Practical 3: Implement CRUD Operation in DJANGO

Subject: - Application Development Framework (2CSDE86)

Name: - Sanket Chheladiya

Roll no. : - 18BCE042

- **Templates/enroll**

Step 1: -

Design HTML page base.html in the template/enroll folder. That will show form and database content into browser.

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CRUD - Sanket Chheladiya</title>
    <link rel="stylesheet" href="{% static 'enroll/css/bootstrap.css' %}">
</head>
<body>

<div class="container mt-5">
    <h2 class="text-center alert alert-danger">CRUD Operations</h2>
    {% block content %}

    {% endblock content %}
</div>

<script src="{% static 'enroll/js/jquery.js' %}"></script>
<script src="{% static 'enroll/js/popper.js' %}"></script>
<script src="{% static 'enroll/js/bootstrap.js' %}"></script>
```

Step 2: -

Designing sub HTML form page that will show new student add from as well as display list of student who already in our database.

```
{% extends 'enroll/base.html' %}
{% block content %}
<div class="row">
    <div class="col-sm-4">
        <h4 class="text-center alert alert-info">Add New Student</h4>
        <form action="" method="POST">
            {% csrf_token %}
            {{form.as_p}}
            <input type="submit" class="btn btn-success" value="Add">
        </form>
    </div>

    <div class="col-sm-7 offset-1">
        <h4 class="text-center alert alert-info">Show Student Information</h4>
        {% if stu %}
        <table class="table table-hover">
            <thead>
                <tr>
                    <th scope="col">ID</th>
                    <th scope="col">First Name</th>
                    <th scope="col">Middle Name</th>
                    <th scope="col">Last Name</th>
```

```
                <th scope="col">Address</th>
                <th scope="col">Email</th>
                <th scope="col">Contact</th>
                <th scope="col">Alternate Email</th>
                <th scope="col">Alternate Contact</th>
                <th scope="col">Father Name</th>
                <th scope="col">Mother Name</th>
                <th scope="col">Institute Name</th>
                <th scope="col">Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for st in stu %}
            <tr>
                <th scope="row">{{st.id}}</th>
                <td>{{st.fname}}</td>
                <td>{{st.mname}}</td>
                <td>{{st.lname}}</td>
                <td>{{st.age}}</td>
                <td>{{st.gender}}</td>
                <td>{{st.address}}</td>
                <td>{{st.email}}</td>
```

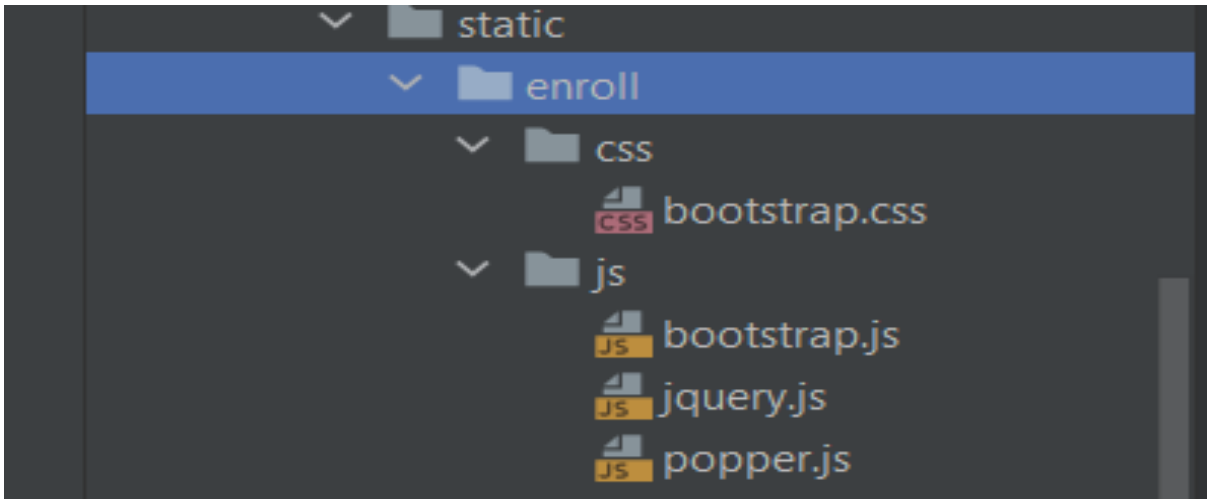
```
                <td>{{st.almobno}}</td>
                <td>{{st.fathername}}</td>
                <td>{{st.mothername}}</td>
                <td>{{st.insname}} </td>
                <td>
                    <a href="{% url 'updatedata' st.id %}" class="btn btn-warning btn-sm">Edit</a>
                    <form action="{% url 'deletedata' st.id  %}" method="post" class="d-inline">
                        {% csrf_token %}
                        <input type="submit" value="Delete" class="btn btn-danger btn-sm">
                    </form>
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    {% else %}
    <h4 class="text-center alert alert-warning">No Records</h4>
    {% endif %}
  </div>
</div>
{% endblock content %}
```

Step 3: -

Designing new sub HTML page that is updatestudent.html when we click update button that is placed into student display table, this sub HTML page will open and all the details related student will be display and whatever we want to change just update that form field with new value and click submit all the records override with existing records and data will be updated. If do not want to update click Back to Home that will redirect to homepage.

```
{% extends 'enroll/base.html' %}
{% block content %}
<div class="row">
    <div class="col-sm-8 offset-2">
        <h4 class="alert alert-info">Update Student Information</h4>
        <form action="" method="post">
            {% csrf_token %}
            {{form.as_p}}
            <input type="submit" class="btn btn-success" value="Update">
            <a href="{% url 'addandshow'  %}" class="btn btn-info">Back to Home</a>
        </form>
    </div>

</div>
{% endblock content %}
```

- **Static/enrol**
  -Contain required Designing tools including bootstrap, jQuery.



- **Migrations folder**

- **This folder is for creating new database for student to store that records.**
- from django.db import migrations, models
- **That has dependency section, operations like migrations.CreateModel that create database and structure for that is**

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]
```

```
        operations = [
            migrations.CreateModel(
                name='User',
                fields=[
                    ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                    ('fname', models.CharField(max_length=20)),
                    ('mname', models.CharField(max_length=20)),
                    ('lname', models.CharField(max_length=20)),
                    ('age', models.CharField(max_length=20)),
                    ('gender', models.CharField(max_length=20)),
                    ('address', models.CharField(max_length=200)),
                    ('email', models.EmailField(max_length=100)),
                    ('mobno', models.CharField(max_length=15)),
                    ('alemail', models.EmailField(max_length=100)),
                    ('almobno', models.CharField(max_length=15)),
                    ('fathername', models.CharField(max_length=100)),
                    ('mothername', models.CharField(max_length=100)),
                    ('insname', models.CharField(max_length=20)),
                ],
            ),
        ]
```

- **admin.py**

```
from django.contrib import admin
from .models import User
# Register your models here.
@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('id', 'fname', 'mname', 'lname', 'age', 'gender', 'address', 'email', 'mobno', 'alemail', 'almobno',
'fathername', 'mothername', 'insname')
```

- **Apps.py**

```
from django.apps import AppConfig


class EnrollConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'enroll'
```

- **Forms.py**

```
from django import forms
from .models import User
class StudentRegistration(forms.ModelForm):
    class Meta:
        model = User
        fields = ['fname', 'mname' , 'lname', 'age', 'gender', 'address', 'email', 'mobno', 'alemail', 'almobno',
'fathername', 'mothername', 'insname']
        widgets = {
            'fname': forms.TextInput(attrs={'class': 'form-control'}),
            'mname': forms.TextInput(attrs={'class': 'form-control'}),
            'lname': forms.TextInput(attrs={'class': 'form-control'}),
            'age': forms.TextInput(attrs={'class': 'form-control'}),
            'gender': forms.TextInput(attrs={'class': 'form-control'}),
            'address': forms.TextInput(attrs={'class': 'form-control'}),
            'email': forms.EmailInput(attrs={'class': 'form-control'}),
            'mobno': forms.TextInput(attrs={'class': 'form-control'}),
```

```
            'alemail': forms.EmailInput(attrs={'class': 'form-control'}),
            'almobno': forms.TextInput(attrs={'class': 'form-control'}),
            'fathername': forms.TextInput(attrs={'class': 'form-control'}),
            'mothername': forms.TextInput(attrs={'class': 'form-control'}),
            'insname': forms.TextInput(attrs={'class': 'form-control'}),

        }
```

- **Models.py**

```
from django.db import models

# Create your models here.
class User(models.Model):
    fname = models.CharField(max_length=20)
    mname = models.CharField(max_length=20)
    lname = models.CharField(max_length=20)
    age = models.CharField(max_length=20)
    gender = models.CharField(max_length=20)
    address = models.CharField(max_length=200)
    email = models.EmailField(max_length=100)
    mobno = models.CharField(max_length=15)
    alemail = models.EmailField(max_length=100)
    almobno = models.CharField(max_length=15)
    fathername = models.CharField(max_length=100)
    mothername = models.CharField(max_length=100)
    insname = models.CharField(max_length=20)
```

- **views.py**

```
cd from django.shortcuts import render, HttpResponseRedirect
from .forms import StudentRegistration
from .models import User
# Create your views here.
#This function will add new item and show all items
def add_show(request):
    if(request.method == 'POST'):
        fm = StudentRegistration(request.POST)
        if(fm.is_valid()):
            fm.save()
            fm = StudentRegistration()
    else:
        fm = StudentRegistration()
    stud = User.objects.all()
    return render(request, 'enroll/addandshow.html', {'form':fm, 'stu':stud})

# This function will update/Edit
def update_data(request, id):
    if request.method == 'POST':
        pi = User.objects.get(pk=id)
        fm = StudentRegistration(request.POST, instance=pi)
        if fm.is_valid():
```

```
            fm.save()
    else:
        pi = User.objects.get(pk=id)
        fm = StudentRegistration(instance=pi)
    return render(request, 'enroll/updatestudent.html', {'form':fm})


# This function will delete
def delete_data(request,id):
    if request.method == 'POST':
        pi = User.objects.get(pk=id)
        pi.delete()
        return HttpResponseRedirect('/')
```

- **settings.py**

```
"""
Django settings for CRUD project.

Generated by 'django-admin startproject' using Django 3.2.6.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-$m@wuzy_82yp250&qrph&^5xg_!y)02k&h3zb2&u9yz-6t3b#b'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'enroll',
```

```
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'CRUD.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'CRUD.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
```

```
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

- urls.py

```
from django.contrib import admin
from django.urls import path
from enroll import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.add_show, name = "addandshow" ),
    path('delete/<int:id>/', views.delete_data, name="deletedata"),
    path('<int:id>/', views.update_data, name="updatedata"),
]
```

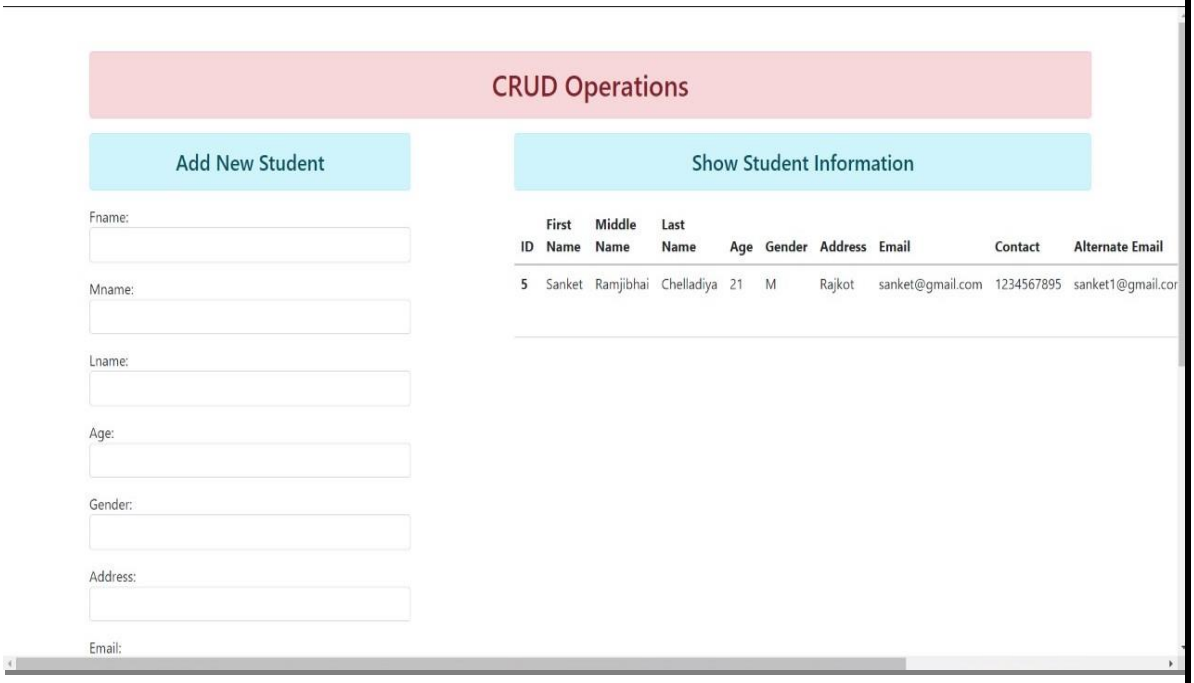
- wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'CRUD.settings')

application = get_wsgi_application()
```

Output: -

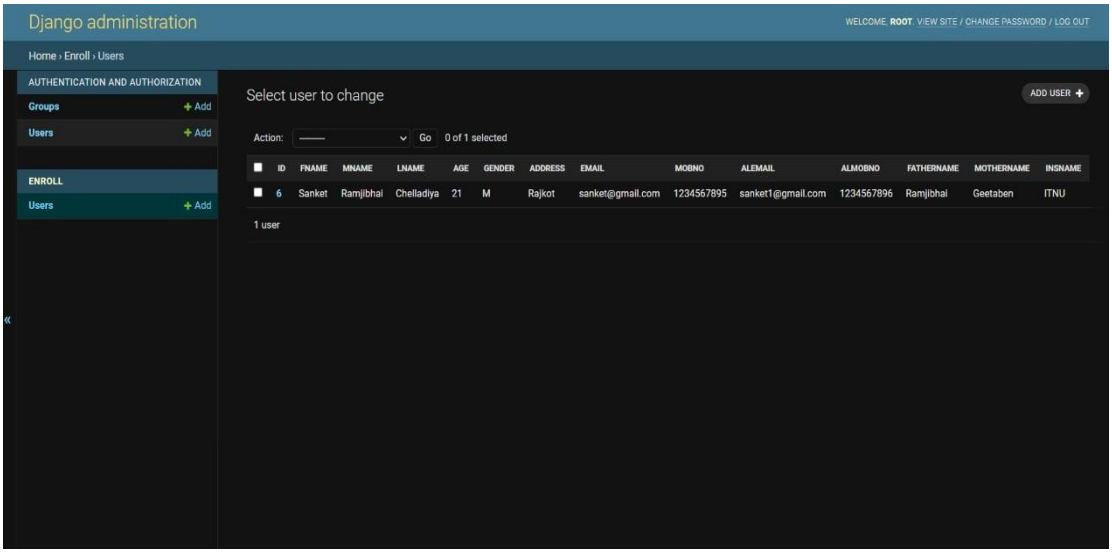
Home page

- **Add new Student Record**

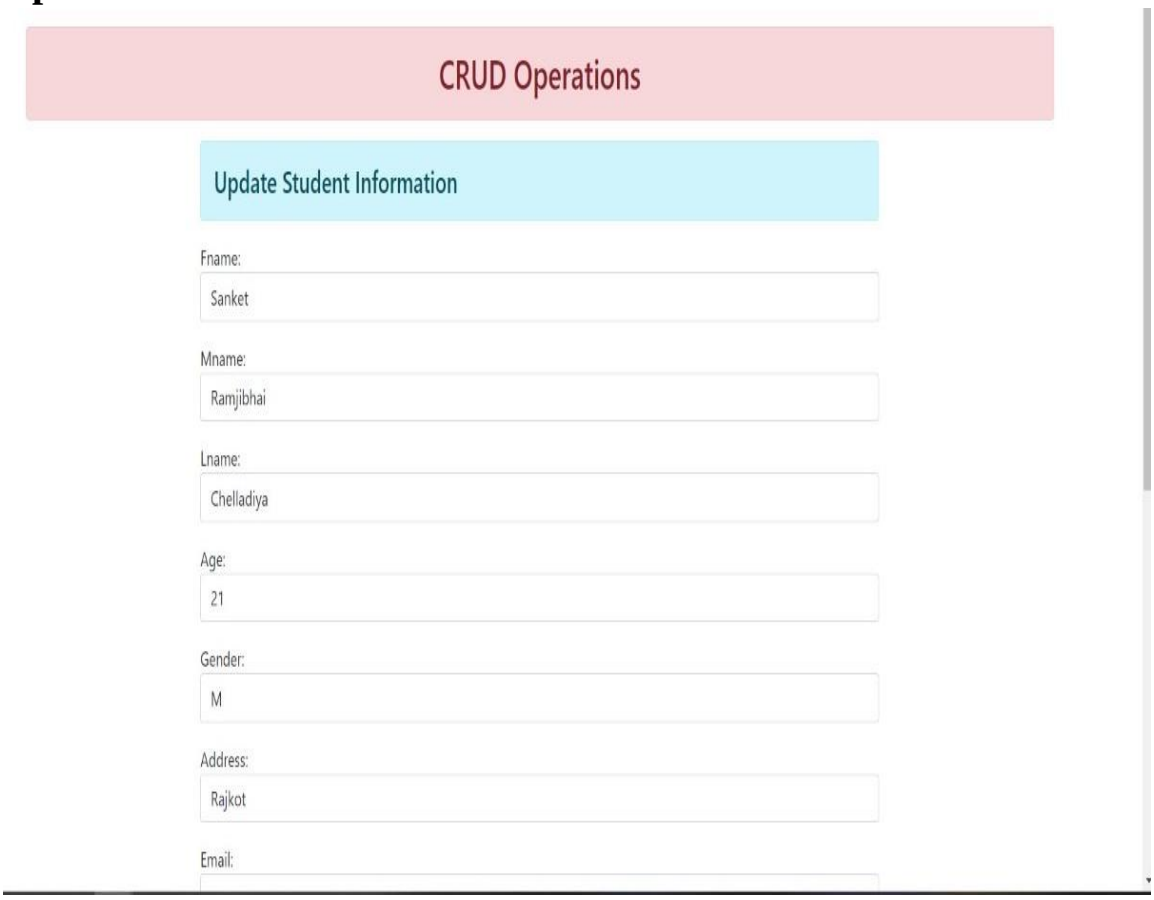


- **View database record into Home page along with Edit and Delete Option**

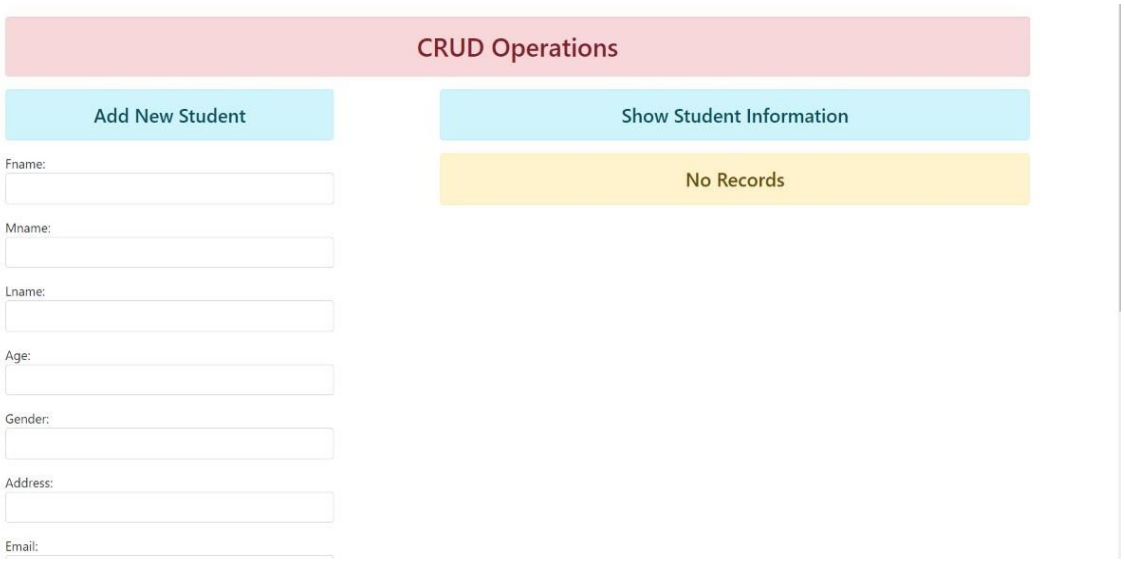
- **Database Records**



- **Update Button Clicked**



- **Delete Button Clicked**



----------Thank you----------