan_ID.value_counts(dropna=False)
```

```
Loan_ID
LP001002    1
LP005941    1
LP005939    1
LP005938    1
LP005937    1
           ..
LP003992    1
LP003991    1
LP003990    1
LP003989    1
LP007873    1
Name: count, Length: 5821, dtype: int64
```

```
[ ] df.Gender.value_counts(dropna=False)
```

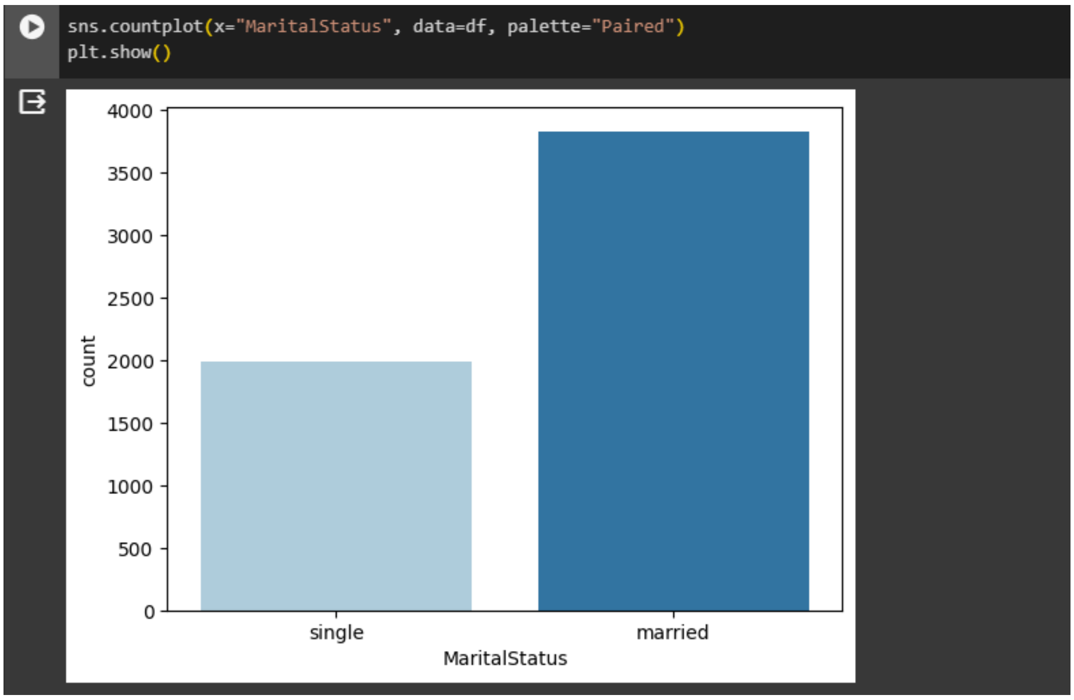
```
Gender
Male      4727
Female    1094
Name: count, dtype: int64
```

It can be seen that there are 5821 unique ID in the dataset



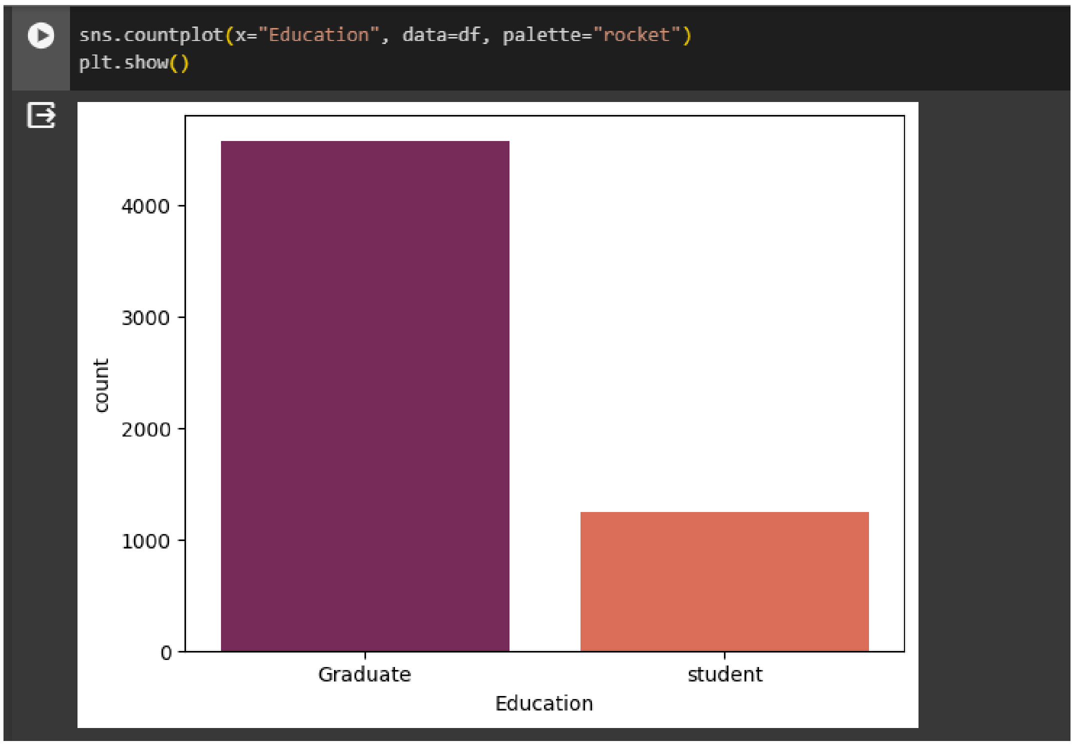
```
sns.countplot(x="Gender", data=df, palette="hls")
plt.show()
```

From the results above, the number of male applicants is higher compared to female applicants.



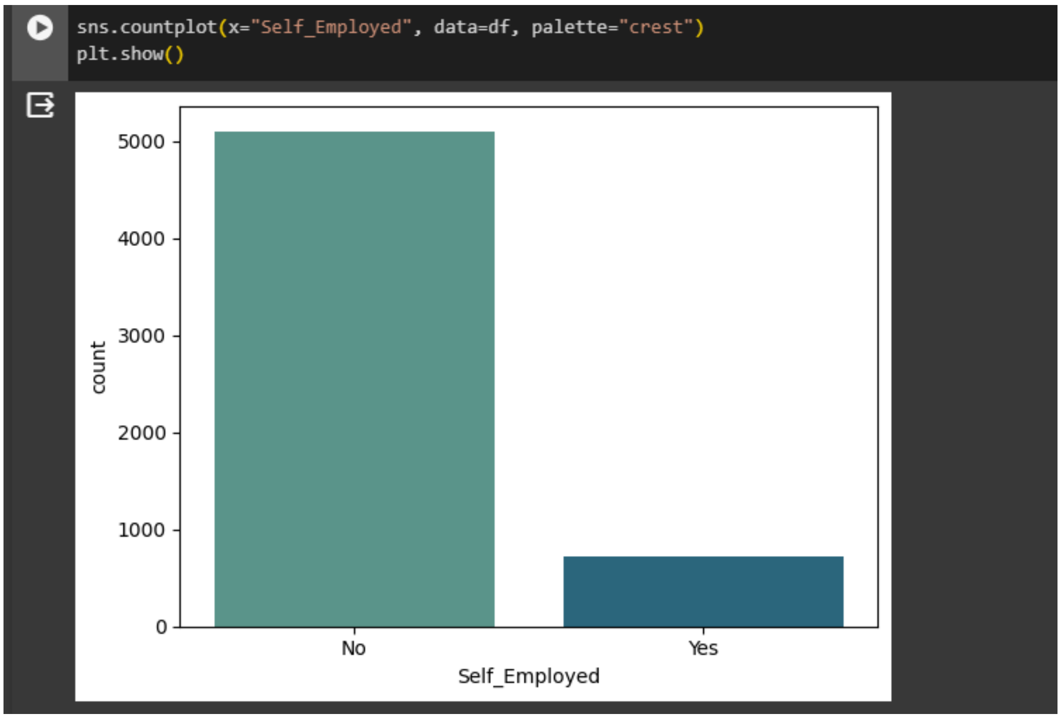
```
sns.countplot(x="MaritalStatus", data=df, palette="Paired")
plt.show()
```

The number of applicants that has been married is higher compared to applicants that single



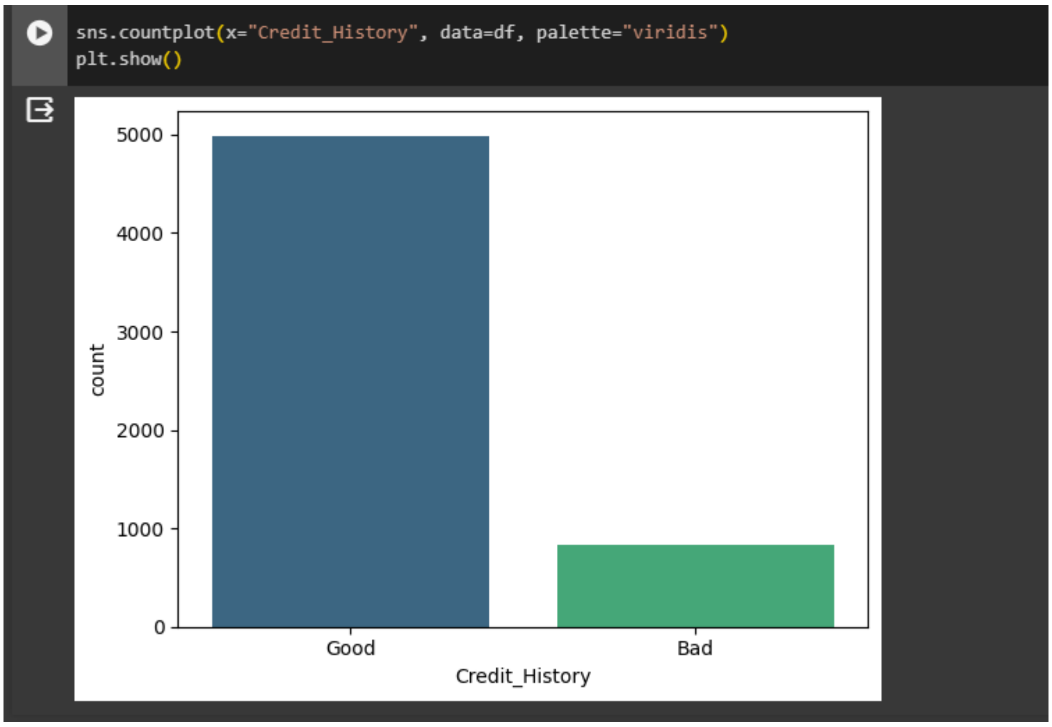
```
sns.countplot(x="Education", data=df, palette="rocket")
plt.show()
```

The number of applicants that has been graduated is higher compared to applicants that has student



```
sns.countplot(x="Self_Employed", data=df, palette="crest")
plt.show()
```

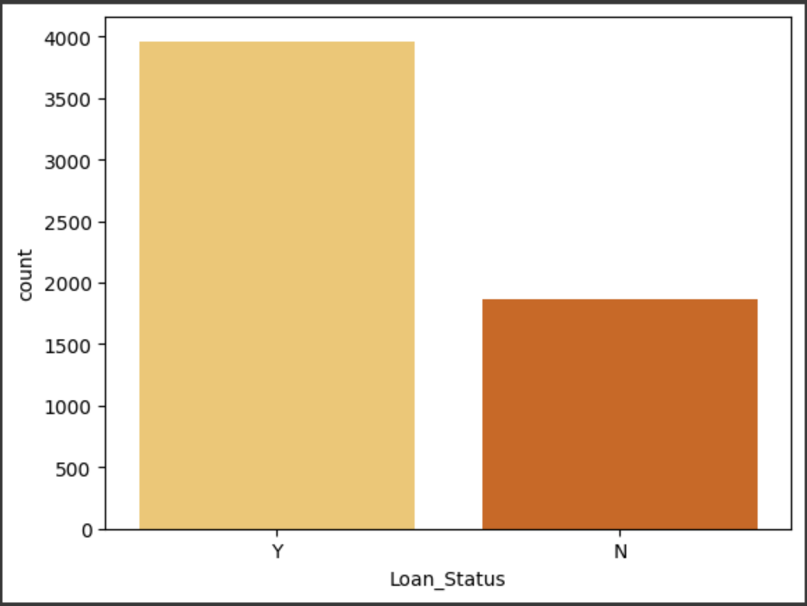
The number of applicants that are not self employed is higher compared to applicants that are self employed



```
sns.countplot(x="Credit_History", data=df, palette="viridis")
plt.show()
```

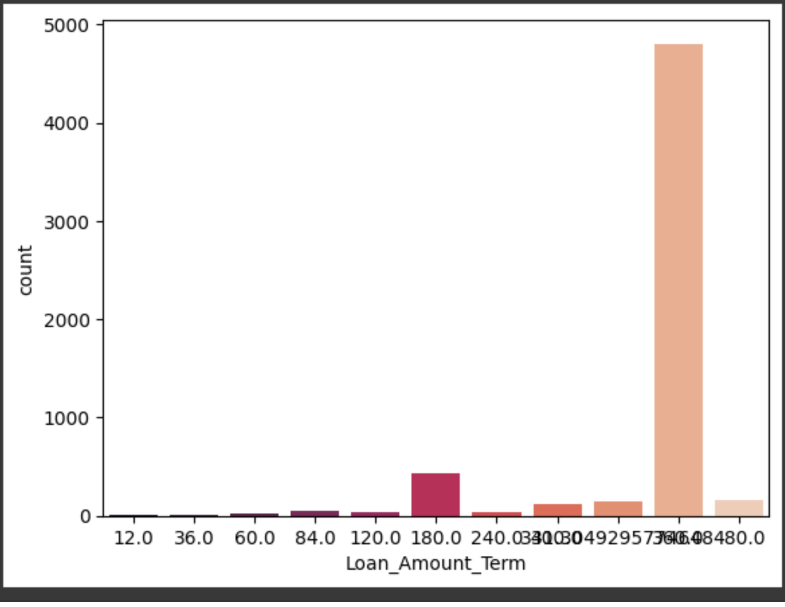
The number of applicants that have good credit history is higher compared to applicants that have bad credit history

```
sns.countplot(x="Loan_Status", data=df, palette="YlOrBr")
plt.show()
```

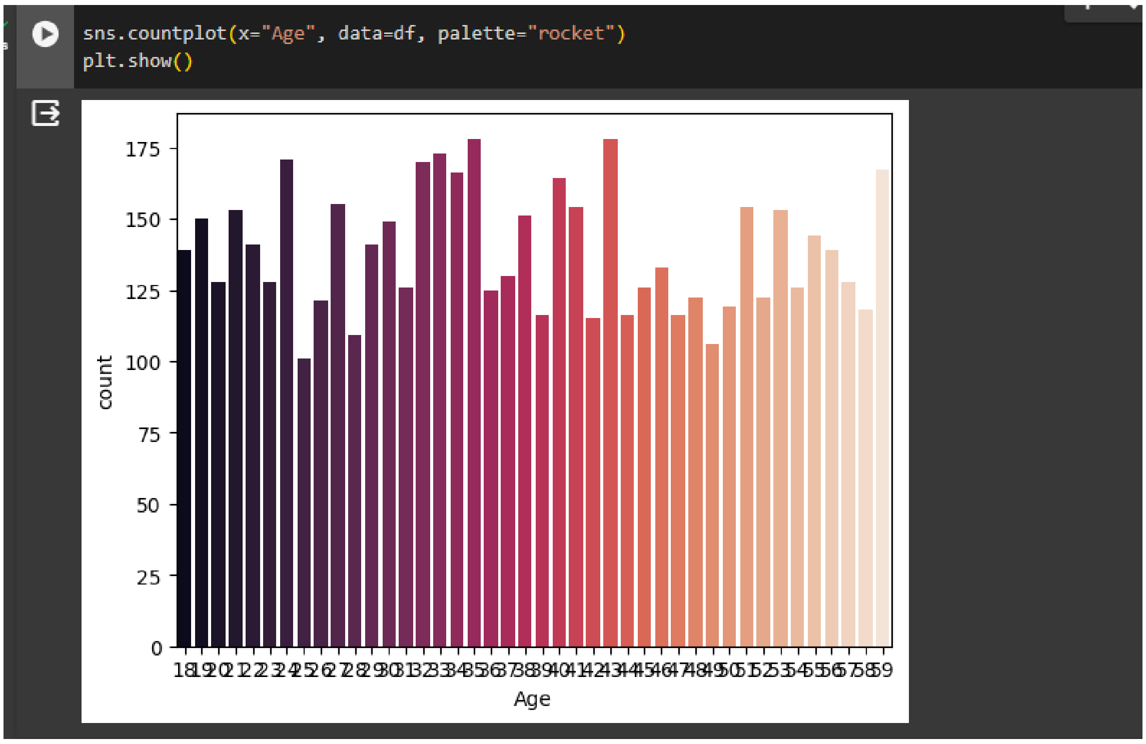


The number of approved loans is higher compared to rejected loans

```
sns.countplot(x="Loan_Amount_Term", data=df, palette="rocket")
plt.show()
```

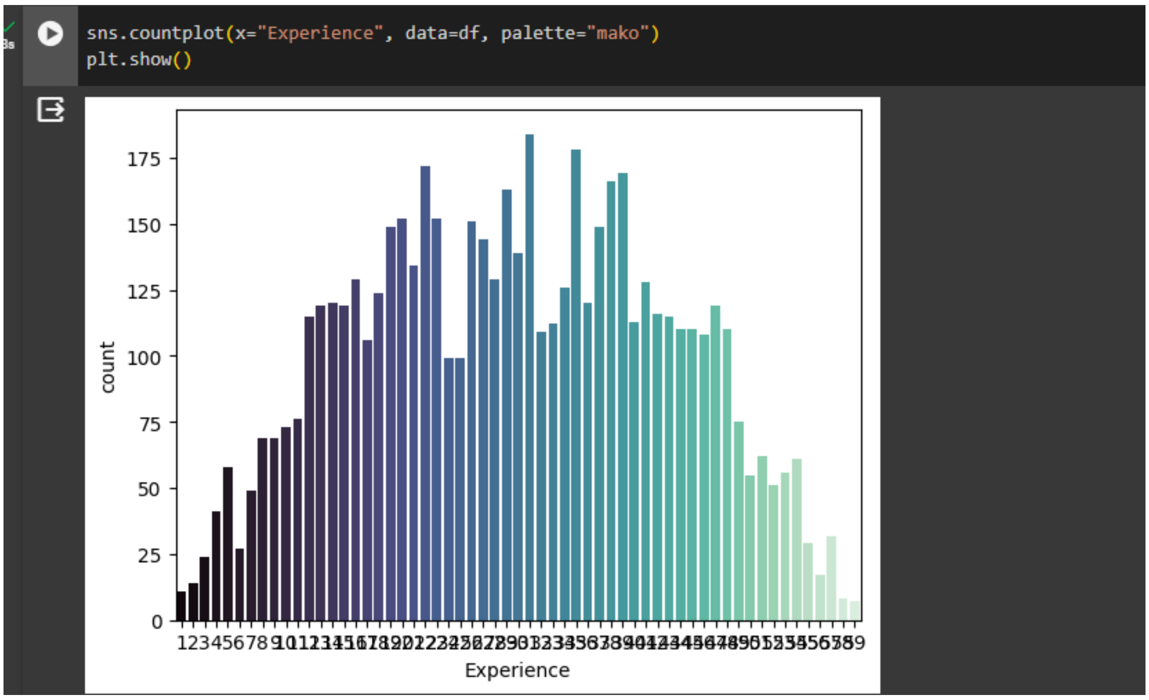


As can be seen from the results, the 360 days loan duration is the most popular compared to others.



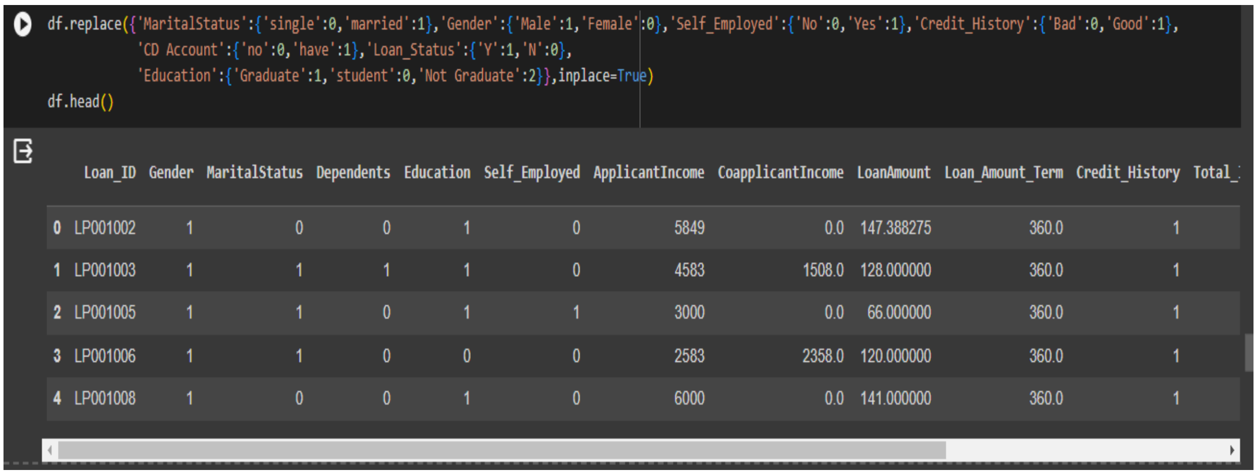
```
sns.countplot(x="Age", data=df, palette="rocket")
plt.show()
```

As can be seen from the results, the 35-45 Ages is the most popular compared to others.



```
sns.countplot(x="Experience", data=df, palette="mako")
plt.show()
```

the most common experience level in the given dataset is 31



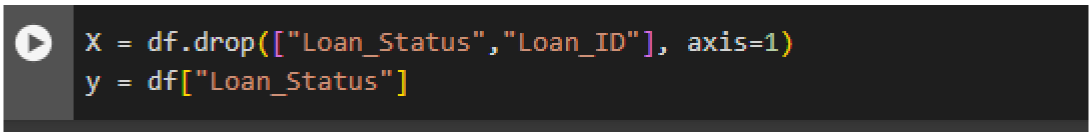
```python
df.replace({'MaritalStatus':{'single':0,'married':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1},'Credit_History':{'Bad':0,'Good':1},
           'CD Account':{'no':0,'have':1},'Loan_Status':{'Y':1,'N':0},
           'Education':{'Graduate':1,'student':0,'Not Graduate':2}},inplace=True)
df.head()
```

```
    Loan_ID  Gender  MaritalStatus  Dependents  Education  Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  Total_
0  LP001002       1              0           0          1              0             5849                0.0  147.388275             360.0               1
1  LP001003       1              1           1          1              0             4583             1508.0  128.000000             360.0               1
2  LP001005       1              1           0          1              1             3000                0.0   66.000000             360.0               1
3  LP001006       1              1           0          0              0             2583             2358.0  120.000000             360.0               1
4  LP001008       1              0           0          1              0             6000                0.0  141.000000             360.0               1
```

To make machine learning work well, we replace the text values in binary form

8.5. Features Separating

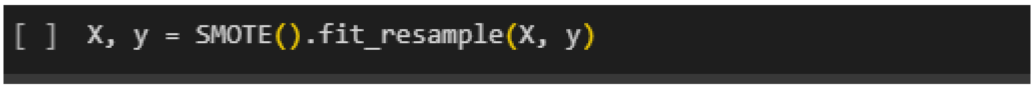
Dependent features (Loan_Status) will be seperated from independent features.



```python
X = df.drop(["Loan_Status","Loan_ID"], axis=1)
y = df["Loan_Status"]
```

SMOTE Technique

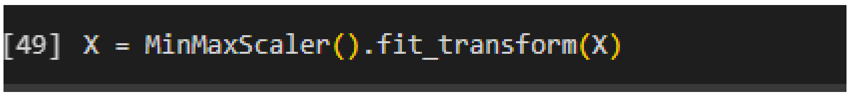
In previous exploration, it can be seen that the number between approved and rejected loan is imbalanced. In this section, oversampling technique will be used to avoid overfitting,



```
[ ]  X, y = SMOTE().fit_resample(X, y)
```

8.6. Data Normalization

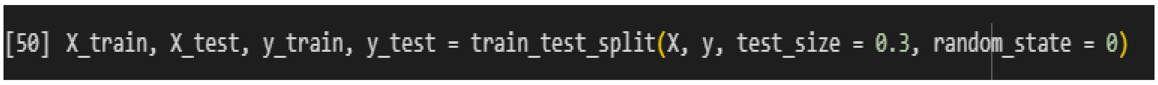
data normalization will be performed to normalize the range of independent variables or features of data.



```
[49]  X = MinMaxScaler().fit_transform(X)
```

8.7. Splitting Data Set

The data set will be split into 70% train and 30% test.



```
[50] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

8.8. Models

8.8.1. Xgboost Boost

using the XGBoost library to train a classification model and make predictions. Here's a breakdown of what each line of code does:

1. `from xgboost import XGBClassifier`: This line imports the `XGBClassifier` class from the XGBoost library. XGBoost is a popular machine learning algorithm that is known for its effectiveness in various types of machine learning tasks, including classification.

2. `XGB = XGBClassifier()`: This line creates an instance of the `XGBClassifier` class and assigns it to the variable `XGB`. This instance represents the classification model that will be trained.

3. `XGB.fit(X_train, y_train)`: This line trains the `XGBClassifier` model using the training data `X_train` and the corresponding labels `y_train`. The model learns to map the input features in `X_train` to the target labels in `y_train`.

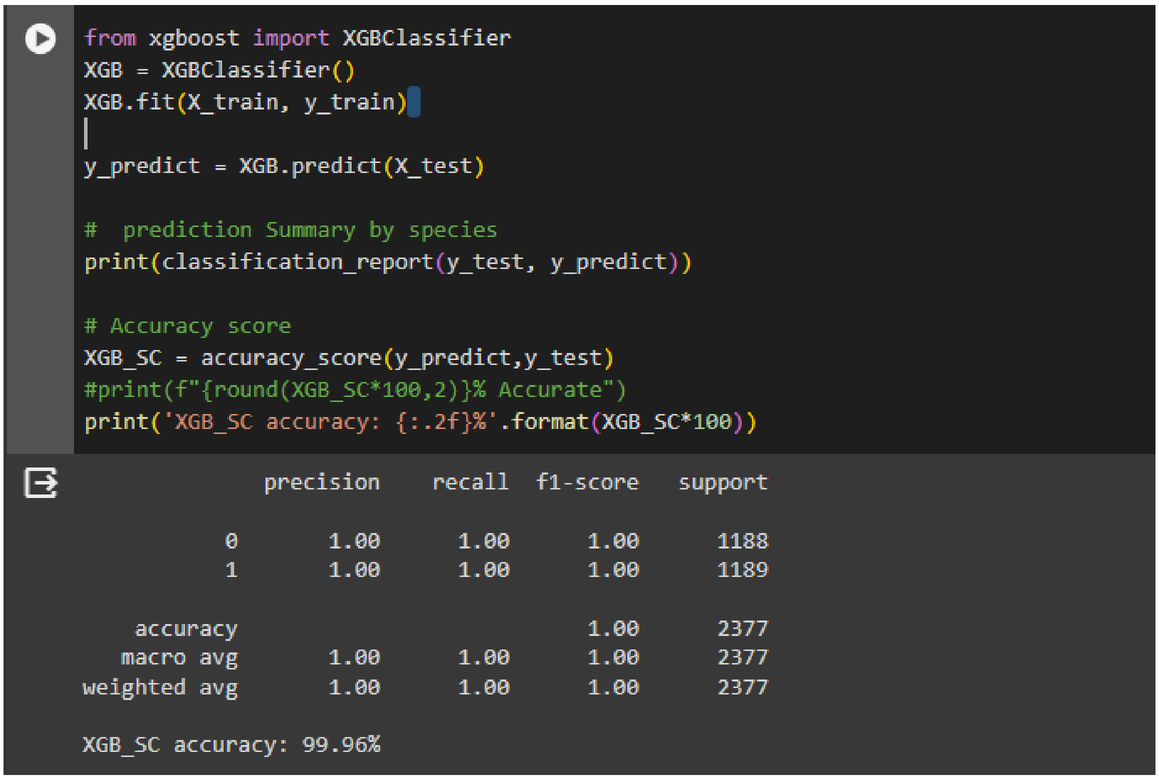
4. `y_predict = XGB.predict(X_test)`: This line uses the trained `XGBClassifier` model to make predictions on the test data `X_test`. The predicted labels are assigned to the variable `y_predict`.

5. `print(classification_report(y_test, y_predict))`: This line prints a summary of the prediction results by species. The `classification_report` function compares the predicted labels `y_predict` with the true labels `y_test` and provides metrics such as precision, recall, F1-score, and support for each class.

6. `XGB_SC = accuracy_score(y_predict,y_test)`: This line calculates the accuracy score of the predictions made by the `XGBClassifier` model. The `accuracy_score` function compares the predicted labels `y_predict` with the true

labels `y_test` and returns the proportion of correctly predicted labels.

7. `print('XGB_SC accuracy: {:.2f}%'.format(XGB_SC*100))`: This line prints the accuracy score as a percentage, formatted to two decimal places. The accuracy score is multiplied by 100 and displayed as a percentage to represent the model's performance in terms of the proportion of correctly classified instances in the test set.



```
from xgboost import XGBClassifier
XGB = XGBClassifier()
XGB.fit(X_train, y_train)

y_predict = XGB.predict(X_test)

#  prediction Summary by species
print(classification_report(y_test, y_predict))

# Accuracy score
XGB_SC = accuracy_score(y_predict,y_test)
#print(f"{round(XGB_SC*100,2)}% Accurate")
print('XGB_SC accuracy: {:.2f}%'.format(XGB_SC*100))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1188
           1       1.00      1.00      1.00      1189

    accuracy                           1.00      2377
   macro avg       1.00      1.00      1.00      2377
weighted avg       1.00      1.00      1.00      2377

XGB_SC accuracy: 99.96%
```

8.8.2. Logistic Regression

using the Logistic Regression algorithm from the scikit-learn library to train a classification model and make predictions. Here's a breakdown of what each line of code does:

1. `LRclassifier = LogisticRegression(solver='saga', max_iter=500, random_state=1)`: This line creates an instance of the LogisticRegression class with the specified parameters. The 'solver' parameter is set to 'saga', which is an optimization algorithm for solving logistic regression problems. The 'max_iter' parameter is set to 500, which determines the maximum number of iterations for the solver to converge. The 'random_state' parameter is set to 1, which initializes a random number generator for reproducibility.

2. `LRclassifier.fit(X_train, y_train)`: This line trains the Logistic Regression model using the training data `X_train` and the corresponding labels `y_train`. The model learns to map the input features in `X_train` to the target labels in `y_train`.

3. `y_pred = LRclassifier.predict(X_test)`: This line uses the trained Logistic Regression model to make predictions on the test data `X_test`. The predicted labels are assigned to the variable `y_pred`.

4. `print(classification_report(y_test, y_pred))`: This line prints a summary of the prediction results. The `classification_report` function compares the predicted labels `y_pred` with the true labels `y_test` and provides

metrics such as precision, recall, F1-score, and support for each class.

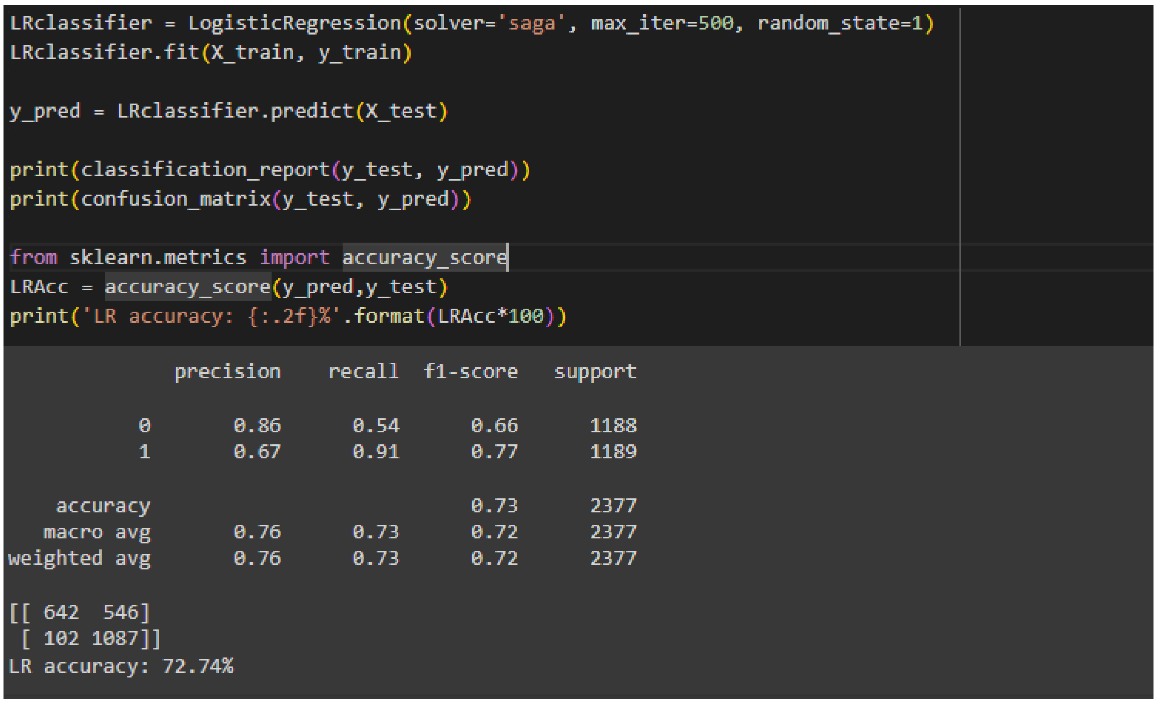
5. `print(confusion_matrix(y_test, y_pred))`: This line prints the confusion matrix, which is a table that shows the number of true positives, false positives, true negatives, and false negatives for each class. The `confusion_matrix` function compares the predicted labels `y_pred` with the true labels `y_test`.

6. `from sklearn.metrics import accuracy_score`: This line imports the `accuracy_score` function from the `sklearn.metrics` module. The `accuracy_score` function is used to calculate the accuracy of the predicted labels.

7. `LRAcc = accuracy_score(y_pred,y_test)`: This line calculates the accuracy score of the predictions made by the Logistic Regression model. The `accuracy_score` function compares the predicted labels `y_pred` with the true labels `y_test` and returns the proportion of correctly predicted labels.

8. `print('LR accuracy: {:.2f}%'.format(LRAcc*100))`: This line prints the accuracy score as a percentage, formatted to two decimal places. The accuracy score is multiplied by 100 and displayed as a percentage to represent the model's

performance in terms of the proportion of correctly classified instances in the test set.



```
LRclassifier = LogisticRegression(solver='saga', max_iter=500, random_state=1)
LRclassifier.fit(X_train, y_train)

y_pred = LRclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
LRAcc = accuracy_score(y_pred,y_test)
print('LR accuracy: {:.2f}%'.format(LRAcc*100))
```

```
              precision    recall  f1-score   support

           0       0.86      0.54      0.66      1188
           1       0.67      0.91      0.77      1189

    accuracy                           0.73      2377
   macro avg       0.76      0.73      0.72      2377
weighted avg       0.76      0.73      0.72      2377

[[ 642  546]
 [ 102 1087]]
LR accuracy: 72.74%
```

8.8.3. K-Nearest Neighbour (KNN)

using the k-nearest neighbors (KNN) algorithm to classify data and determine the best value for the number of neighbors (k) to use. Here's a breakdown of the code:

1. `scoreListknn = []`: This creates an empty list called `scoreListknn` to store the accuracy scores obtained from the KNN classifier.

2. `for i in range(1, 21)`: This loop iterates through the values from 1 to 20 (excluding 21). These values will be used as the number of neighbors (k) for the KNN classifier.

3. `KNclassifier = KNeighborsClassifier(n_neighbors=i)`: This line creates an instance of the KNeighborsClassifier class from the scikit-learn library and sets the number of neighbors (k) to the current value of `i`.

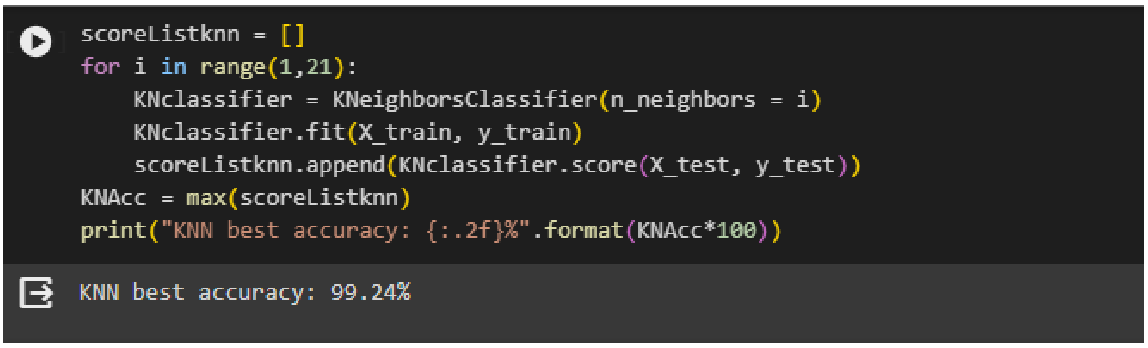
4. `KNclassifier.fit(X_train, y_train)`: This line trains the KNN classifier on the training data `X_train` (input features) and `y_train` (target labels).

5. `scoreListknn.append(KNclassifier.score(X_test, y_test))`: This line calculates the accuracy of the trained KNN classifier on the test data `X_test` and `y_test` and appends the accuracy score to the `scoreListknn` list.

6. `KNAcc = max(scoreListknn)`: This line finds the maximum accuracy score from the `scoreListknn` list and assigns it to the variable `KNAcc`.

7. `print("KNN best accuracy: {:.2f}%".format(KNAcc*100))`: This line prints the best accuracy score achieved by the KNN classifier as a percentage, with two decimal places.

In summary, the code performs a loop to train KNN classifiers with different values of k, stores the accuracy scores in a list, and then determines and prints the highest accuracy achieved among all the classifiers.



```
scoreListknn = []
for i in range(1,21):
    KNclassifier = KNeighborsClassifier(n_neighbors = i)
    KNclassifier.fit(X_train, y_train)
    scoreListknn.append(KNclassifier.score(X_test, y_test))
KNAcc = max(scoreListknn)
print("KNN best accuracy: {:.2f}%".format(KNAcc*100))
```

```
KNN best accuracy: 99.24%
```

8.8.4. Decision Tree

using the Decision Tree algorithm to classify data and determine the best value for the maximum number of leaf nodes in the tree. Here's a breakdown of the code:

1. `scoreListDT = []`: This creates an empty list called `scoreListDT` to store the accuracy scores obtained from the Decision Tree classifier.

2. `for i in range(2, 21)`: This loop iterates through the values from 2 to 20 (excluding 21). These values will be used as the maximum number of leaf nodes for the Decision Tree classifier.

3. `DTclassifier = DecisionTreeClassifier(max_leaf_nodes=i)`: This line creates an instance of the DecisionTreeClassifier class from the scikit-learn library and sets the maximum number of leaf nodes to the current value of `i`.

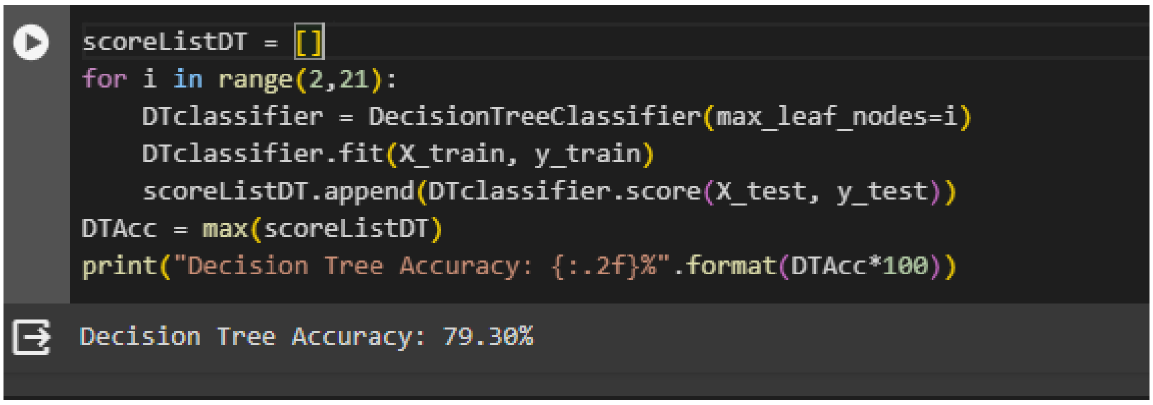
4. `DTclassifier.fit(X_train, y_train)`: This line trains the Decision Tree classifier on the training data `X_train` (input features) and `y_train` (target labels).

5. `scoreListDT.append(DTclassifier.score(X_test, y_test))`: This line calculates the accuracy of the trained Decision Tree classifier on the test data `X_test` and `y_test` and appends the accuracy score to the `scoreListDT` list.

6. `DTAcc = max(scoreListDT)`: This line finds the maximum accuracy score from the `scoreListDT` list and assigns it to the variable `DTAcc`.

7. `print("Decision Tree Accuracy: {:.2f}%".format(DTAcc*100))`: This line prints the best accuracy score achieved by the Decision Tree classifier as a percentage, with two decimal places.

In summary, the code performs a loop to train Decision Tree classifiers with different values for the maximum number of leaf nodes, stores the accuracy scores in a list, and then determines and prints the highest accuracy achieved among all the classifiers.



```
scoreListDT = []
for i in range(2,21):
    DTclassifier = DecisionTreeClassifier(max_leaf_nodes=i)
    DTclassifier.fit(X_train, y_train)
    scoreListDT.append(DTclassifier.score(X_test, y_test))
DTAcc = max(scoreListDT)
print("Decision Tree Accuracy: {:.2f}%".format(DTAcc*100))
```

```
Decision Tree Accuracy: 79.30%
```

8.8.5. Random Forest

using the Random Forest algorithm to classify data and determine the best value for the maximum number of leaf nodes in each decision tree within the Random Forest. Here's a breakdown of the code:

1. `scoreListRF = []`: This creates an empty list called `scoreListRF` to store the accuracy scores obtained from the Random Forest classifier.

2. `for i in range(2, 25)`: This loop iterates through the values from 2 to 24 (excluding 25). These values will be used as the maximum number of leaf nodes for each decision tree within the Random Forest.

3. `RFclassifier = RandomForestClassifier(n_estimators=1000, random_state=1, max_leaf_nodes=i)`: This line creates an instance of the RandomForestClassifier class from the scikit-learn library. It sets the number of estimators (decision trees) in the Random Forest to 1000, the random state to 1 for reproducibility, and the maximum number of leaf nodes to the current value of `i`.

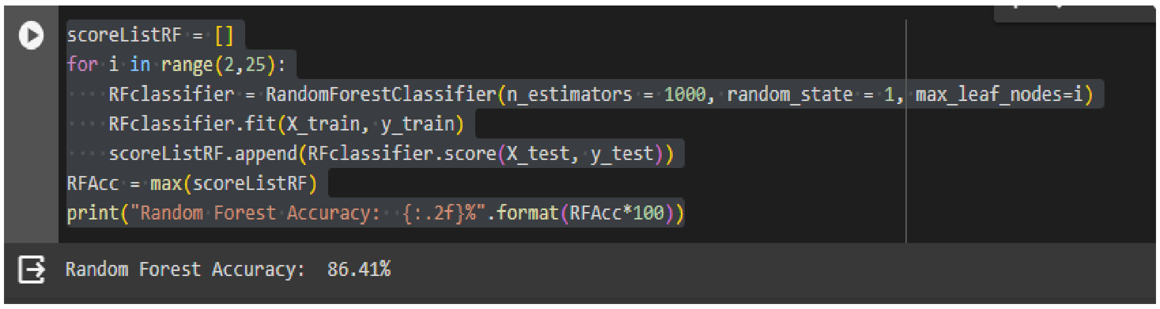
4. `RFclassifier.fit(X_train, y_train)`: This line trains the Random Forest classifier on the training data `X_train` (input features) and `y_train` (target labels).

5. `scoreListRF.append(RFclassifier.score(X_test, y_test))`: This line calculates the accuracy of the trained Random Forest classifier on the test data `X_test` and `y_test` and appends the accuracy score to the `scoreListRF` list.

6. `RFAcc = max(scoreListRF)`: This line finds the maximum accuracy score from the `scoreListRF` list and assigns it to the variable `RFAcc`.

7. `print("Random Forest Accuracy: {:.2f}%".format(RFAcc*100))`: This line prints the best accuracy score achieved by the Random Forest classifier as a percentage, with two decimal places.

In summary, the code performs a loop to train Random Forest classifiers with different values for the maximum number of leaf nodes in each decision tree, stores the accuracy scores in a list, and then determines and prints the highest accuracy achieved among all the classifiers.



```python
scoreListRF = []
for i in range(2,25):
    RFclassifier = RandomForestClassifier(n_estimators = 1000, random_state = 1, max_leaf_nodes=i)
    RFclassifier.fit(X_train, y_train)
    scoreListRF.append(RFclassifier.score(X_test, y_test))
RFAcc = max(scoreListRF)
print("Random Forest Accuracy:  {:.2f}%".format(RFAcc*100))
```

```
Random Forest Accuracy:  86.41%
```

8.8.6. Neural Network

using the MLPClassifier class from scikit-learn. Here's a breakdown of the code:

1. `from sklearn.neural_network import MLPClassifier`: This line imports the MLPClassifier class from the `sklearn.neural_network` module, which provides functionality for training and using a multi-layer perceptron (neural network) classifier.

2. `NNclassifier = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=1)`: This line creates an instance of the MLPClassifier class. It specifies the structure of the neural network with a single hidden layer containing

100 neurons. The `max_iter` parameter sets the maximum number of iterations for training the neural network, and `random_state` ensures reproducibility of results.

3. `NNclassifier.fit(X_train, y_train)`: This line trains the neural network classifier on the training data `X_train` (input features) and `y_train` (target labels).

4. `y_pred_nn = NNclassifier.predict(X_test)`: This line uses the trained neural network classifier to make predictions on the test data `X_test` and assigns the predicted labels to the `y_pred_nn` variable.

5. `print("Neural Network Classification Report:")`: This line prints a header indicating that the classification report will be displayed.

6. `print(classification_report(y_test, y_pred_nn))`: This line prints the classification report, which includes metrics such as precision, recall, F1-score, and support for each class. The `y_test` and `y_pred_nn` arrays are used to compute these metrics.

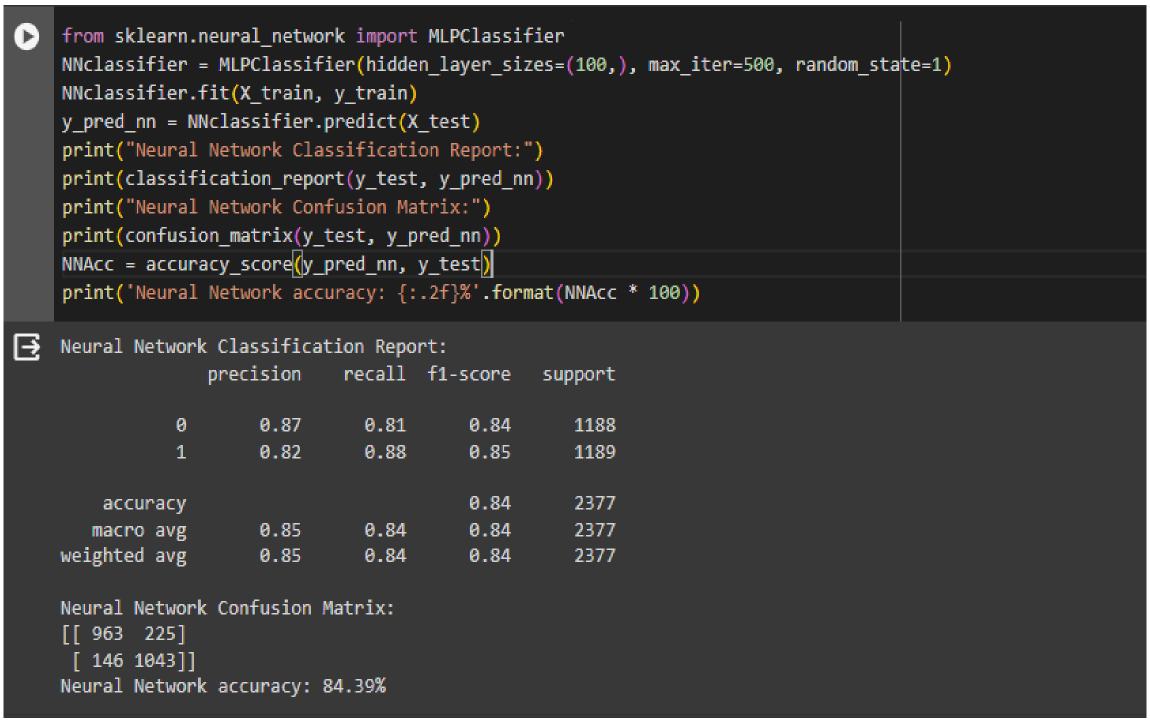
7. `print("Neural Network Confusion Matrix:")`: This line prints a header indicating that the confusion matrix will be displayed.

8. `print(confusion_matrix(y_test, y_pred_nn))`: This line prints the confusion matrix, which provides information about the number of true positives, true negatives, false positives, and false negatives. The `y_test` and `y_pred_nn` arrays are used to compute the confusion matrix.

9. `NNAcc = accuracy_score(y_pred_nn, y_test)`: This line calculates the accuracy of the neural network classifier by comparing the predicted labels `y_pred_nn` with the true labels `y_test` using the `accuracy_score` function.

10. `print('Neural Network accuracy: {:.2f}%'.format(NNAcc * 100))`: This line prints the accuracy of the neural network classifier as a percentage, with two decimal places.

In summary, the code trains a neural network classifier, makes predictions on the test data, and then evaluates the performance of the classifier by printing a classification report, confusion matrix, and accuracy score.



```
from sklearn.neural_network import MLPClassifier
NNclassifier = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=1)
NNclassifier.fit(X_train, y_train)
y_pred_nn = NNclassifier.predict(X_test)
print("Neural Network Classification Report:")
print(classification_report(y_test, y_pred_nn))
print("Neural Network Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_nn))
NNAcc = accuracy_score(y_pred_nn, y_test)
print('Neural Network accuracy: {:.2f}%'.format(NNAcc * 100))
```

```
Neural Network Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.81      0.84      1188
           1       0.82      0.88      0.85      1189

    accuracy                           0.84      2377
   macro avg       0.85      0.84      0.84      2377
weighted avg       0.85      0.84      0.84      2377

Neural Network Confusion Matrix:
[[ 963  225]
 [ 146 1043]]
Neural Network accuracy: 84.39%
```

we using neural network algorithm ,why?

We use artificial neural networks because they learn very efficiently and adaptively. They have the capability to learn "how" to solve a specific problem from the training data it receives. After learning, it can be used to solve that specific problem very quickly and efficiently with high accuracy. And as can be seen, the distribution of database are not balanced.

8.8.7. Naive Bayes

1. `from sklearn.naive_bayes import GaussianNB`: This line imports class `GaussianNB` from module `sklearn.naive_bayes`. `GaussianNB` is a Gaussian Naive Bayes classifier, which is a type of Naive Bayes algorithm used in classification tasks.

2. `NBclassifier = GaussianNB()`: This line creates an instance of the class `GaussianNB` and assigns it to the variable `NBclassifier`. The `GaussianNB` class is a machine learning model that uses the Gaussian Naive Bayes algorithm to make predictions.

3. `NBclassifier.fit(X_train, y_train)`: This line trains the `NBclassifier` model on the `X_train` and `y_train` training data. The `fit()` method is used to fit a Gaussian Naive Bayes model to the input data.

4. `y_pred_nb = NBclassifier.predict(X_test)`: This line uses the trained `NBclassifier` model to make predictions on the `X_test` test data. The ``predict()`` method is used to generate predictions for the input data.

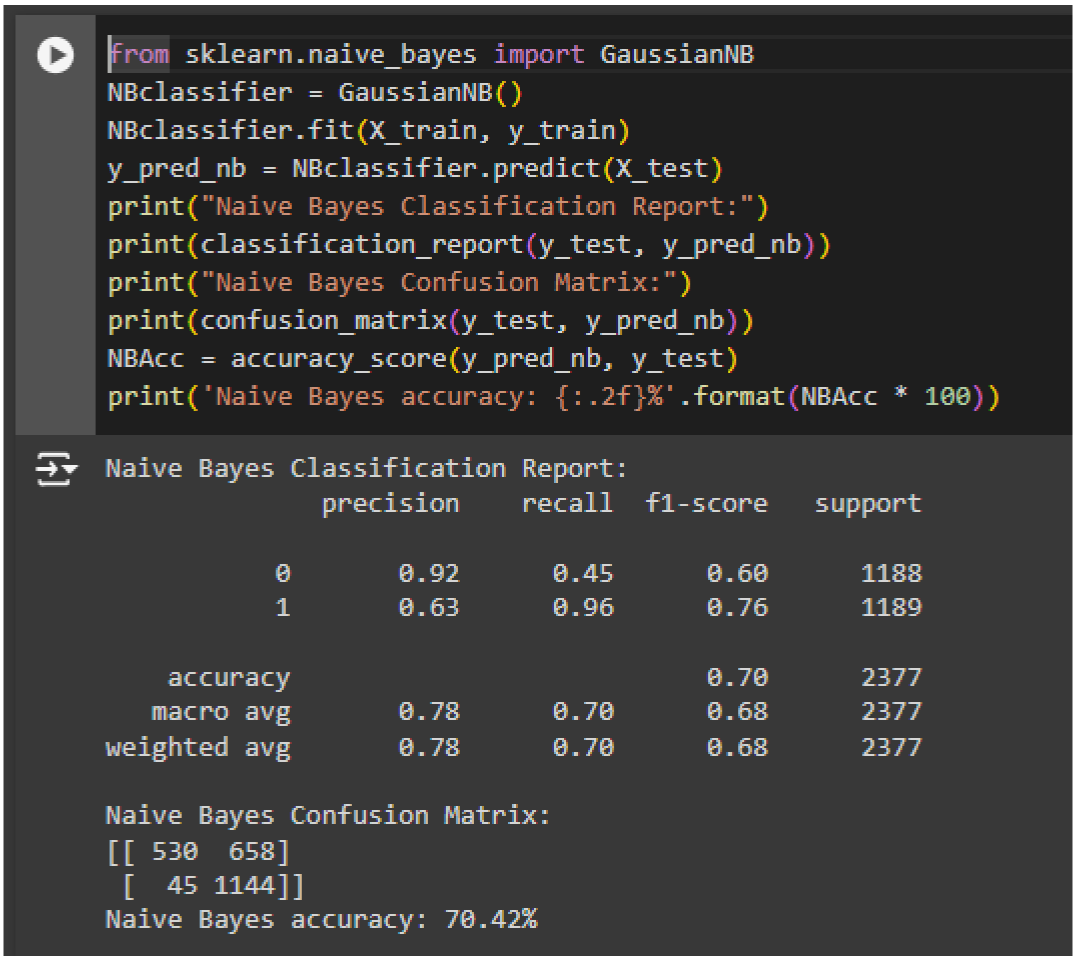
5. `print("Naive Bayes Classification Report:")`, `print(classification_report(y_test, y_pred_nb)`): These lines print the classification report for the Naive Bayes model. The rating report provides different metrics, such as precision, recall, F1 score, and precision, to evaluate the model's performance.

6. `print("Naive Bayes confusion matrix:")`, `print(confusion_matrix(y_test, y_pred_nb)`): These lines print the confusion matrix of the Naive Bayes model. A

confusion matrix is a table that depicts the performance of a classification model, showing the number of true positives, false positives, true negatives, and false negatives.

7. `NBAcc = accuracy_score(y_pred_nb, y_test)`, `print('Naive Bayes accuracy: {:.2f}%'.format(NBAcc * 100))`: These lines calculate the accuracy score for the Naive Bayes model and print the accuracy percentage .

Briefly, this code demonstrates the use of a Gaussian Naive Bayes classifier from the scikit-learn (sklearn) library. It trains the model on the provided training data, makes predictions on the test data, and evaluates the model's performance using various metrics, such as classification report, confusion matrix, and accuracy score.



```
from sklearn.naive_bayes import GaussianNB
NBclassifier = GaussianNB()
NBclassifier.fit(X_train, y_train)
y_pred_nb = NBclassifier.predict(X_test)
print("Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
print("Naive Bayes Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_nb))
NBAcc = accuracy_score(y_pred_nb, y_test)
print('Naive Bayes accuracy: {:.2f}%'.format(NBAcc * 100))
```

```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.45      0.60      1188
           1       0.63      0.96      0.76      1189

    accuracy                           0.70      2377
   macro avg       0.78      0.70      0.68      2377
weighted avg       0.78      0.70      0.68      2377

Naive Bayes Confusion Matrix:
[[ 530  658]
 [  45 1144]]
Naive Bayes accuracy: 70.42%
```

8.8.8. Support Vactor Machine

1. `SVCclassifier = SVC(kernel='rbf', max_iter=500)`: This line creates an instance of the `SVC` (Support Vector Classifier) class from the `sklearn.svm` module. The `SVC` class is a type of support vector machine (SVM) used for classification tasks. The `kernel='rbf'` parameter specifies the kernel function to be used, which in this case is the radial basis function (RBF) kernel. The parameter `max_iter=500` specifies the maximum number of iterations of the optimization algorithm.

2. `SVCclassifier.fit(X_train, y_train)`: This line trains the `SVCclassifier` model on the training data `X_train` and `y_train`. The `fit()` method is used to fit the SVM model to the input data.

3. `y_pred = SVCclassifier.predict(X_test)`: This line uses the trained `SVCclassifier` model to make predictions on the `X_test` test data. The `predict()` method is used to generate predictions for the input data.

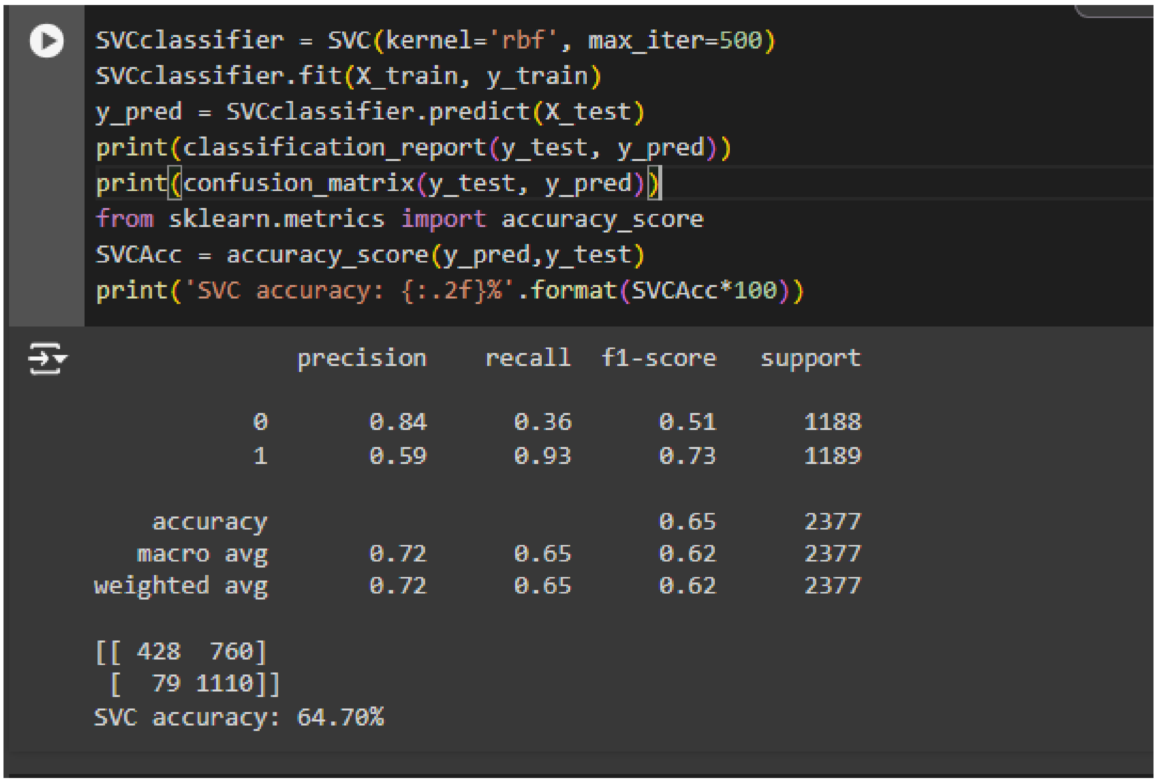
4. `print(classification_report(y_test, y_pred))`: This line prints the classification report for the SVC model. The rating report provides different metrics, such as precision, recall, F1 score, and precision, to evaluate the model's performance.

5. ``print(confusion_matrix(y_test, y_pred))`: This line prints the confusion matrix of the SVC model. A confusion matrix is a table that depicts the performance of a classification model, showing the number of true positives, false positives, true negatives, and false negatives.

6. `SVCAcc = Accuracy_score(y_pred,y_test)`: This line calculates the accuracy score for the SVC model by comparing the predicted labels (`y_pred`) with the true labels (`y_test`).

7. `print('SVC precision: {:.2f}%'.format(SVCAcc*100))`: This line prints the precision percentage of the SVC model.

Briefly, this code demonstrates the use of the Support Vector Classifier (SVC) from the scikit-learn (sklearn) library. It creates an SVC model, trains it on the provided training data, makes predictions on the test data, and evaluates the model's performance using a classification report, confusion matrix, and accuracy score.



```
SVCclassifier = SVC(kernel='rbf', max_iter=500)
SVCclassifier.fit(X_train, y_train)
y_pred = SVCclassifier.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
SVCAcc = accuracy_score(y_pred,y_test)
print('SVC accuracy: {:.2f}%'.format(SVCAcc*100))
```

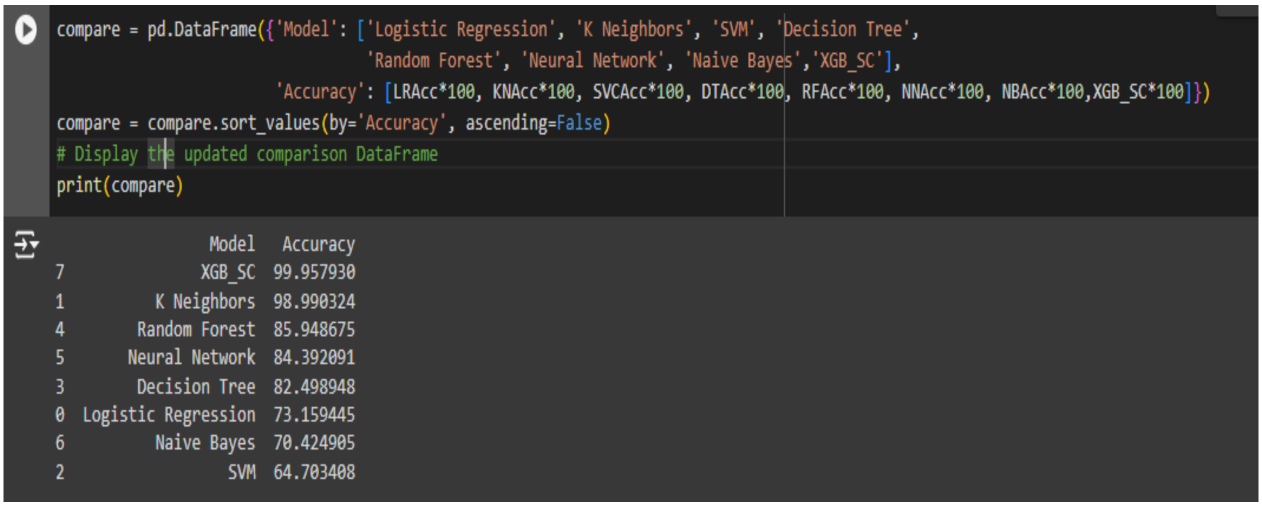
```
              precision    recall  f1-score   support

           0       0.84      0.36      0.51      1188
           1       0.59      0.93      0.73      1189

    accuracy                           0.65      2377
   macro avg       0.72      0.65      0.62      2377
weighted avg       0.72      0.65      0.62      2377

[[ 428  760]
 [  79 1110]]
SVC accuracy: 64.70%
```

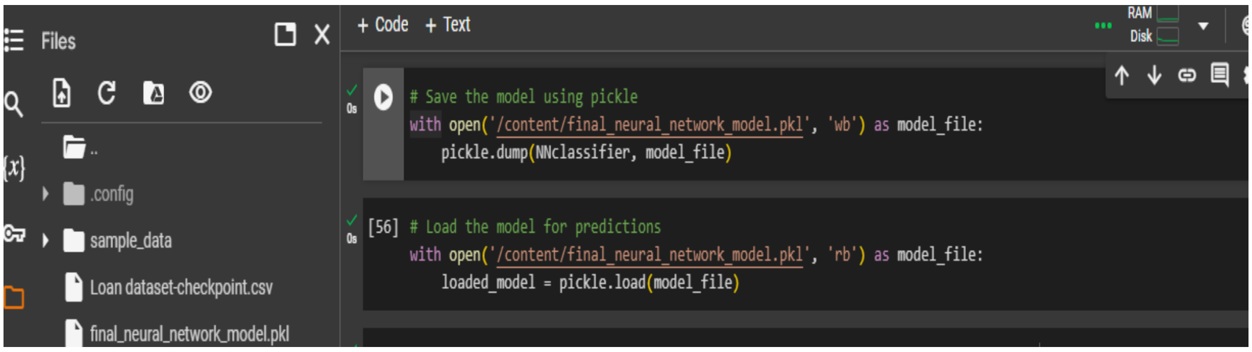
Comparison of the algorithms used


```
compare = pd.DataFrame({'Model': ['Logistic Regression', 'K Neighbors', 'SVM', 'Decision Tree',
                                  'Random Forest', 'Neural Network', 'Naive Bayes','XGB_SC'],
                        'Accuracy': [LRAcc*100, KNAcc*100, SVCAcc*100, DTAcc*100, RFAcc*100, NNAcc*100, NBAcc*100,XGB_SC*100]})
compare = compare.sort_values(by='Accuracy', ascending=False)
# Display the updated comparison DataFrame
print(compare)
```

```
                 Model   Accuracy
7               XGB_SC  99.957930
1          K Neighbors  98.990324
4        Random Forest  85.948675
5       Neural Network  84.392091
3        Decision Tree  82.498948
0  Logistic Regression  73.159445
6          Naive Bayes  70.424905
2                  SVM  64.703408
```

Save the model using pickle

Load the model for predictions



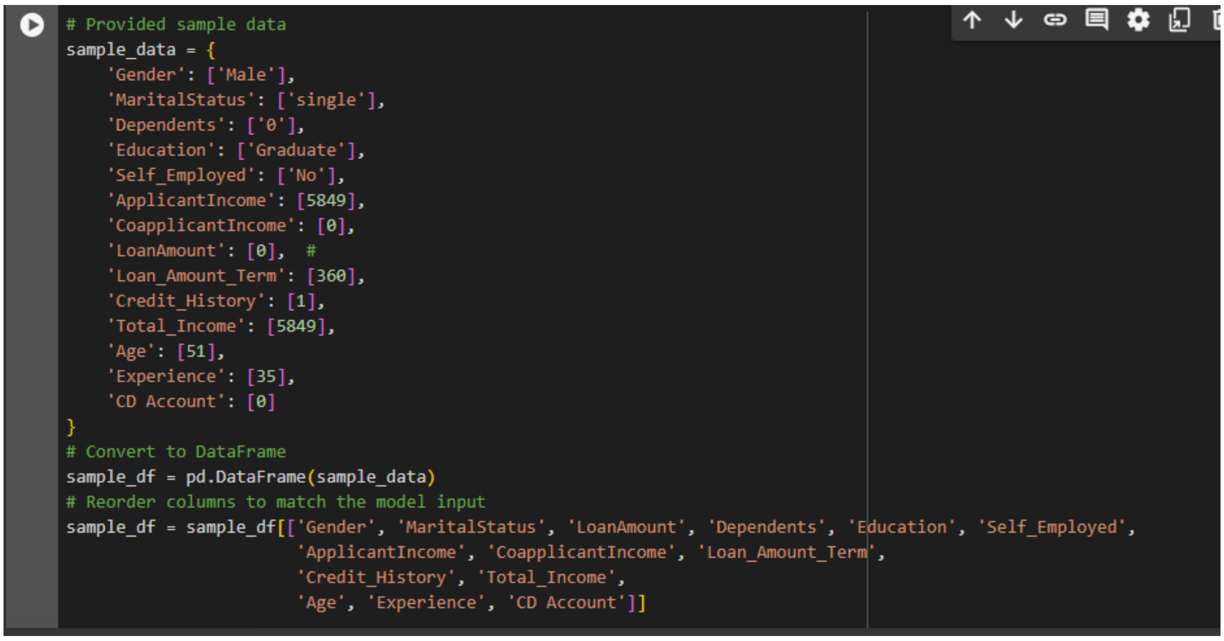
```
# Save the model using pickle
with open('/content/final_neural_network_model.pkl', 'wb') as model_file:
    pickle.dump(NNclassifier, model_file)
```

```
[56] # Load the model for predictions
with open('/content/final_neural_network_model.pkl', 'rb') as model_file:
    loaded_model = pickle.load(model_file)
```

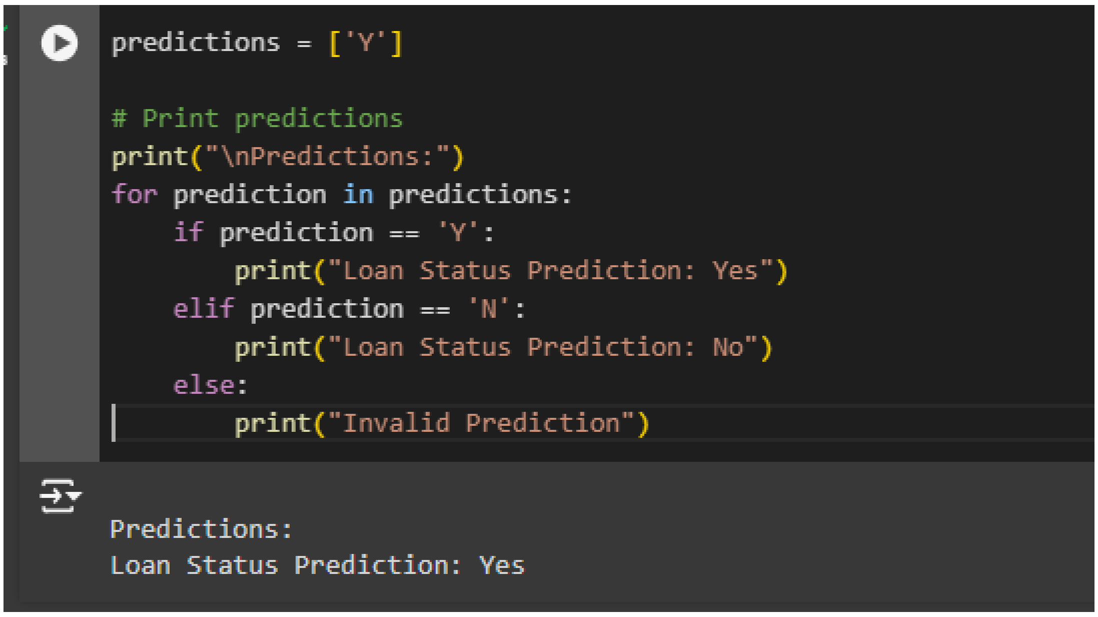
Sample Data

There are 14 input;

Gender,MaritalStatus,Dependents,Education,Self_Employed,ApplicantIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term,Credit_History,Total_Income,Age,Experience ,CD Account



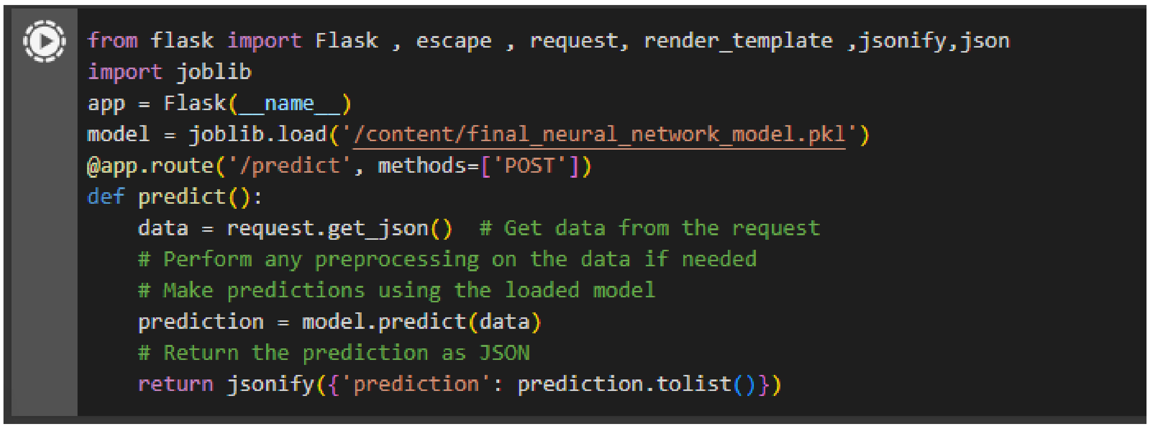
```python
# Provided sample data
sample_data = {
    'Gender': ['Male'],
    'MaritalStatus': ['single'],
    'Dependents': ['0'],
    'Education': ['Graduate'],
    'Self_Employed': ['No'],
    'ApplicantIncome': [5849],
    'CoapplicantIncome': [0],
    'LoanAmount': [0],  #
    'Loan_Amount_Term': [360],
    'Credit_History': [1],
    'Total_Income': [5849],
    'Age': [51],
    'Experience': [35],
    'CD Account': [0]
}
# Convert to DataFrame
sample_df = pd.DataFrame(sample_data)
# Reorder columns to match the model input
sample_df = sample_df[['Gender', 'MaritalStatus', 'LoanAmount', 'Dependents', 'Education', 'Self_Employed',
                       'ApplicantIncome', 'CoapplicantIncome', 'Loan_Amount_Term',
                       'Credit_History', 'Total_Income',
                       'Age', 'Experience', 'CD Account']]
```



```python
predictions = ['Y']

# Print predictions
print("\nPredictions:")
for prediction in predictions:
    if prediction == 'Y':
        print("Loan Status Prediction: Yes")
    elif prediction == 'N':
        print("Loan Status Prediction: No")
    else:
        print("Invalid Prediction")
```

```
Predictions:
Loan Status Prediction: Yes
```



```python
from flask import Flask , escape , request, render_template ,jsonify,json
import joblib
app = Flask(__name__)
model = joblib.load('/content/final_neural_network_model.pkl')
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()  # Get data from the request
    # Perform any preprocessing on the data if needed
    # Make predictions using the loaded model
    prediction = model.predict(data)
    # Return the prediction as JSON
    return jsonify({'prediction': prediction.tolist()})
```

Conclusions

9. Conclusions

The quick Loan website provides many features, including:

1. Ease of use: The customer loan program is easy to use, as it provides a simple and simplified user interface for customers to easily submit loan applications. Customers can browse information and fill out the required forms easily and conveniently.
2. Saving time: It reduces the need to go personally to the branch, as customers can request loans and find out the terms and conditions via the website.
3. Transparency of information: The bank’s website provides accurate information about available loans, their conditions, and interest rates.
4. Secure operations: The bank provides a secure system for submitting applications online and processing personal data with complete confidentiality.
5. Contact us: Contact information to contact the customer service team for additional assistance or inquiries.
6. Application process:
   - Clear steps for the application process online or in the branch.
   - An explanation of the approval, issuance, and payment scheduling process.

Future Work

10. Future work

10.1. Application development steps

Developing a website for bank loan services requires several steps:

- First step: Developing the user interface in a more elaborate and attractive manner, focusing on user experience and ease of use.
- Second step: Develop a database that stores customer information and requests in a safe and organized manner.
- Third step: publishing the website on the Internet and ensuring its availability and access to the target audience.
- Fourth step: We market the site to increase awareness of it and increase the number of visitors and potential customers.
- Fifth step: We monitor and maintain the site regularly to ensure its continuity and improvement over time.

10.2. Completed work

The application was completed as we initially planned:
We have completed the design of the entire application and it contains:

Five interactive features, all of which we use to save time:

A. We chose the logo at the beginning so that it shows the form of the program
B. On the main page, it contains all the basic operations that the program performs, which consists of five features
C. Here we highlight each feature and the work it does in order to clarify what is completed in the application:
    a. Converting from image to text through which text can be extracted and operations performed on it, such as translating into more than one language and listening to the text
    b. Solve arithmetic equations such as addition, subtraction, division, and multiplication
    c. Extract the link of the QR Code without the Internet
    d. Extracting the name of the traffic lights signs after taking pictures at any angle
    e. Extract the document and can also be heard
D. The conversion of the entire application to Arabic or English has been activated
E. Use camera and gallery enabled

References

11. References

- https://nextjs.org/
- https://github.com/
- https://www.freepik.com/
- https://icons8.com/icons
- https://www.photoroom.com/tools/background-remover
- https://legacy.reactjs.org/
- http://localhost/phpmyadmin
- https://www.w3schools.com/js/default.asp
- https://www.w3schools.com/php/default.asp
- https://www.w3schools.com/css/default.asp
- https://www.w3schools.com/react/default.asp
- https://www.w3schools.com/mysql/default.asp
- https://keras.io/api/models/model_training_apis/?fbclid=IwAR2_FwRWo93
- https://www.kaggle.com/code/caesarmario/loan-prediction-w-various-ml-models

- https://www.w3schools.com/python/default.asp
- https://www.w3schools.com/python/python_file_write.asp
- https://www.w3schools.com/python/numpy/default.asp
- https://www.w3schools.com/python/pandas/default.asp
- https://www.w3schools.com/python/pandas/pandas_csv.asp
- https://www.w3schools.com/python/pandas/pandas_cleaning_empty_cells.asp
- https://www.w3schools.com/datascience/ds_python.asp
- https://www.w3schools.com/python/python_ml_getting_started.asp
- https://www.w3schools.com/python/python_ml_linear_regression.asp
- https://www.w3schools.com/python/python_ml_train_test.asp

- https://www.w3schools.com/python/python_ml_decision_tree.asp
- https://www.w3schools.com/python/python_ml_logistic_regression.asp
- https://www.w3schools.com/python/python_ml_knn.asp