Application Building

Team ID	PNT2022TMID13268
Project Name	Smart Lender Applicant Credibility Prediction for Loan Approval

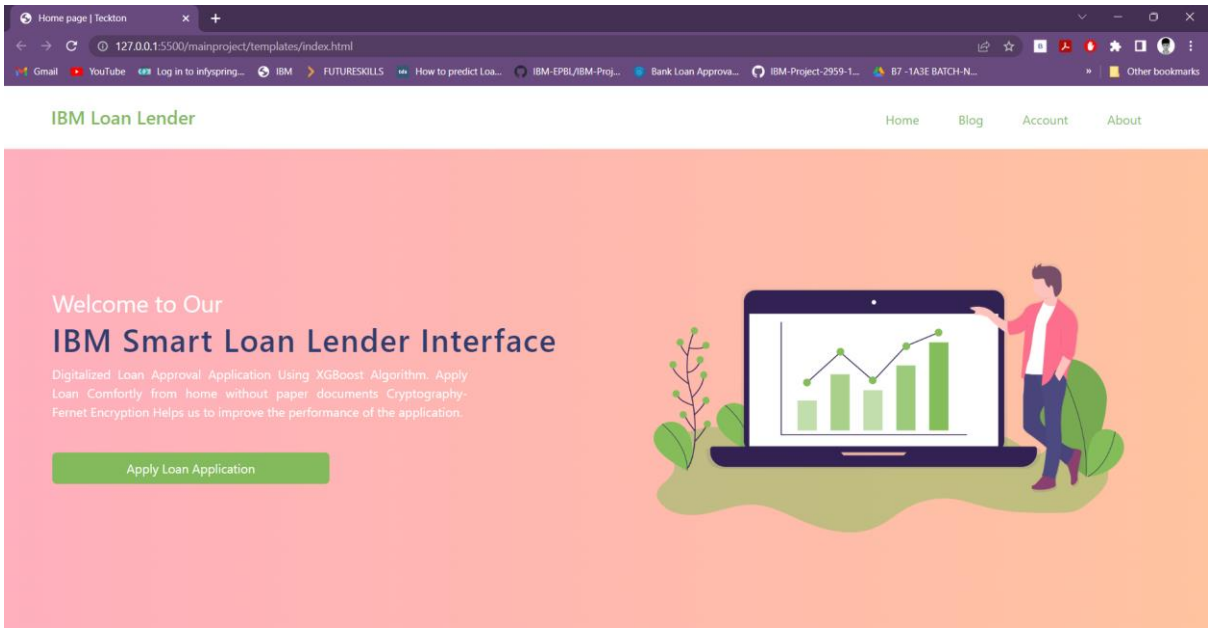
Application Building:

This section has the following tasks

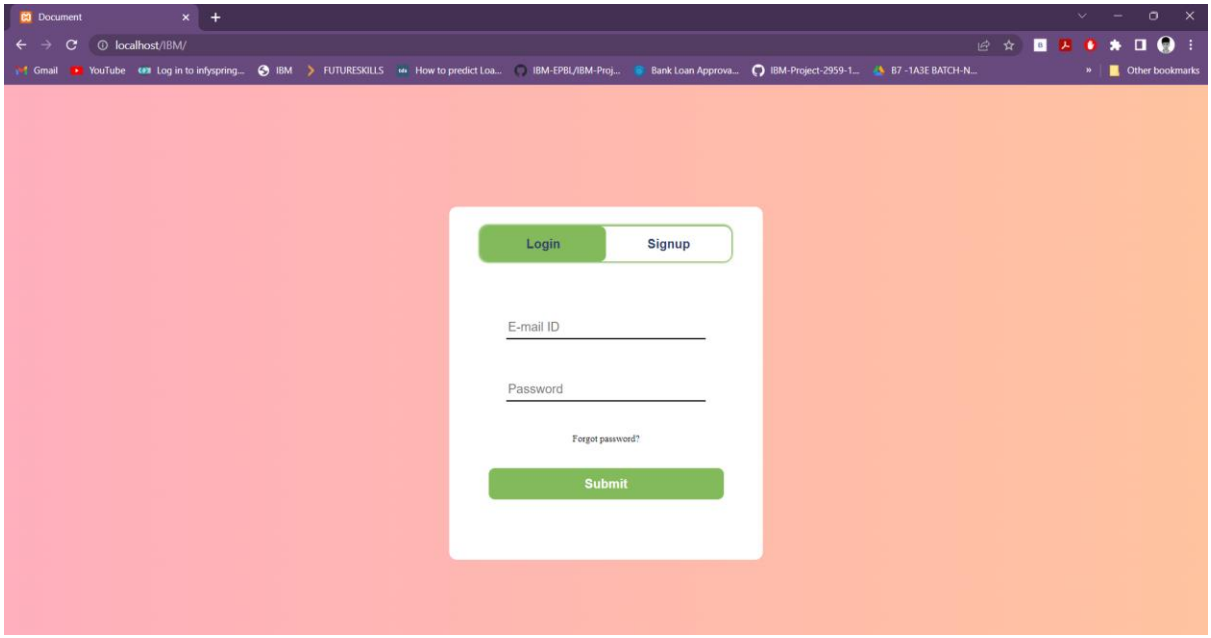
- Building HTML Pages
- Building serverside script

Building HTML Pages

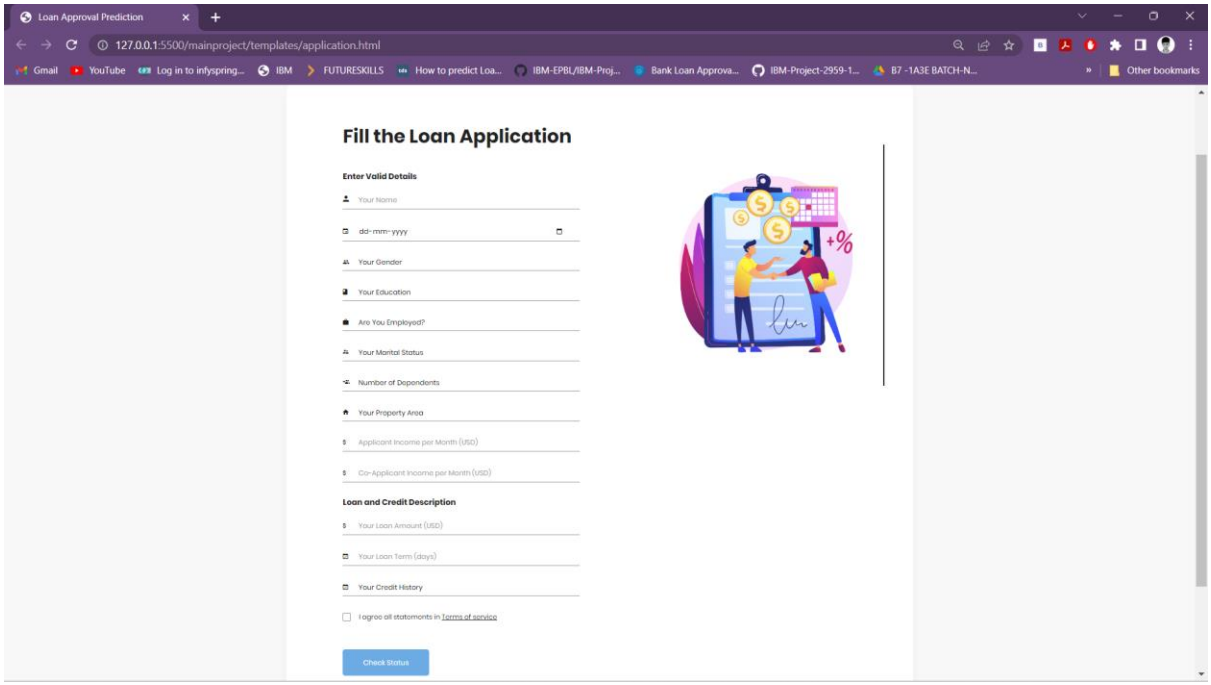
index.html



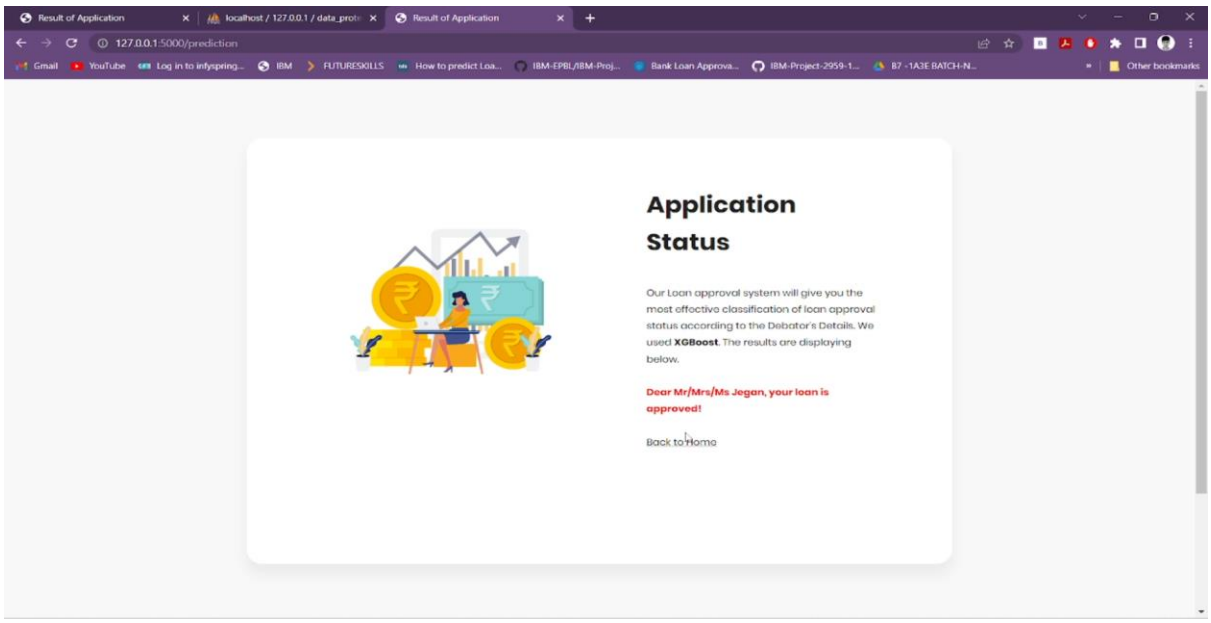
Login/Register Page:



application.html



prediction.html



Build Python Code

sever1.js Machine Learning Model Dev.ipynb mainserver.py

mainproject > mainserver.py > ...

```python
from flask import Flask,redirect,url_for,request,jsonify, render_template
# Data manipulation
import pandas as pd
# Matrices manipulation
import numpy as np
# Script logging
import logging
# ML model
import joblib
# JSON manipulation
import json
# Utilities
import sys
import os

import json
import requests
from cryptography.fernet import Fernet
import mysql.connector
def sample(e,r1):
    a=["http://localhost:8084/","http://localhost:8081/","http://localhost:8082/","http://localhost:8083/"]
    def datainsert(a):
        mydb = mysql.connector.connect(
          host="localhost",
          user="root",
          password="",
          database="data_protection"
        )
        mycursor = mydb.cursor()

        sql = "INSERT INTO data (emailid,password1,password2,password3,password4,key1,key2,key3,key4) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        val = (a,"","","","","","","","")
        mycursor.execute(sql, val)

        mydb.commit()
    def sep(s):

        c=int(len(s)/4)
        k=c
        data=[]
        i=0
        for j in range(4):
            if (j==3):
                data.append(s[i:])
            else:
                data.append(s[i:c])
                i=i+k
                c=c+k
        return data
```

mainproject > mainserver.py > ...

```python
        return data

    def encrypt(message):
        data=[]
        key = Fernet.generate_key()
        print(key)
        data.append(key)
        fernet = Fernet(key)
        encMessage = fernet.encrypt(message.encode())
        print(encMessage)
        data.append(encMessage)
        return data
    data=encrypt(r1)
    password=sep(data[1])
    key=sep(data[0])

    def postdata(a,b,c):
        response=requests.post(a,{
            "email":e,
            "password":b,
            "key":c
        })
    datainsert(e)
    for i in range(4):
        postdata(a[i],password[i],key[i])
def checkpass(a1,b):
    password=[]
    key=[]
    a=["http://localhost:8084/","http://localhost:8081/","http://localhost:8082/","http://localhost:8083/"]
    for i in a:
        x = requests.get(i,data=a1)
        r=x.text.split(" ")
        print(x.text)
        password.append(r[0])
        key.append(r[1])
    password="".join(password).encode()
    key="".join(key).encode()
    f=Fernet(key)
    rp=f.decrypt(password).decode()
    if (rp==b):
        return 1
    else:
        return 0
def checkemail(a):
    mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="",
```

mainproject > mainserver.py > ...

```python
    )

    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM data WHERE emailid='"+a+"'")

    myresult = mycursor.fetchall()
    if(len(myresult)!=0):
        return 1
    else:
        return 0
app = Flask(__name__)
a=[]
@app.route('/',methods=["POST","GET"])
def hello_world():
    if request.method=="POST":
        user=request.form['emails']
        user1=request.form['passwords']
        if(checkemail(user)==0):
            sample(user,user1)
            return render_template("index.html")
        elif(checkemail(user)==1):
            return redirect("http://localhost/IBM/index.php?signid=alr")
    else:
        return "success"
@app.route('/login',methods=["POST","GET"])
def hello_world1():
    if request.method=="POST":
        user=request.form['email']
        user1=request.form['password']
        if(checkemail(user)==0):
            return redirect("http://localhost/IBM/index.php?id=not")
        elif(checkpass(user,user1)==0):
             return redirect("http://localhost/IBM/index.php?pass=not")
        else:

            return render_template("index.html")
    else:
        return "success"

# Current directory
current_dir = os.path.dirname(__file__)

# Function
def ValuePredictor(data = pd.DataFrame):
    # Model name
    model_name = 'bin/xgboostModel.pkl'
    # Directory where the model is stored
```

mainproject > mainserver.py > ...

```python
    model_name = 'bin/xgboostModel.pkl'
    # Directory where the model is stored
    model_dir = os.path.join(current_dir, model_name)
    # Load the model
    loaded_model = joblib.load(open(model_dir, 'rb'))
    # Predict the data
    result = loaded_model.predict(data)
    return result[0]

# Application page
@app.route('/application')
def home():
    return render_template('application.html')

# Prediction page
@app.route('/prediction', methods = ['POST'])
def predict():
    if request.method == 'POST':
        # Get the data from form
        name = request.form['name']
        gender = request.form['gender']
        education = request.form['education']
        self_employed = request.form['self_employed']
        marital_status = request.form['marital_status']
        dependents = request.form['dependents']
        applicant_income = request.form['applicant_income']
        coapplicant_income = request.form['coapplicant_income']
        loan_amount = request.form['loan_amount']
        loan_term = request.form['loan_term']
        credit_history = request.form['credit_history']
        property_area = request.form['property_area']

        # Load template of JSON file containing columns name
        # Schema name
        schema_name = 'data/columns_set.json'
        # Directory where the schema is stored
        schema_dir = os.path.join(current_dir, schema_name)
        with open(schema_dir, 'r') as f:
            cols =  json.loads(f.read())
        schema_cols = cols['data_columns']

        # Parse the categorical columns
        # Column of dependents
        try:
            col = ('Dependents_' + str(dependents))
            if col in schema_cols.keys():
                schema_cols[col] = 1
            else:
```

mainproject > mainserver.py > ...

```python
            else:
                pass
        except:
            pass
        # Column of property area
        try:
            col = ('Property_Area_' + str(property_area))
            if col in schema_cols.keys():
                schema_cols[col] = 1
            else:
                pass
        except:
            pass

        # Parse the numerical columns
        schema_cols['ApplicantIncome'] = applicant_income
        schema_cols['CoapplicantIncome'] = coapplicant_income
        schema_cols['LoanAmount'] = loan_amount
        schema_cols['Loan_Amount_Term'] = loan_term
        schema_cols['Gender_Male'] = gender
        schema_cols['Married_Yes'] = marital_status
        schema_cols['Education_Not Graduate'] = education
        schema_cols['Self_Employed_Yes'] = self_employed
        schema_cols['Credit_History_1.0'] = credit_history

        # Convert the JSON into data frame
        df = pd.DataFrame(
                data = {k: [v] for k, v in schema_cols.items()},
                dtype = float
            )

        # Create a prediction
        print(df.dtypes)
        result = ValuePredictor(data = df)

        # Determine the output
        if int(result) == 1:
            prediction = 'Dear Mr/Mrs/Ms {name}, your loan is approved!'.format(name = name)
        else:
            prediction = 'Sorry Mr/Mrs/Ms {name}, your loan is rejected!'.format(name = name)

        # Return the prediction
        return render_template('prediction.html', prediction = prediction)

    # Something error
    else:
        # Return error
        return render_template('error.html', prediction = prediction)
```

```python
        # Determine the output
        if int(result) == 1:
            prediction = 'Dear Mr/Mrs/Ms {name}, your loan is approved!'.format(name = name)
        else:
            prediction = 'Sorry Mr/Mrs/Ms {name}, your loan is                    e)
                                                  (variable) prediction: str
                                                  (variable) prediction: str
        # Return the prediction
        return render_template('prediction.html', prediction = prediction)

    # Something error
    else:
        # Return error
        return render_template('error.html', prediction = prediction)


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

Run The Application:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER   COMMENTS                                   Python

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS V:\XAMPP\htdocs\IBM> & C:/Users/jegan/AppData/Local/Programs/Python/Python310/python.exe v:/XAMPP/htdocs/IBM/mainproject/mainserver.py
 * Serving Flask app 'mainserver'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 307-777-028

Ln 211, Col 52   Spaces: 4   UTF-8   LF   Python   3.10.8 64-bit   Go Live
```