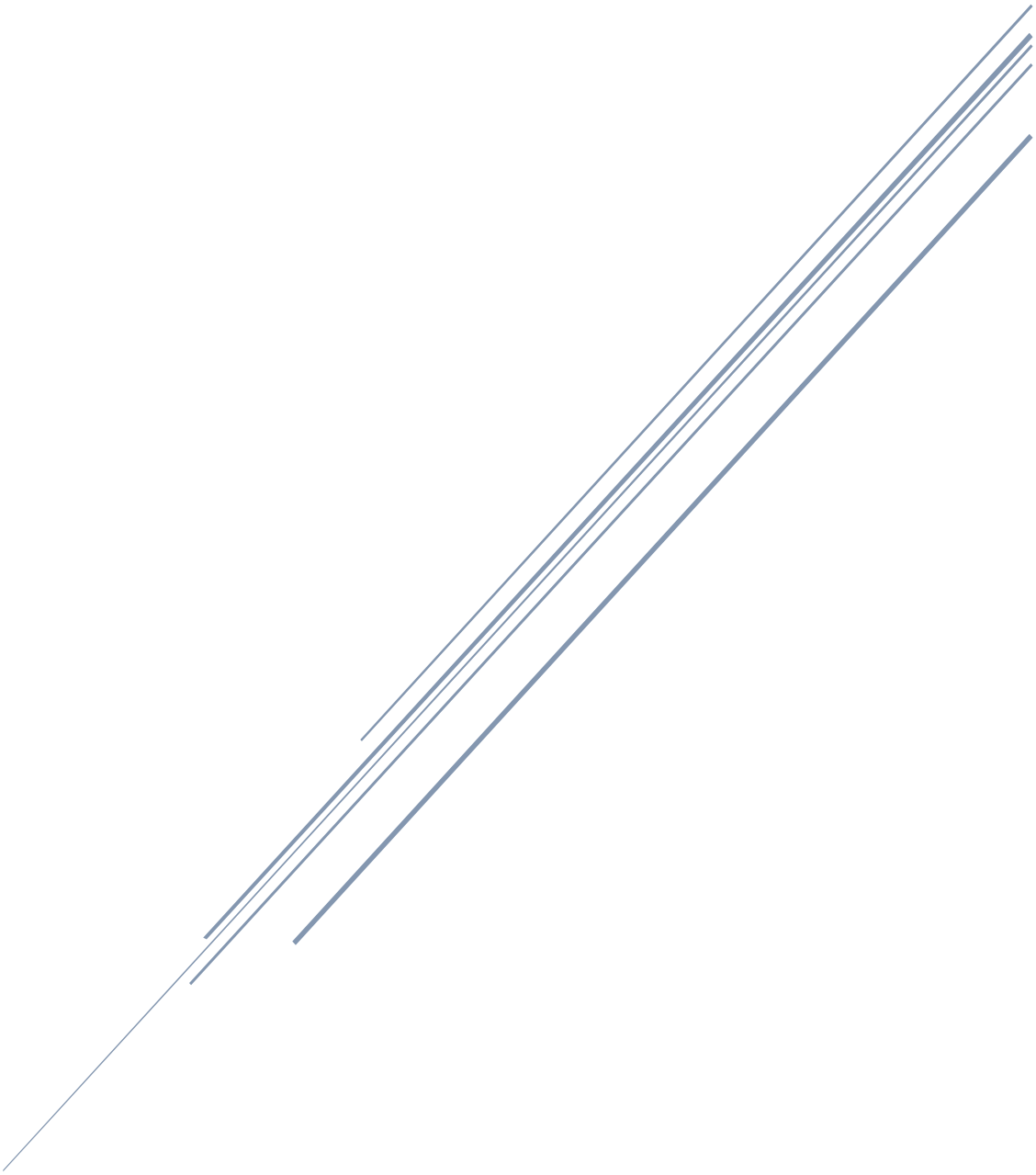
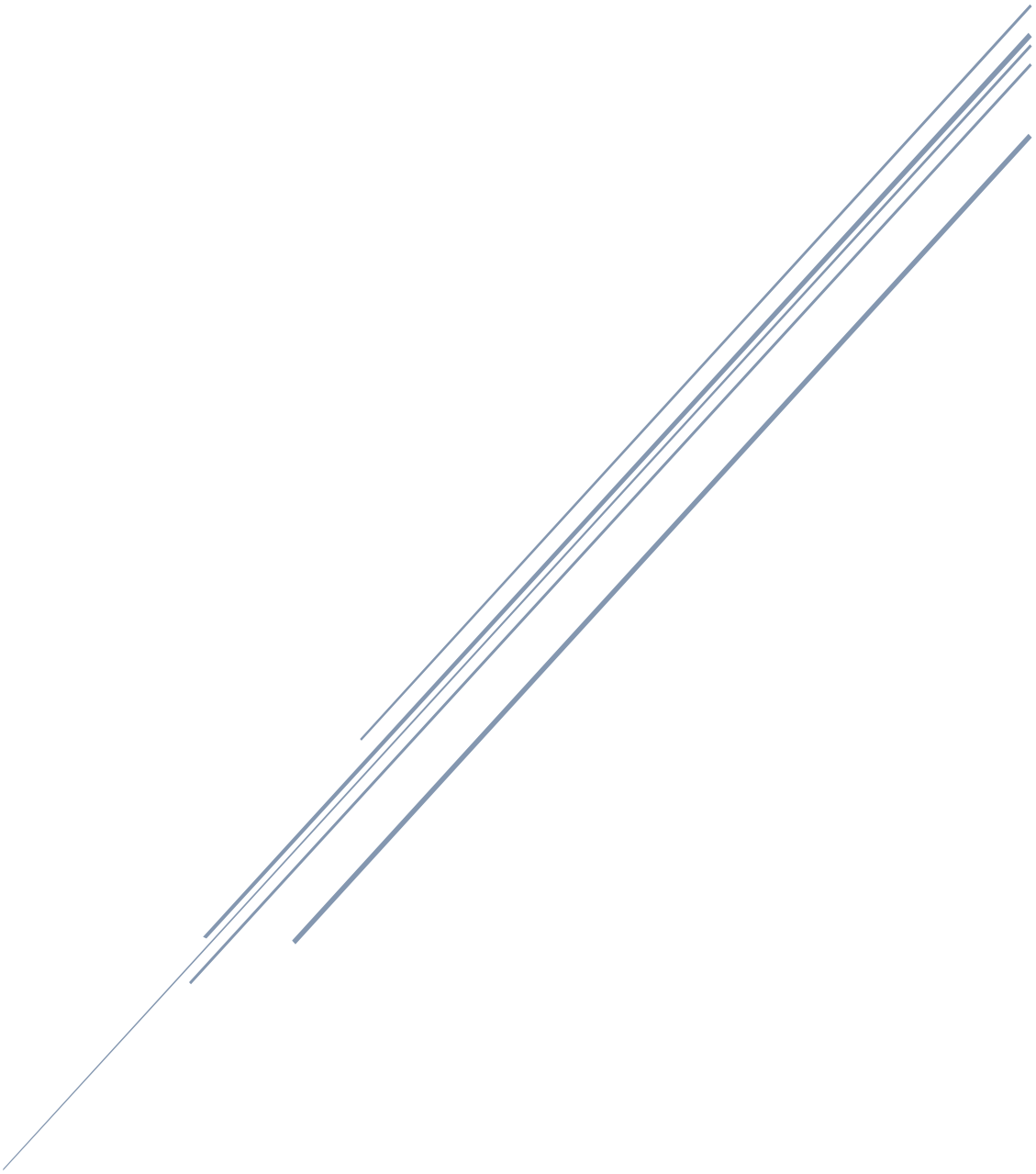
# COURSEWORK CODE

Danny Burrows



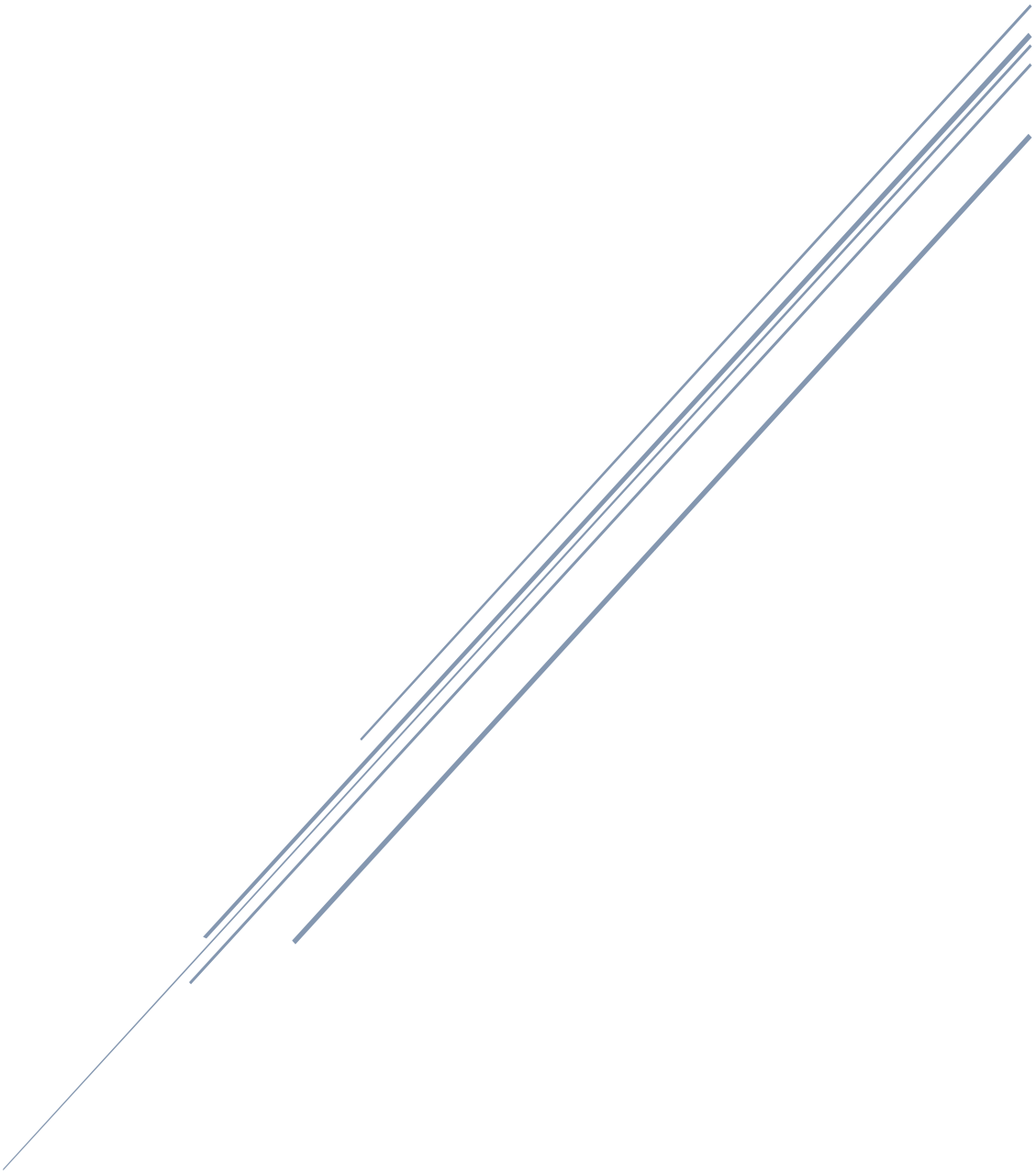
Scarborough Sixth Form College
Computer Science Coursework

# __Init__.py



```python
# Initialization file contains the server settings and config options. It also
# builds the app and initializes the server.

import os
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from flask_mail import Mail

app = Flask(__name__)
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'
app.config['MAIL_SERVER'] = 'smtp.googlemail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = os.environ.get('EMAIL_USER')
app.config['MAIL_PASSWORD'] = os.environ.get('EMAIL_PASS')
mail = Mail(app)

app.config['PDF_FILE_DUMP'] = "projectCode/static/pdf_gen/"

from projectCode import routes
```

# forms.py



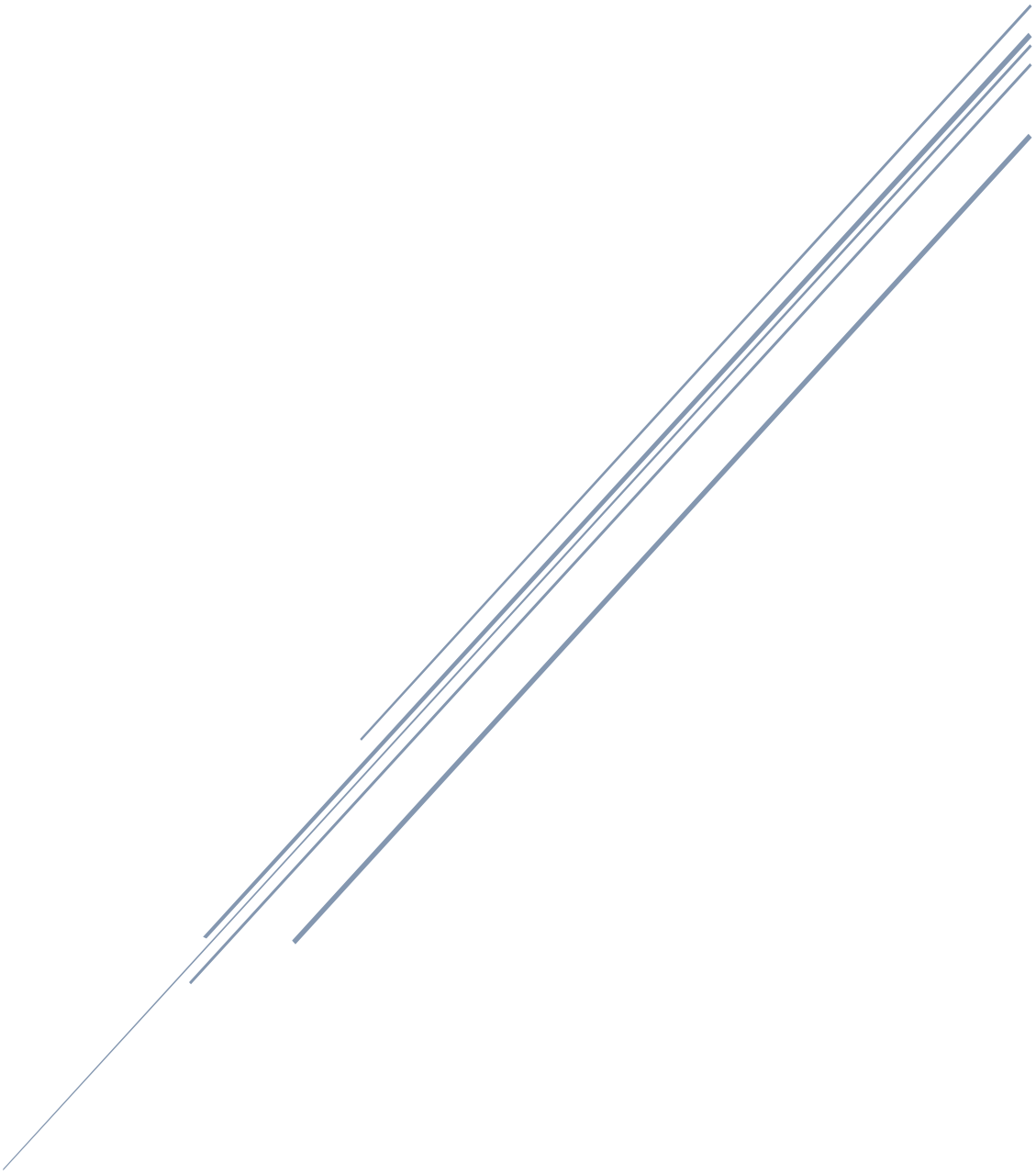
```
# This is all the form classes, they are imported and intansiated in the routes.py file
# and used to build form objects in the flask app which are validated on submission
# of the forms by the user.


from flask_wtf import FlaskForm, Form
from flask_wtf.file import FileField, FileAllowed
from flask_login import current_user
from wtforms import StringField, PasswordField, SubmitField, BooleanField, TextAreaField, DateField, IntegerField,
SelectField, FormField, FieldList
from wtforms.validators import DataRequired, Optional, Length, Email, NumberRange, EqualTo, ValidationError
from wtforms.fields.html5 import EmailField, DateField
from projectCode.models import User, Class, Course
import datetime
import re


# Generic search form for students, topics, users, posts e.t.c.
class SearchForm(FlaskForm):
    search_query = StringField('Search', validators=[DataRequired()])
    submit = SubmitField('đ')


# Registration form for new users.
class RegistrationForm(FlaskForm):
    username = StringField('Username',
                           validators=[DataRequired(), Length(min=2, max=20)])
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Sign Up')

    # Checking that the username entered is unique.
    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a different one.')

    # Checking that email is not taken by another user.
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please choose a different one.')


class LoginForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me') # Check if the user wants a cookie for login.
    submit = SubmitField('Login')


class UpdateAccountForm(FlaskForm):
    username = StringField('Username',
                           validators=[DataRequired(), Length(min=2, max=20)])
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    picture = FileField('Update Profile Picture', validators=[FileAllowed(['jpg', 'png'])])
    submit = SubmitField('Update')

    # Checking that the username entered is unique.
    def validate_username(self, username):
        if username.data != current_user.username:
```

```
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is taken. Please choose a different one.')

    # Checking that email is not taken by another user.
    def validate_email(self, email):
        if email.data != current_user.email:
            user = User.query.filter_by(email=email.data).first()
            if user:
                raise ValidationError('That email is taken. Please choose a different one.')


class PostForm(FlaskForm):
    title = StringField('Title', validators=[DataRequired(), Length(max=100)])
    content = TextAreaField('Content', validators=[DataRequired()])
    submit = SubmitField('Post')


class RequestResetForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that email. You must register first.')


class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
                                     validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Reset Password')


class ClassForm(FlaskForm):
    class_name = StringField('Class Name', validators=[DataRequired(), Length(max=50)])
    course_id = SelectField('Add To Course', coerce=int)
    class_starting_date = DateField('Academic Start Date', validators=[DataRequired("Date before year 3000 is required")])
    submit = SubmitField('Submit')


class StudentForm(FlaskForm):
    class_id = SelectField('Add To Class', coerce=int, validators=[DataRequired()])
    name = StringField('Student Name', validators=[DataRequired(), Length(max=50)])
    email = EmailField('Student Email', validators=[DataRequired(), Email()])
    address = TextAreaField('Student Address', validators=[DataRequired()])
    parent_phone = StringField('Parent Phone Number', validators=[DataRequired()])
    predicted_grade = StringField('Predicted Grade', validators=[DataRequired()])
    submit = SubmitField('Submit')

    def validate_parent_phone(self, parent_phone):
        rule = re.compile(r'^(07[\d]{8,12}|447[\d]{7,11})$')
        if not rule.search(parent_phone.data):
            msg = "Invalid mobile number."
            raise ValidationError(msg)


class AddStudentToClass(FlaskForm):
    class_id = SelectField('Add To Class', coerce=int, validators=[DataRequired()])
    submit = SubmitField('Submit')


class RemoveStudentFromClass(FlaskForm):
    class_id = SelectField('Remove From Class', coerce=int, validators=[DataRequired()])
```

```
    submit = SubmitField('Submit')


class AddStudentsToClass(FlaskForm):
    students = SelectField('Add Students', coerce=int, validators=[DataRequired()])
    submit = SubmitField('Submit')


class TopicForm(FlaskForm):
    name = StringField('Topic Name', validators=[DataRequired(), Length(max=50)])
    course_id = SelectField('Add To Course', coerce=int)
    begin_date = DateField('Begin Date', validators=[DataRequired("Date before year 3000 is required")])
    end_date = DateField('End Date', validators=[DataRequired("Date before year 3000 is required")])
    submit = SubmitField('Submit')


class HomeworkForm(FlaskForm):
    name = StringField('Homework Name', validators=[DataRequired()])
    max_mark = IntegerField('Maximum Mark (Numbers Only)', validators=[DataRequired()])
    due_date = DateField('Due Date', validators=[DataRequired()])
    submit = SubmitField('Submit')

    def validate_max_mark(self, mark):
        if mark.data > 10000:
            raise ValidationError("Mark must be bellow 10000.")


class TestForm(FlaskForm):
    name = StringField('Test Name', validators=[DataRequired()])
    max_mark = IntegerField('Maximum Mark (Numbers Only)', validators=[DataRequired()])
    date = DateField('Date', validators=[DataRequired()])
    submit = SubmitField('Submit')


class ExamForm(FlaskForm):
    name = StringField('Exam Name', validators=[DataRequired(), Length(max=50)])
    date = DateField('Date', validators=[DataRequired("Date before year 3000 is required")])
    max_mark = IntegerField('Maximum Mark (Numbers Only)', validators=[DataRequired("Integer Required")])
    submit = SubmitField('Submit')

class CommentForm(FlaskForm):
    comment = TextAreaField('Leave A Comment', validators=[DataRequired()])
    submit = SubmitField('Submit')


class CourseForm(FlaskForm):
    name = StringField('Course Name', validators=[DataRequired(), Length(max=100)])
    start_date = DateField('Start Date', validators=[DataRequired("Date before year 3000 is required")])
    year_num = IntegerField('Course Length (In Years)', validators=[DataRequired("Must enter valid integer")])
    grade_system = SelectField('Grading System', coerce=int)
    submit = SubmitField('Submit')

    def validate_year_num(self, year_num):
        if year_num.data > 50:
            raise ValidationError("Year number cannot exceed 50 years.")


class HomeworkMarkForm(FlaskForm):
    marks = FieldList(StringField('Mark'), min_entries=5)
    submit = SubmitField('Submit')


class TestMarkForm(FlaskForm):
    marks = FieldList(StringField('Mark'), min_entries=5)
    submit = SubmitField('Submit')
```

```
class ExamMarkForm(FlaskForm):
    marks = FieldList(StringField('Mark'), min_entries=5)
    submit = SubmitField('Submit')
```

# models.py



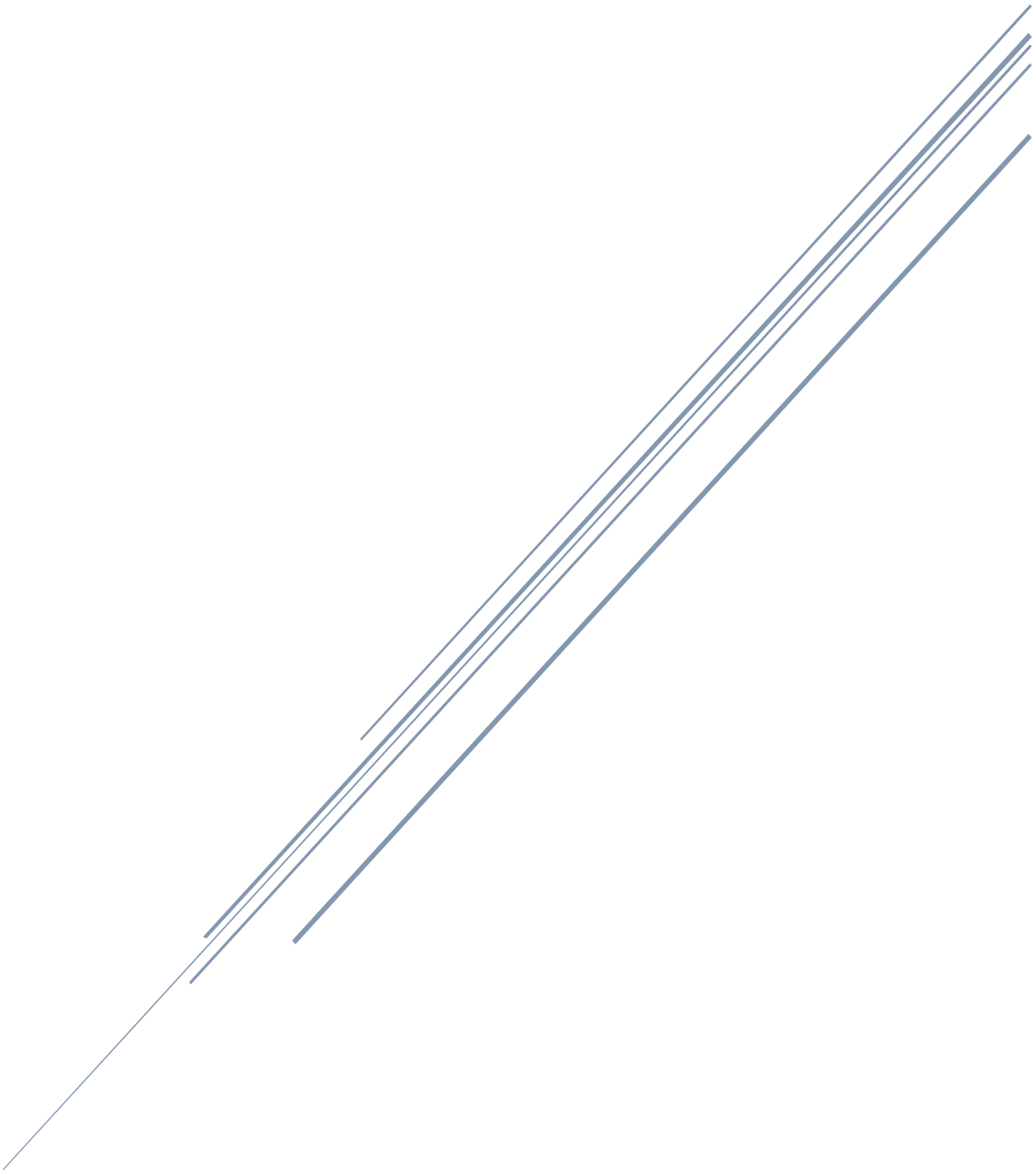
```python
# This file contains the database models, they are instansiated when the server
# is started and their objects are used to make all actions on the database.
# All database models contain the tables fields as attributes and all methods
# in the model are the processes used to manipulate their data.


from datetime import datetime
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from projectCode import db, login_manager, app
from flask_login import UserMixin


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


# The User model contains all the fields in the User table. It also contains the
# function needed to validate the password reset token when the user wishes to
# reset their password.
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='def.png')
    password = db.Column(db.String(60), nullable=False)

    posts = db.relationship('Post', backref='author', lazy=True)
    courses = db.relationship('Course', backref='teacher', lazy=True)
    comments = db.relationship('Comment', backref='commenter', lazy=True)


    def get_reset_token(self, expires_sec=1800):
        s = Serializer(app.config['SECRET_KEY'], expires_sec)
        return s.dumps({'user_id': self.id}).decode('utf-8')

    @staticmethod
    def verify_reset_token(token):
        s = Serializer(app.config['SECRET_KEY'])
        try:
            user_id = s.loads(token)['user_id']
        except:
            return None
        return User.query.get(user_id)

    def __repr__(self):
        return f"User('{self.username}', '{self.email}', '{self.image_file}')"


class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    date_posted = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.Text, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

    comments = db.relationship('Comment', backref='post', lazy=True)

    def __repr__(self):
        return f"Post('{self.title}', '{self.date_posted}')"


class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    date_posted = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.Text, nullable=False)
```

```
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'), nullable=False)

    def __repr__(self):
        return f"Comment('{self.content}', '{self.date_posted}')"

# Register link table between classes and students.
ClassRegister = db.Table('ClassRegister',
                db.Column('class_id', db.Integer, db.ForeignKey('class.id')),
                db.Column('student_id', db.Integer, db.ForeignKey('student.id')))


class Class(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    class_name = db.Column(db.String(100), nullable=False)
    class_starting_date = db.Column(db.DateTime, nullable=False)
    course_id = db.Column(db.Integer, db.ForeignKey('course.id'), nullable=False)

    students = db.relationship('Student', secondary=ClassRegister, backref=db.backref('classes'))

    def __repr__(self):
        return f"Class('{self.class_name}', '{self.class_starting_date}')"


class Student(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(35))
    address = db.Column(db.Text)
    parent_phone = db.Column(db.String(35))
    predicted_grade = db.Column(db.String(35))

    homework_marks = db.relationship('HomeworkMark', backref='student', lazy=True)
    test_marks = db.relationship('TestMark', backref='student', lazy=True)
    exam_marks = db.relationship('ExamMark', backref='student', lazy=True)

    def __repr__(self):
        return f"Student('{self.name}', '{self.email}', '{self.address}', '{self.parent_phone}', '{self.predicted_grade}')"


class Topic(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    course_id = db.Column(db.Integer, db.ForeignKey('course.id'), nullable=False)

    start_date = db.Column(db.DateTime, default=datetime.utcnow)
    end_date = db.Column(db.DateTime)

    homeworks = db.relationship('Homework', backref='topic', lazy=True)
    tests = db.relationship('Test', backref='topic', lazy=True)

    def __repr__(self):
        return f"Topic('{self.name}')"


class Homework(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    due_date = db.Column(db.DateTime, nullable=False)
    max_mark = db.Column(db.Integer, nullable=False)
    topic_id = db.Column(db.Integer, db.ForeignKey('topic.id'), nullable=False)

    homework_marks = db.relationship('HomeworkMark', backref='homework', lazy=True)

    def __repr__(self):
        return f"Homework('{self.name}', '{self.due_date}', '{self.topic_id}', '{self.max_mark}')"
```

```
class HomeworkMark(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    mark = db.Column(db.Integer, nullable=False)
    grade = db.Column(db.String(10), nullable=True)
    date_given = db.Column(db.DateTime, default=datetime.utcnow)
    date_handed_in = db.Column(db.DateTime, default=datetime.utcnow)
    homework_id = db.Column(db.Integer, db.ForeignKey('homework.id'), nullable=False)
    student_id = db.Column(db.Integer, db.ForeignKey('student.id'), nullable=False)

    def __repr__(self):
        return f"HomeworkMark('{self.mark}', '{self.grade}', '{self.date_given}', '{self.date_handed_in}', '{self.homework_id}',
'{self.student_id}')"


class Test(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.DateTime, nullable=False)
    topic_id = db.Column(db.Integer, db.ForeignKey('topic.id'), nullable=False)
    max_mark = db.Column(db.Integer, nullable=False)
    topic_id = db.Column(db.Integer, db.ForeignKey('topic.id'), nullable=False)

    test_marks = db.relationship('TestMark', backref='test', lazy=True)

    def __repr__(self):
        return f"Test('{self.name}', '{self.date}', '{self.topic_id}', '{self.max_mark}')"


class TestMark(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    mark = db.Column(db.Integer, nullable=False)
    grade = db.Column(db.String(10), nullable=True)
    date_completed = db.Column(db.DateTime, default=datetime.utcnow)
    test_id = db.Column(db.Integer, db.ForeignKey('test.id'), nullable=False)
    student_id = db.Column(db.Integer, db.ForeignKey('student.id'), nullable=False)

    def __repr__(self):
        return f"TestMark('{self.mark}', '{self.grade}')"


class Exam(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    date = db.Column(db.DateTime, nullable=False)
    max_mark = db.Column(db.Integer, nullable=False)
    course_id = db.Column(db.Integer, db.ForeignKey('course.id'), nullable=False)

    exam_marks = db.relationship('ExamMark', backref='exam', lazy=True)

    def __repr__(self):
        return f"Exam('{self.name}', '{self.date}', '{self.topic_id}', '{self.grade_system}', '{self.max_mark}')"


class ExamMark(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    mark = db.Column(db.Integer, nullable=False)
    grade = db.Column(db.String(10), nullable=True)
    date_completed = db.Column(db.DateTime, default=datetime.utcnow)
    exam_id = db.Column(db.Integer, db.ForeignKey('exam.id'), nullable=False)
    student_id = db.Column(db.Integer, db.ForeignKey('student.id'), nullable=False)

    def __repr__(self):
        return f"ExamMark('{self.mark}', '{self.grade}')"
```

```python
class Course(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)

    start_date = db.Column(db.DateTime, default=datetime.utcnow)
    year_num = db.Column(db.Integer, nullable=False)

    grading_system = db.Column(db.String(10), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

    classes = db.relationship('Class', backref='course', lazy=True)
    topics = db.relationship('Topic', backref='course', lazy=True)
    exams = db.relationship('Exam', backref='course', lazy=True)

    def __repr__(self):
        return f"Course('{self.name}', '{self.grading_system}')"
```

# routes.py



```
# This file specifies the logic to be executed when a user requests a known url from the webserver.
# It's arranged in functions that are called when the route parameter in the decorator of the function
# is requested from the client.

# The following imports are mostly tools flask provide to build a webserver which is capable of
# responding to users POST and GET requests and build custom html templates.
import os
import secrets
from datetime import datetime, timedelta
from collections import OrderedDict
from PIL import Image
from flask import (render_template, url_for, flash, redirect, request, abort,
            Markup)
from projectCode import app, db, bcrypt, mail
from projectCode.forms import (RegistrationForm, LoginForm, UpdateAccountForm,
                        PostForm, RequestResetForm, ResetPasswordForm,
                        ClassForm, StudentForm, AddStudentToClass,
                        RemoveStudentFromClass, TopicForm, HomeworkForm,
                        TestForm, CommentForm, CourseForm, ExamForm,
                        HomeworkMarkForm, TestMarkForm, ExamMarkForm,
                        SearchForm)
from projectCode.models import (User, Post, Class, Student, Topic, Homework,
                        Test, Comment, Course, Exam, HomeworkMark,
                        TestMark, ExamMark)
from flask_login import login_user, current_user, logout_user, login_required
from flask_mail import Message


class GraphDataset():
    def __init__(self, label):
        self.label = label
        self.data = []


class Graph():
    def __init__(self, title, type):
        self.title = title
        self.type = type
        self.labels = []
        self.datasets = []


# DEPRECATED
def monthList(dates):
    start, end = [datetime.strptime(_, "%Y-%m-%d") for _ in dates]
    total_months = lambda dt: dt.month + 12 * dt.year
    mlist = []
    for tot_m in range(total_months(start)-1, total_months(end)):
        y, m = divmod(tot_m, 12)
        mlist.append(datetime(y, m+1, 1).strftime("%b-%y"))
    return mlist


def exMonthList(start, end):
    # start, end = [datetime.strptime(_, "%Y-%m-%d") for _ in dates]
    total_months = lambda dt: dt.month + 12 * dt.year
    mlist = []
    for tot_m in range(total_months(start)-1, total_months(end)):
        y, m = divmod(tot_m, 12)
        mlist.append(datetime(y, m+1, 1).strftime("%b-%y"))
    return mlist



# Any route which has a @login_required() decorator will check if the current user has a recognised
# session id, if so it will attempt to log them into the specified session. If the user isnt
```

```
# authenticated they will be redirected to a login page.

# The home route, which is also the default url when the site is accessed will fetch posts from users
# and display them. The program will organise the posts in pages with the ability to paginate between
# them. The home.html template is then built by flask using the posts passed as a parameter and the
# logic defined in the html page. Flask uses ********jinger to build custom templates in html, then
# serve them to the client, the logic for the custom templates is specified in the html page iself.
@app.route("/", methods=['GET', 'POST'])
@app.route("/home", methods=['GET', 'POST'])
def home():
    # Initializing the search and comment forms...
    s_form = SearchForm()
    c_form = CommentForm()
    page = request.args.get('page', 1, type=int)
    posts = Post.query.order_by(Post.date_posted.desc())

    # If a post request is recieved then the following if statement will run...
    srch = 0
    if s_form.validate_on_submit():
        # Getting the search results...
        search_results = posts.filter(Post.title.like('%' + s_form.search_query.data + '%'))
        search_user_results = User.query.filter(User.username.like('%' + s_form.search_query.data + '%'))
        srch = 1 # Search has been performed and results will be shown.
    else:
        search_results = []
        search_user_results = []

    posts = posts.paginate(page=page, per_page=5)
    return render_template('home.html',
                           posts=posts,
                           form=c_form,
                           s_form=s_form,
                           srch=srch,
                           search_results=search_results,
                           search_user_results=search_user_results)

# The dash route contains all the calculation and response for the data displayed
# in the dashboard page.
@app.route("/dash")
def dash():

    courses = Course.query.filter_by(teacher=current_user)

    cur_tops = []
    for course in courses:
        for topic in course.topics:
            if datetime.utcnow() > topic.start_date and datetime.utcnow() < topic.end_date:
                cur_tops.append([course, topic])

    upcoming_homeworks, due_homeworks = [], []
    for cur_top in cur_tops:
        for homework in cur_top[1].homeworks:
            if homework.due_date > datetime.utcnow():
                upcoming_homeworks.append([homework, cur_top[1]])
            elif homework.due_date <= datetime.utcnow():
                due_homeworks.append([homework, cur_top[1]])


    upcoming_tests, due_tests = [], []
    for cur_top in cur_tops:
        for test in cur_top[1].tests:
            if test.date > datetime.utcnow():
                upcoming_tests.append([test, cur_top[1]])
            elif test.date <= datetime.utcnow():
                due_tests.append([test, cur_top[1]])
```

```
cur_top_classes = set()
for cur_top in cur_tops:

    cur_top_classes.update(cur_top[0].classes)


homework_marks = {}
for homework in due_homeworks:
    homework_marks[homework[0].id] = {}

    for xclass in cur_top_classes:

        for student in xclass.students:
            mark = HomeworkMark.query.filter_by(homework_id=homework[0].id).filter_by(student_id=student.id).all()
            homework_marks[homework[0].id][student.id] = mark


            # CHECK IF HANDED IN LATE
            try:
                if len(homework_marks[homework[0].id][student.id]):
                    hmwk = Homework.query.filter_by(id=mark[0].homework_id).first_or_404()
                    if homework_marks[homework[0].id][student.id][0].date_handed_in > hmwk.due_date:
                        print("")
                        homework_marks[homework[0].id][student.id][0].late = True
            except:
                pass



test_marks = {}
for test in due_tests:
    test_marks[test[0].id] = {}

    for xclass in cur_top_classes:

        for student in xclass.students:
            mark = TestMark.query.filter_by(test_id=test[0].id).filter_by(student_id=student.id).all()
            test_marks[test[0].id][student.id] = mark


            # CHECK IF HANDED IN LATE
            try:
                if len(test_marks[test[0].id][student.id]):
                    tst = Test.query.filter_by(id=mark[0].test_id).first_or_404()
                    if test_marks[test[0].id][student.id][0].date_completed > tst.date:
                        print("")
                        test_marks[test[0].id][student.id][0].late = True
            except Exception as e:
                print("PASSING DATE CHECK (PROBABLY HAND IN DATE IS NULL)", test_marks[test[0].id][student.id][0])
                pass


# Class performance in last test graph.
class_perf = Graph("Avg Class Performance On Last Test", "bar")
class_perf_data = {"label":"Avg Class Performance Last Test", "data":[]}

for xclass in cur_top_classes:
    class_perf.labels.append(xclass.class_name)

cls_avgs = []
mrks = []
for test in due_tests:
    for xclass in cur_top_classes:

        for student in xclass.students:
```

```
            try:
                mrks.append(TestMark.query.filter_by(test_id=test[0].id).filter_by(student_id=student.id).all()[0].mark)
            except IndexError:
                pass


        try:
            avg = sum(mrks) / len(mrks)
            class_perf_data["data"].append(avg)
        except ZeroDivisionError:
            class_perf_data["data"].append(0)
        mrks = []

class_perf.datasets.append(class_perf_data)


classes_perf_time = Graph("Avg Classes Performance Over Time", "line")

dates = ["2014-10-10", "2016-01-07"]
classes_perf_time.labels = monthList(dates)


courses = db.session.query(Course.id).filter_by(teacher=current_user)
classes = Class.query.filter(Class.course_id.in_(courses)).all()


# Getting date span for graph...
topics_start_asc = Topic.query.filter(Topic.course_id.in_(courses)).order_by(Topic.start_date.asc()).all()
topics_end_desc = Topic.query.filter(Topic.course_id.in_(courses)).order_by(Topic.end_date.desc()).all()

if len(topics_start_asc) and len(topics_end_desc):

    earliest_start = topics_start_asc[0].start_date
    latest_end = topics_end_desc[0].end_date

    dates = [earliest_start, latest_end]
    classes_perf_time.labels = exMonthList(earliest_start, latest_end)


    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses)).all()


    import random

    for xclass in classes:
        class_perf_data = GraphDataset(xclass.class_name)
        class_perf_data = {"label":xclass.class_name, "data":[]}

        for i in range(30):
            class_perf_data["data"].append(random.randint(0,100))

        classes_perf_time.datasets.append(class_perf_data)


# Class performance per assignment test.
classes_perf_per_test = Graph("Avg Classes Performance Over Time", "line")


courses = db.session.query(Course.id).filter_by(teacher=current_user)
topics = Topic.query.filter(Topic.course_id.in_(courses)).all()
classes = Class.query.filter(Class.course_id.in_(courses)).all()

temp_data = []
labels = []
```

```
    for xclass in classes:
        classes_perf_per_test_data = {"label":xclass.class_name, "data":[]}

        for topic in topics:

            for hmwk in topic.homeworks:
                labels.append("HMWK "+topic.name+" "+hmwk.name)

                # Get avg perf
                mrks = []
                for student in xclass.students:
                    try:
                        mrks.append(HomeworkMark.query.filter_by(homework_id=hmwk.id).filter_by(student_id=student.id).all()
[0].mark)
                    except IndexError:
                        pass

                c_sum = 0
                c_len = 0
                for mrk in mrks:
                    try:
                        c_sum += mrk
                        c_len += 1
                    except TypeError:
                        pass


                if c_len:
                    avg = c_sum // c_len
                    classes_perf_per_test_data["data"].append(avg)
                else:
                    classes_perf_per_test_data["data"].append(0)

                mrks = []


            for test in topic.tests:
                labels.append("TEST "+topic.name+" "+test.name)

                # Get avg perf
                mrks = []
                for student in xclass.students:
                    try:
                        mrks.append(TestMark.query.filter_by(test_id=test.id).filter_by(student_id=student.id).all()[0].mark)
                    except IndexError:
                        pass

                c_sum = 0
                c_len = 0
                for mrk in mrks:
                    try:
                        c_sum += mrk
                        c_len += 1
                    except TypeError:
                        pass


                if c_len:
                    avg = c_sum // c_len
                    classes_perf_per_test_data["data"].append(avg)
                else:
                    classes_perf_per_test_data["data"].append(0)

                mrks = []
```

```
        classes_perf_per_test.datasets.append(classes_perf_per_test_data)

    # Remove duplicates...
    n_labels = []
    for label in labels:
        if not label in n_labels:
            n_labels.append(label)

    classes_perf_per_test.labels = n_labels


    return render_template('dash.html', cur_tops=cur_tops,
                           upcoming_homeworks=upcoming_homeworks,
                           due_homeworks=due_homeworks,
                           due_tests=due_tests,
                           homework_marks=homework_marks,
                           test_marks=test_marks,
                           class_perf=class_perf,
                           classes_perf_time=classes_perf_time,
                           classes_perf_per_test=classes_perf_per_test,
                           cur_top_classes=cur_top_classes)



# Route to allow user to post the result of a homework via the dashboard.
@app.route('/submit_homework', methods=['POST'])
def submit_homework():
    mark = request.form['mark']
    date_handed_in = request.form['date_handed_in']
    homework_id = request.form['homework_id']
    student_id = request.form['student_id']

    try:
        int(mark)
    except ValueError:
        flash('Mark must be an int.', 'danger')
        return redirect(url_for('dash'))

    homework = Homework.query.first_or_404(homework_id)
    print(mark)
    print(homework.max_mark)
    if int(mark) > homework.max_mark:
        flash('Mark is more than the Homeworks maximum mark.', 'danger')
        return redirect(url_for('dash'))


    homework_mark = HomeworkMark(mark=mark,
                           homework_id=homework_id,
                           student_id=student_id,
                           date_handed_in=datetime.strptime(date_handed_in, '%Y-%m-%d'))

    db.session.add(homework_mark)
    db.session.commit()
    flash('Homework has been marked!', 'success')
    return redirect(url_for('dash'))


# Route to allow user to post the result of a test via the dashboard.
@app.route('/submit_test', methods=['POST'])
def submit_test():
    mark = request.form['mark']
    date_completed = request.form['date_handed_in']
    test_id = request.form['test_id']
    student_id = request.form['student_id']
```

```
    try:
        int(mark)
    except ValueError:
        flash('Mark must be an int.', 'danger')
        return redirect(url_for('dash'))

    test = Test.query.first_or_404(test_id)
    if int(mark) > test.max_mark:
        flash('Mark is more than the Tests maximum mark.', 'danger')
        return redirect(url_for('dash'))


    test_mark = TestMark(mark=mark,
                         test_id=test_id,
                         student_id=student_id,
                         date_completed=datetime.strptime(date_completed, '%Y-%m-%d'))

    db.session.add(test_mark)
    db.session.commit()
    flash('Test has been marked!', 'success')
    return redirect(url_for('dash'))

# The comment route is for users to sumbit a comment to a post.
@app.route('/post/<int:post_id>/comment', methods=['POST'])
@login_required
def add_comment(post_id):
    post = Post.query.get_or_404(post_id)
    comment_content = request.form['comment']

    if len(comment_content) > 250:
        flash('Comment is too long!', 'danger')
        return redirect(url_for('post', post_id=post_id))

    if not len(comment_content):
        flash('No comment found!', 'danger')
        return redirect(url_for('post', post_id=post_id))

    comment = Comment(content=comment_content, commenter=current_user, post=post)
    db.session.add(comment)
    db.session.commit()
    flash('Your comment has been posted!', 'success')
    return redirect(url_for('post', post_id=post_id))

# '/post/<int:post_id>/comment/<int:comment_id>/update'
@app.route('/comment/<int:comment_id>/update', methods=['POST'])
@login_required
def update_comment(comment_id):
    comment = Comment.query.get_or_404(comment_id)
    post_id = int(comment.post_id)
    if comment.commenter != current_user:
        abort(403)

    comment_content = request.form['comment']

    print(comment_content)

    if len(comment_content) > 500:
        return "Post is too long!", 201

    if len(comment_content) < 2:
        return "Post content too short!", 201


    comment.content = comment_content
    db.session.commit()
```

```
    return "Your Post Was Updated Successfully!", 201


@app.route("/post/<int:post_id>/comment/<int:comment_id>/delete", methods=['GET'])
@login_required
def delete_comment(post_id, comment_id):
    comment = Comment.query.get_or_404(comment_id)
    if comment.commenter != current_user:
        abort(403)
    db.session.delete(comment)
    db.session.commit()
    flash('Your comment has been deleted!', 'success')
    return redirect(url_for('post', post_id=post_id))


# The about route simply returns a static about page which explains the system is some detail and
# the setup process.
@app.route("/about")
def about():
    return render_template('about.html', title='About')


# The register route can be called with multiple methods, post and get. It first checks if the user
# is already signed in, if so the user is redirected to the home page. If the user isn't signed in
# then a form is build using the specification in the forms.py file. "form.validate_on_submit()"
# basically checks if the method of requests is POST, it then validates the user input based upon
# the validation rules that are specified in the forms.py file. If the method of request is GET,
# then the system simply returns the register.html page with the necissary options for a user to
# register an account. If validation is successful the user is added to the database.
@app.route("/register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data, password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your account has been created! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('acc_register.html', title='Register', form=form)


# The login route works very simularly to the register route however there is a extra option
# for if the validation and verification of user input is failed. Messages are "flashed" to the
# user if login is successful, or unsuccussful.
@app.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash('Login Unsuccessful. Please check email and password', 'danger')
    return render_template('acc_login.html', title='Login', form=form)


# Route calls to the "logout_user()" function which simply terminates the users session and
# ensures that their session id is not usable anymore.
@app.route("/logout")
```

```python
def logout():
    logout_user()
    return redirect(url_for('home'))


# This route is for saving a picture when the user wishes to update their profile picture.
def save_picture(form_picture):
    random_hex = secrets.token_hex(8)
    _, f_ext = os.path.splitext(form_picture.filename)
    picture_fn = random_hex + f_ext
    picture_path = os.path.join(app.root_path, 'static/profile_pics', picture_fn)

    output_size = (125, 125)
    i = Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)

    return picture_fn


# Route checks if user is authenticated, then gets and displays their account information.
# This method also takes post requests and allows the user to update their account information.
@app.route("/account", methods=['GET', 'POST'])
@login_required
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated!', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
    image_file = url_for('static', filename='profile_pics/' + current_user.image_file)
    return render_template('account.html', title='Account',
                           image_file=image_file, form=form)


# Allows the user to make a new post and post to the server. The GET method, responds with
# a post enter page, and the POST method stores the information to the database.
@app.route("/post/new", methods=['GET', 'POST'])
@login_required
def new_post():
    form = PostForm()
    if form.validate_on_submit():
        post = Post(title=form.title.data, content=form.content.data, author=current_user)
        db.session.add(post)
        db.session.commit()
        flash('Your post has been created!', 'success')
        return redirect(url_for('home'))
    return render_template('new_post.html', title='New Post',
                           form=form, legend='New Post')


# Allows the user to view a specified post, if the user owns the post then they will have
# the option to edit the title or contence of the post while other users can only view the
# contence of the post.
@app.route("/post/<int:post_id>")
def post(post_id):
    form = CommentForm()
    post = Post.query.get_or_404(post_id)
```

```
    comments = Comment.query.filter_by(post=post)
    return render_template('single_post.html', title=post.title, form=form, post=post, comments=comments)


# Checks for the user to be authenticated and allows them to update the post if the post
# belongs to them.
@app.route("/post/<int:post_id>/update", methods=['GET', 'POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    form = PostForm()
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        flash('Your post has been updated!', 'success')
        return redirect(url_for('post', post_id=post.id))
    elif request.method == 'GET':
        form.title.data = post.title
        form.content.data = post.content
    return render_template('new_post.html', title='Update Post',
                           form=form, legend='Update Post')


# If the post author matches the current user then the post specified in
# the url will be deleted.
@app.route("/post/<int:post_id>/delete", methods=['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    for comment in post.comments:
        db.session.delete(comment)
    db.session.delete(post)
    db.session.commit()
    flash('Your post has been deleted!', 'success')
    return redirect(url_for('home'))


# This route is here to allow any user to see all posts of a specified user. It
# will return all of the posts whos author corrisponds to the user specified.
@app.route("/user/<string:username>")
def user_posts(username):
    page = request.args.get('page', 1, type=int)
    user = User.query.filter_by(username=username).first_or_404()
    posts = Post.query.filter_by(author=user)\
        .order_by(Post.date_posted.desc())\
        .paginate(page=page, per_page=5)
    return render_template('all_user_posts.html', posts=posts, user=user)


# When this function is called by the server, a password reset email will be
# sent to the user specified in the parameter.
def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
                  sender='noreply@demo.com',
                  recipients=[user.email])
    msg.body = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}

If you did not make this request then simply ignore this email and no changes will be made.
'''
```

```
    mail.send(msg)


# Route is used by a user to request a password reset on their account. (Current user is
# to identify the user).
@app.route("/reset_password", methods=['GET', 'POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset your password.', 'info')
        return redirect(url_for('login'))
    return render_template('reset_request.html', title='Reset Password', form=form)


# Checks for a token we can use to identify a user that has requested a password reset.
# If the user has requested a reset and is authenticated.
@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    user = User.verify_reset_token(token)
    if user is None:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user.password = hashed_password
        db.session.commit()
        flash('Your password has been updated! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('reset_token.html', title='Reset Password', form=form)


# Route for creating a new class. If POST method is used then user input is validated and
# class is added to database if all validation rules are passed.
@app.route("/new_class", methods=['GET', 'POST'])
@login_required
def new_class():

    form = ClassForm()

    courses = Course.query.filter_by(teacher=current_user).all()
    form.course_id.choices = [(course.id, course.name) for num, course in enumerate(courses)]

    if not len(courses):
        flash(Markup('To add a class you must first <a href="' + url_for('new_course') + '">add a course</a>.'), 'danger')

    # If user doesn't request for a course, then enable the select box and set the result as the course_id...
    try:
        course_id = request.args['course_id']
        form.course_id.value = course_id
    except KeyError:
        form.course_id.enabled = True
        course_id = form.course_id.data

    if form.validate_on_submit():

        course = Course.query.get_or_404(course_id)
        if course.teacher != current_user:
            abort(403)
```

```
        new_class = Class(class_name=form.class_name.data,
                          class_starting_date=form.class_starting_date.data,
                          course=course)

        db.session.add(new_class)
        db.session.commit()
        flash(f'New class has been added to {course.name}!', 'success')

        next_page = request.args.get('next')
        return redirect(next_page) if next_page else redirect(url_for('classes'))

    return render_template('new_class.html', title='Create Class', form=form, legend="Add A Class To Manage")


# Route to allow the user to view all of their classes, the classes are queried from the database and the "teacher" parameter is used
# to ensure only classes belonging to the current user is returned.
@app.route("/classes")
@login_required
def classes():
    page = request.args.get('page', 1, type=int)
    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses)).paginate(page=page, per_page=5)
    return render_template('all_classes.html', title="My Classes", classes=classes, user=current_user)


# Allows the user to view all the details of a specific class who's id will be passed into the url. This is mostly done
# automatically by a button on the interface. If the user is trying to access a class that they do not teach then a 403
# FORBIDDEN responce is issued.
@app.route("/class/<int:class_id>")
@login_required
def xclass(class_id):
    xclass = Class.query.get_or_404(class_id)
    if xclass.course.teacher != current_user:
        abort(403)



    classes_perf_per_test = Graph("Avg Classes Performance Over Time", "line")


    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    topics = Topic.query.filter(Topic.course_id.in_(courses)).all()
    classes = Class.query.filter(Class.course_id.in_(courses)).all()

    temp_data = []
    labels = []



    print(xclass.class_name)
    classes_perf_per_test_data = {"label":xclass.class_name, "data":[]}

    for topic in topics:

        for hmwk in topic.homeworks:
            print(hmwk.name)
            labels.append("HMWK "+topic.name+" "+hmwk.name)

            # Get avg perf
            mrks = []
            for student in xclass.students:
                try:
                    mrks.append(HomeworkMark.query.filter_by(homework_id=hmwk.id).filter_by(student_id=student.id).all()[0].mark)
                except IndexError:
                    pass
```

```
        c_sum = 0
        c_len = 0
        for mrk in mrks:
            try:
                c_sum += mrk
                c_len += 1
            except TypeError:
                pass


        if c_len:
            avg = c_sum // c_len
            classes_perf_per_test_data["data"].append(avg)
        else:
            classes_perf_per_test_data["data"].append(0)

        mrks = []


    for test in topic.tests:
        print(test.name)
        labels.append("TEST "+topic.name+" "+test.name)

        # Get avg perf
        mrks = []
        for student in xclass.students:
            try:
                mrks.append(TestMark.query.filter_by(test_id=test.id).filter_by(student_id=student.id).all()[0].mark)
            except IndexError:
                pass

        c_sum = 0
        c_len = 0
        for mrk in mrks:
            try:
                c_sum += mrk
                c_len += 1
            except TypeError:
                pass


        if c_len:
            avg = c_sum // c_len
            classes_perf_per_test_data["data"].append(avg)
        else:
            classes_perf_per_test_data["data"].append(0)

        mrks = []



print("\n\n", classes_perf_per_test_data["data"])


classes_perf_per_test.datasets.append(classes_perf_per_test_data)

# Get avg perf

# Remove duplicates...
n_labels = []
for label in labels:
    if not label in n_labels:
        n_labels.append(label)

classes_perf_per_test.labels = n_labels
```

```
    print(n_labels)




    return render_template('single_class.html', title=xclass.class_name, xclass=xclass, class_perf=classes_perf_per_test)


# Allows the user to update the details of a class, it returns a form with all current values filled in.
# If the user is trying to update the details of a class that they do not teach then a 403 FORBIDDEN responce is issued.
@app.route("/class/<int:class_id>/update", methods=['GET', 'POST'])
@login_required
def update_class(class_id):
    xclass = Class.query.get_or_404(class_id)
    if xclass.course.teacher != current_user:
        abort(403)
    form = ClassForm()

    courses = Course.query.filter_by(teacher=current_user).all()
    form.course_id.choices = [(course.id, course.name) for num, course in enumerate(courses)]

    if form.validate_on_submit():
        xclass.class_name = form.class_name.data
        xclass.class_starting_date = form.class_starting_date.data
        xclass.course_id = form.course_id.data
        db.session.commit()
        flash('Your class has been updated!', 'success')
        return redirect(url_for('xclass', class_id=xclass.id))
    elif request.method == 'GET':
        form.class_name.data = xclass.class_name
        form.class_starting_date.data = xclass.class_starting_date
        form.course_id.data = xclass.course.id
        form.course_id.enabled = True
    return render_template('new_class.html', title='Update Class',
                           form=form, legend='Update Class')


# An endpoint that only takes a POST request, if the user is the teacher of the class then it will be deleted on request.
@app.route("/class/<int:class_id>/delete", methods=['POST'])
@login_required
def delete_class(class_id):
    xclass = Class.query.get_or_404(class_id)
    if xclass.course.teacher != current_user:
        abort(403)
    db.session.delete(xclass)
    db.session.commit()
    flash('Your class has been deleted!', 'success')
    return redirect(url_for('classes'))


@app.route("/class/<int:class_id>/report")
@login_required
def class_report(class_id):
    xclass = Class.query.get_or_404(class_id)
    if xclass.course.teacher != current_user:
        abort(403)

    pdf = FPDF('P', 'mm', 'A4')
    pdf.add_page()
    pdf.set_font("Arial", size=16)

    top = pdf.y
    pdf.multi_cell(158, 10, txt="Student", align="C")
    pdf.y = top
```

```
pdf.set_text_color(200, 0, 0)
pdf.multi_cell(193, 10, txt="Track ", align="C")
pdf.y = top
pdf.set_text_color(0, 0, 0)
pdf.multi_cell(225, 10, txt="Report", align="C")


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Class Info")
pdf.set_font("Arial", size=10)
pdf.ln(3)
pdf.multi_cell(0, 5, ('Class Name: %s' % xclass.class_name))
pdf.ln(1)
pdf.multi_cell(0, 5, ('Assigned To: %s' % xclass.course.name))
pdf.ln()


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Students")
pdf.set_font("Arial", size=10)
pdf.ln(3)

for student in xclass.students:
    pdf.multi_cell(0, 5, "- " + student.name)
    pdf.ln(1)
pdf.ln()


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Class Avg Performance")

pdf.set_font("Arial", size=10)
pdf.ln(3)

temp_data = []
labels = []

data = []

for topic in xclass.course.topics:

    for hmwk in topic.homeworks:
        top = pdf.y
        pdf.multi_cell(0, 5, topic.name +" (Hmwk) "+hmwk.name)

        # Get avg perf
        mrks = []
        for student in xclass.students:
            try:
                mrks.append(HomeworkMark.query.filter_by(homework_id=hmwk.id).filter_by(student_id=student.id).all()[0].mark)
            except IndexError:
                pass

        c_sum = 0
        c_len = 0
        for mrk in mrks:
            try:
                c_sum += mrk
                c_len += 1
            except TypeError:
                pass
```

```
            pdf.y = top
            pdf.x = 100

            if c_len:
                avg = c_sum // c_len
                pdf.multi_cell(0, 5, str(avg) + "/" + str(hmwk.max_mark))
            else:
                pdf.multi_cell(0, 5, "No Mark Awarded")

            mrks = []


        for test in topic.tests:
            top = pdf.y
            pdf.multi_cell(0, 5, topic.name+" (Test) "+test.name)
            # labels.append("TEST "+topic.name+" "+test.name)

            # Get avg perf
            mrks = []
            for student in xclass.students:
                try:
                    mrks.append(TestMark.query.filter_by(test_id=test.id).filter_by(student_id=student.id).all()[0].mark)
                except IndexError:
                    pass

            c_sum = 0
            c_len = 0
            for mrk in mrks:
                try:
                    c_sum += mrk
                    c_len += 1
                except TypeError:
                    pass

            pdf.y = top
            pdf.x = 100

            if c_len:
                avg = c_sum // c_len
                pdf.multi_cell(0, 5, str(avg) + "/" + str(test.max_mark))
            else:
                pdf.multi_cell(0, 5, "No Mark Awarded")

            mrks = []


    pdf.output(app.config['PDF_FILE_DUMP'] + "student_report.pdf")

    return send_file("static/pdf_gen/student_report.pdf", attachment_filename='student_report.pdf')


# Route for adding a new student. If POST method is used then user input is validated and
# student is added to database if all validation rules are passed. All the current users classes are queried and
# passed into the form so that the teacher can select a class to add the student to.
@app.route("/new_student", methods=['GET', 'POST'])
@login_required
def new_student():
    form = StudentForm()
    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses))

    if not classes.first():
        flash(Markup('To add a student you must first <a href="' + url_for('new_class') + '">add a class</a>.'), 'danger')

    form.class_id.choices = [(xclass.id, xclass.class_name) for xclass in classes]
    if form.validate_on_submit():
```

```
        new_student = Student(name=form.name.data,
                              email=form.email.data,
                              address=form.address.data,
                              parent_phone=form.parent_phone.data,
                              predicted_grade=form.predicted_grade.data)
        join_class = Class.query.filter_by(id=form.class_id.data).first_or_404()
        new_student.classes.append(join_class)
        db.session.add(new_student)
        db.session.commit()
        flash('New student has been added!', 'success')
        return redirect(url_for('students'))

    return render_template('new_student.html', title='Add Student', form=form, legend="Add Student")


# Route to allow the user to view all of their students, the students are queried from the database and the "teacher" parameter is used
# to ensure only students that are taught by the current user is returned.
@app.route("/students", methods=['GET', 'POST'])
@login_required
def students():
    page = request.args.get('page', 1, type=int)

    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses))

    students = set()
    for xclass in classes:
        students.update(xclass.students)

    s_form = SearchForm()
    srch = 0
    search_results = []
    if s_form.validate_on_submit():
        print("Search For:", s_form.search_query.data)

        student_ids = [student.id for student in students]

        search_results = Student.query.filter(Student.name.like('%' + s_form.search_query.data + '%'))
        search_results = search_results.filter(Student.id.in_(student_ids))
        srch = 1

    return render_template('all_students.html',
                           title="My Students",
                           students=students,
                           user=current_user,
                           s_form=s_form,
                           search_results=search_results,
                           srch=srch)


# Allows the user to view all the details of a specific student who's id will be passed into the url. This is mostly done
# automatically by a button on the interface. If the user is trying to access the details of a student that they do not
# teach then a 403 FORBIDDEN responce is issued.
@app.route("/student/<int:student_id>")
@login_required
def student(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)


    classes_perf_per_test = Graph("Avg Performance Per Test", "line")


    courses = db.session.query(Course.id).filter_by(teacher=current_user)
```

```
    topics = Topic.query.filter(Topic.course_id.in_(courses)).all()
    classes = Class.query.filter(Class.course_id.in_(courses)).all()

    temp_data = []
    labels = []

    classes_perf_per_test_data = {"label":student.name, "data":[]}

    for topic in topics:

        for hmwk in topic.homeworks:
            print(hmwk.name)
            labels.append("HMWK "+topic.name+" "+hmwk.name)

            try:
                mark = HomeworkMark.query.filter_by(homework_id=hmwk.id).filter_by(student_id=student.id).all()[0].mark

                try:
                    int(mark)
                    classes_perf_per_test_data["data"].append(mark)
                except ValueError:
                    classes_perf_per_test_data["data"].append(0)

            except IndexError:
                pass



        for test in topic.tests:
            print(test.name)
            labels.append("TEST "+topic.name+" "+test.name)

            try:
                mark = TestMark.query.filter_by(test_id=test.id).filter_by(student_id=student.id).all()[0].mark

                try:
                    int(mark)
                    classes_perf_per_test_data["data"].append(mark)
                except ValueError:
                    classes_perf_per_test_data["data"].append(0)
            except IndexError:
                pass



    print("\n\n", classes_perf_per_test_data["data"])


    classes_perf_per_test.datasets.append(classes_perf_per_test_data)

    # Remove duplicates...
    n_labels = []
    for label in labels:
        if not label in n_labels:
            n_labels.append(label)

    classes_perf_per_test.labels = n_labels

    print(n_labels)


    return render_template('single_student.html', title=student.name, student=student, class_perf=classes_perf_per_test)


# Allows the user to update the details of a student, it returns a form with all current values filled in.
# If the user is trying to update the details of a student that they do not teach then a 403 FORBIDDEN responce is issued.
```

```python
@app.route("/student/<int:student_id>/update", methods=['GET', 'POST'])
@login_required
def update_student(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)
    form = StudentForm()

    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses))
    form.class_id.choices = [(xclass.id, xclass.class_name) for xclass in classes]
    if form.validate_on_submit():
        join_class = Class.query.filter_by(id=form.class_id.data).first_or_404()
        student.classes[0] = join_class # Only the first class
        student.name = form.name.data
        student.email = form.email.data
        student.address = form.address.data
        student.parent_phone = form.parent_phone.data
        student.predicted_grade = form.predicted_grade.data
        db.session.commit()
        flash('Your class has been updated!', 'success')
        return redirect(url_for('student', student_id=student.id))
    elif request.method == 'GET':
        form.class_id.data = student.classes[0].id # Only the first class
        form.name.data = student.name
        form.email.data = student.email
        form.address.data = student.address
        form.parent_phone.data = student.parent_phone
        form.predicted_grade.data = student.predicted_grade
    return render_template('new_student.html', title='Update Student',
                           form=form, legend='Update Student')


# An endpoint that only takes a POST request, if the user is the teacher of the student then it will be deleted on request.
@app.route("/student/<int:student_id>/delete", methods=['POST'])
@login_required
def delete_student(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)
    db.session.delete(student)
    db.session.commit()
    flash('Your student has been deleted!', 'success')
    return redirect(url_for('students'))


# This endpoint takes the id of a student and allows the user to add them to a class. This is usually used from the student
# page from a button called add to class.
@app.route("/student/<int:student_id>/add_to_class", methods=['GET', 'POST'])
@login_required
def add_to_class(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)
    form = AddStudentToClass()
    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses))
    form.class_id.choices = [(xclass.id, xclass.class_name) for xclass in classes]
    # form.class_id.choices = [(xclass.id, xclass.class_name) for xclass in Class.query.filter_by(teacher=current_user)]
    if form.validate_on_submit():
        join_class = Class.query.filter_by(id=form.class_id.data).first_or_404()
        student.classes.append(join_class)
        db.session.commit()
        flash('Your class has been updated!', 'success')
        return redirect(url_for('student', student_id=student.id))
    elif request.method == 'GET':
```

```
        current_classes = student.classes
        for j in range(len(form.class_id.choices)):
            for i, xclass in enumerate(form.class_id.choices):
                for curclass in current_classes:
                    if xclass[0] == curclass.id:
                        del form.class_id.choices[i]

    return render_template('add_student_to_class.html', title='Add Student To Class',
                           form=form, legend='Add Student To Class')


# This endpoint takes the id of a student and allows the user to remove them from a class. This is usually used from the student
# page from a button called remove from class.
@app.route("/student/<int:student_id>/remove_from_class", methods=['GET', 'POST'])
@login_required
def remove_from_class(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)
    form = RemoveStudentFromClass()
    courses = db.session.query(Course.id).filter_by(teacher=current_user)
    classes = Class.query.filter(Class.course_id.in_(courses))
    form.class_id.choices = [(xclass.id, xclass.class_name) for xclass in classes if xclass in student.classes]
    if form.validate_on_submit():
        remove_class = Class.query.filter_by(id=form.class_id.data).first_or_404()

        if not (len(student.classes) > 1):
            flash('You cannot delete the only class a student is assigned to.', 'danger')
            return redirect(url_for('student', student_id=student.id))

        for i, xclass in enumerate(student.classes):
            if xclass.id == remove_class.id:
                del student.classes[i]

        db.session.commit()
        flash('Your class has been updated!', 'success')
        return redirect(url_for('student', student_id=student.id))
    elif request.method == 'GET':
        pass

    return render_template('add_student_to_class.html', title='Remove Student From Class',
                           form=form, legend='Remove Student From Class')



from flask import send_file
from fpdf import FPDF

@app.route("/student/<int:student_id>/report")
@login_required
def student_report(student_id):
    student = Student.query.get_or_404(student_id)
    if student.classes[0].course.teacher != current_user:
        abort(403)

    pdf = FPDF('P', 'mm', 'A4')
    pdf.add_page()
    pdf.set_font("Arial", size=16)

    top = pdf.y
    pdf.multi_cell(158, 10, txt="Student", align="C")
    pdf.y = top
    pdf.set_text_color(200, 0, 0)
    pdf.multi_cell(193, 10, txt="Track ", align="C")
    pdf.y = top
    pdf.set_text_color(0, 0, 0)
```

```
pdf.multi_cell(225, 10, txt="Report", align="C")


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Student Info")
pdf.set_font("Arial", size=10)
pdf.ln(3)
pdf.multi_cell(0, 5, ('Name: %s' % student.name))
pdf.ln(1)
pdf.multi_cell(0, 5, ('Email: %s' % student.email))
pdf.ln(1)
pdf.multi_cell(0, 5, ('Address: %s' % student.address))
pdf.ln(1)
pdf.multi_cell(0, 5, ('Parent Phone Number: %s' % student.parent_phone))
pdf.ln(1)
pdf.multi_cell(0, 5, ('Predicted Grade: %s' % student.predicted_grade))
pdf.ln()


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Students Classes")
pdf.set_font("Arial", size=10)
pdf.ln(3)

for xclass in student.classes:
    pdf.multi_cell(0, 5, "- " + xclass.class_name)
    pdf.ln(1)
pdf.ln()


pdf.set_font("Arial", "B", size=12)
pdf.ln()
pdf.multi_cell(0, 5, "Student Performance")

pdf.set_font("Arial", "B", size=10)
pdf.ln()
pdf.multi_cell(0, 1, "Homework")
pdf.set_font("Arial", size=10)
pdf.ln(3)

# FOR HOMEWORKS
marks = HomeworkMark.query.filter_by(student_id=student.id).all()

for mark in marks:
    hmwk = mark.homework

    top = pdf.y
    pdf.multi_cell(0, 5, "("+hmwk.topic.name +") "+hmwk.name)
    pdf.y = top
    pdf.x = 95

    try:
        int(mark.mark)
        if mark.mark != "":
            pdf.multi_cell(0, 5, str(mark.mark) + "/" + str(hmwk.max_mark))
        else:
            pdf.multi_cell(0, 5, "No Mark Awarded")
    except ValueError:
        pdf.multi_cell(0, 5, "No Mark Awarded")
    pdf.ln(1)


pdf.set_font("Arial", "B", size=10)
pdf.ln()
```

```
    pdf.multi_cell(0, 1, "Tests")
    pdf.set_font("Arial", size=10)
    pdf.ln(3)

    # FOR TESTS
    marks = TestMark.query.filter_by(student_id=student.id).all()

    for mark in marks:
        test = mark.test

        top = pdf.y
        pdf.multi_cell(0, 5, "("+test.topic.name +") "+test.name)
        pdf.y = top
        pdf.x = 95

        try:
            int(mark.mark)
            if mark.mark != "":
                pdf.multi_cell(0, 5, str(mark.mark) + "/" + str(test.max_mark))
            else:
                pdf.multi_cell(0, 5, "No Mark Awarded")
        except ValueError:
            pdf.multi_cell(0, 5, "No Mark Awarded")
        pdf.ln(1)

    pdf.ln()

    pdf.output(app.config['PDF_FILE_DUMP'] + "student_report.pdf")

    return send_file("static/pdf_gen/student_report.pdf", attachment_filename='student_report.pdf')



# HOMEWORKS

@app.route("/new_homework/<int:topic_id>", methods=['GET', 'POST'])
@login_required
def new_homework(topic_id):
    form = HomeworkForm()
    topic = Topic.query.first_or_404(topic_id)
    if topic.course.teacher != current_user:
        abort(403)
    if form.validate_on_submit():

        topic = Topic.query.first_or_404(topic_id)

        if form.due_date.data > topic.start_date.date() and form.due_date.data < topic.end_date.date():
            new_homework = Homework(name=form.name.data,
                                    due_date=form.due_date.data,
                                    max_mark=form.max_mark.data,
                                    topic_id=topic_id)

            db.session.add(new_homework)
            db.session.commit()
            flash('New homework has been added!', 'success')
            return redirect(url_for('topics'))

        else:
            flash("Homework must be due between "+topic.start_date.strftime("%d %b %Y")+" to "+topic.end_date.strftime("%d %b
%Y") + " for it to be a part of the topic '"+topic.name+"'.", "danger")


    return render_template('new_homework.html', title='New Homework', form=form, legend="New Homework")


@app.route("/homework/<int:homework_id>")
```

```
@login_required
def homework(homework_id):
    homework = Homework.query.get_or_404(homework_id)
    if homework.topic.course.teacher != current_user:
        abort(403)

    cur_top_classes = homework.topic.course.classes

    class_perf = Graph("Avg Class Performance On " + homework.name, "bar")
    class_perf_data = {"label":"Avg Class Performance On " + homework.name, "data":[]}

    for xclass in cur_top_classes:
        class_perf.labels.append(xclass.class_name)

    cls_avgs = []
    mrks = []
    for xclass in cur_top_classes:
        for student in xclass.students:
            try:
                mark = HomeworkMark.query.filter_by(homework_id=homework.id).filter_by(student_id=student.id).all()[0].mark
            except IndexError:
                mark = 0
            try:
                mrks.append(int(mark))
            except:
                mrks.append(0)

        try:
            avg = sum(mrks) / len(mrks)
        except ZeroDivisionError:
            avg = 0
        class_perf_data["data"].append(avg)
        mrks = []


    if sum(class_perf_data["data"]) != 0:
        class_perf.datasets.append(class_perf_data)

    return render_template('single_homework.html', title=homework.name, homework=homework, class_perf=class_perf)


@app.route("/homework/<int:homework_id>/update", methods=['GET', 'POST'])
@login_required
def update_homework(homework_id):
    homework = Homework.query.get_or_404(homework_id)
    if homework.topic.course.teacher != current_user:
        abort(403)
    form = HomeworkForm()

    courses = Course.query.filter_by(teacher=current_user).all()

    if form.validate_on_submit():

        if form.due_date.data > homework.topic.start_date.date() and form.due_date.data < homework.topic.end_date.date():
            homework.name = form.name.data
            homework.due_date = form.due_date.data
            homework.max_mark = form.max_mark.data
            db.session.commit()
            flash('Your homework has been updated!', 'success')
            return redirect(url_for('topics'))
        else:
            flash("Homework must be due between "+homework.topic.start_date.strftime("%d %b %Y")+" to
"+homework.topic.end_date.strftime("%d %b %Y") + " for it to be a part of the topic '"+homework.topic.name+"'.", "danger")


    elif request.method == 'GET':
```

```
        form.name.data = homework.name
        form.due_date.data = homework.due_date
        form.max_mark.data = homework.max_mark
    return render_template('new_homework.html', title='Update Homework',
                           form=form, legend='Update Homework')


@app.route("/homework/<int:homework_id>/delete", methods=['POST'])
@login_required
def delete_homework(homework_id):
    homework = Homework.query.get_or_404(homework_id)
    if homework.topic.course.teacher != current_user:
        abort(403)

    for mark in homework.homework_marks:
        db.session.delete(mark)

    db.session.delete(homework)
    db.session.commit()
    flash('Your homework has been deleted!', 'success')
    return redirect(url_for('topics'))

from flask_wtf import FlaskForm
from wtforms import FieldList, StringField

@app.route("/homework/<int:homework_id>/mark", methods=['GET', 'POST'])
@login_required
def mark_homework(homework_id):

    homework = Homework.query.get_or_404(homework_id)
    classes = homework.topic.course.classes

    student_number = 0
    for xclass in classes:
        student_number += len(xclass.students)

    class LocalForm(HomeworkMarkForm):pass
    LocalForm.marks = FieldList(StringField('Mark'), min_entries=student_number)
    form = LocalForm()

    mark_fields = {}
    c = 0
    for xclass in classes:
        mark_fields[xclass.id] = []
        students = xclass.students

        print(xclass.class_name)
        for student in xclass.students:
            print(student.name)
            form.marks[c].label.text = student.name + "'s Mark"
            mark_fields[xclass.id].append([student, form.marks[c]])
            c += 1

    if form.validate_on_submit():
        students = []
        for xclass in classes:
            students = students + xclass.students

        marks = []
        for i, mark in enumerate(form.marks.data):

            if mark:
                marks.append((students[i], mark))

        print(marks)
```

```
        for mark in marks:

            try:

                if int(mark[1]) > homework.max_mark:
                    flash("The maximum mark for this homework is "+str(homework.max_mark)+". All marks must be bellow this.",
"danger")
                    return render_template('mark_homework.html', title='Mark Homework',
                                           form=form, classes=classes, mark_fields=mark_fields, legend='Mark Homework')

                homework_mark = HomeworkMark(mark=mark[1],
                                             homework_id=homework_id,
                                             student_id=mark[0].id)

                print(homework_mark)
                db.session.add(homework_mark)

            except ValueError:
                flash("All marks must be ints!", "danger")
                return render_template('mark_homework.html', title='Mark Homework',
                                       form=form, classes=classes, mark_fields=mark_fields, legend='Mark Homework')

        db.session.commit()
        flash('Homework has been marked!', 'success')
        return redirect(url_for('dash'))

    else:
        print(form.errors)

    if homework.topic.course.teacher != current_user:
        abort(403)

    return render_template('mark_homework.html', title='Mark Homework',
                           form=form, classes=classes, mark_fields=mark_fields, legend='Mark Homework')


# TESTS

@app.route("/new_test/<int:topic_id>", methods=['GET', 'POST'])
@login_required
def new_test(topic_id):
    form = TestForm()
    topic = Topic.query.first_or_404(topic_id)
    if topic.course.teacher != current_user:
        abort(403)

    if form.validate_on_submit():


        if form.date.data > topic.start_date.date() and form.date.data < topic.end_date.date():
            new_test = Test(name=form.name.data,
                            date=form.date.data,
                            max_mark=form.max_mark.data,
                            topic_id=topic_id)

            db.session.add(new_test)
            db.session.commit()
            flash('New test has been added!', 'success')
            return redirect(url_for('topics'))

        else:
            flash("Test must be due between "+topic.start_date.strftime("%d %b %Y")+" to "+topic.end_date.strftime("%d %b %Y") +
" for it to be a part of the topic '"+topic.name+"'.", "danger")


    return render_template('new_test.html', title='New Test', form=form, legend="New Test")
```

```
@app.route("/test/<int:test_id>")
@login_required
def test(test_id):
    test = Test.query.get_or_404(test_id)
    if test.topic.course.teacher != current_user:
        abort(403)


    cur_top_classes = test.topic.course.classes

    class_perf = Graph("Avg Class Performance On " + test.name, "bar")
    class_perf_data = {"label":"Avg Class Performance On " + test.name, "data":[]}

    for xclass in cur_top_classes:
        class_perf.labels.append(xclass.class_name)

    cls_avgs = []
    mrks = []
    for xclass in cur_top_classes:
        for student in xclass.students:
            try:
                mark = TestMark.query.filter_by(homework_id=homework.id).filter_by(student_id=student.id).all()[0].mark
            except IndexError:
                mark = 0
            try:
                mrks.append(int(mark))
            except:
                mrks.append(0)

        try:
            avg = sum(mrks) / len(mrks)
        except ZeroDivisionError:
            avg = 0
        class_perf_data["data"].append(avg)
        mrks = []

    if sum(class_perf_data["data"]) != 0:
        class_perf.datasets.append(class_perf_data)

    return render_template('single_test.html', title=test.name, test=test, class_perf=class_perf)


@app.route("/test/<int:test_id>/update", methods=['GET', 'POST'])
@login_required
def update_test(test_id):
    test = Test.query.get_or_404(test_id)
    if test.topic.course.teacher != current_user:
        abort(403)
    form = TestForm()

    if form.validate_on_submit():

        if form.date.data > test.topic.start_date.date() and form.date.data < test.topic.end_date.date():
            test.name = form.name.data
            test.max_mark = form.max_mark.data
            test.date = form.date.data
            db.session.commit()
            flash('Your test has been updated!', 'success')
            return redirect(url_for('topics'))

        else:
            flash("Test must be due between "+test.topic.start_date.strftime("%d %b %Y")+" to "+test.topic.end_date.strftime("%d
%b %Y") + " for it to be a part of the topic '"+test.topic.name+"'.", "danger")
```

```
    elif request.method == 'GET':
        form.name.data = test.name
        form.max_mark.data = test.max_mark
        form.date.data = test.date

    return render_template('new_test.html', title='Update Test',
                           form=form, legend='Update Test')


@app.route("/test/<int:test_id>/delete", methods=['POST'])
@login_required
def delete_test(test_id):
    test = Test.query.get_or_404(test_id)
    if test.topic.course.teacher != current_user:
        abort(403)

    marks = TestMark.query.filter_by(test_id=test.id)

    for mark in marks:
        db.session.delete(mark)

    db.session.delete(test)
    db.session.commit()
    flash('Your test has been deleted!', 'success')
    return redirect(url_for('topics'))


@app.route("/test/<int:test_id>/mark", methods=['GET', 'POST'])
@login_required
def mark_test(test_id):

    test = Test.query.get_or_404(test_id)
    classes = test.topic.course.classes

    student_number = 0
    for xclass in classes:
        student_number += len(xclass.students)

    class LocalForm(TestMarkForm):pass
    LocalForm.marks = FieldList(StringField('Mark'), min_entries=student_number)
    form = LocalForm()

    mark_fields = {}
    c = 0
    for xclass in classes:
        mark_fields[xclass.id] = []
        students = xclass.students

        print(xclass.class_name)
        for student in xclass.students:
            print(student.name)
            form.marks[c].label.text = student.name + "'s Mark"
            mark_fields[xclass.id].append([student, form.marks[c]])
            c += 1

    if form.validate_on_submit():
        students = []
        for xclass in classes:
            students = students + xclass.students

        marks = []
        for i, mark in enumerate(form.marks.data):

            if mark:
                marks.append((students[i], mark))
```

```
        print(marks)

        for mark in marks:

            try:

                if int(mark[1]) > test.max_mark:
                    flash("The maximum mark for this test is "+str(test.max_mark)+". All marks must be bellow this.", "danger")
                    return render_template('mark_test.html', title='Mark Test',
                                           form=form, classes=classes, mark_fields=mark_fields, legend='Mark Test')

                test_mark = TestMark(mark=mark[1],
                                     test_id=test_id,
                                     student_id=mark[0].id)

                print(test_mark)
                db.session.add(test_mark)


            except ValueError:
                flash("All marks must be ints!", "danger")
                return render_template('mark_test.html', title='Mark Test',
                                       form=form, classes=classes, mark_fields=mark_fields, legend='Mark Test')

        db.session.commit()
        flash('Test has been marked!', 'success')
        return redirect(url_for('dash'))

    else:
        print(form.errors)

    if test.topic.course.teacher != current_user:
        abort(403)

    return render_template('mark_test.html', title='Mark Test',
                           form=form, classes=classes, mark_fields=mark_fields, legend='Mark Test')


# EXAMS

@app.route("/new_exam/<int:course_id>", methods=['GET', 'POST'])
@login_required
def new_exam(course_id):
    form = ExamForm()
    if form.validate_on_submit():
        new_exam = Exam(name=form.name.data,
                        date=form.date.data,
                        max_mark=form.max_mark.data,
                        course_id=course_id)

        db.session.add(new_exam)
        db.session.commit()
        flash('New exam has been added!', 'success')
        return redirect(url_for('courses'))
    return render_template('new_exam.html', title='New Exam', form=form, legend="New Exam")


@app.route("/exam/<int:exam_id>")
@login_required
def exam(exam_id):
    exam = Homework.query.get_or_404(exam_id)
    if exam.course.teacher != current_user:
        abort(403)
    return render_template('single_exam.html', title=exam.name, exam=exam)
```

```
@app.route("/exam/<int:exam_id>/update", methods=['GET', 'POST'])
@login_required
def update_exam(exam_id):
    exam = Exam.query.get_or_404(exam_id)
    if exam.course.teacher != current_user:
        abort(403)
    form = ExamForm()

    if form.validate_on_submit():
        exam.name = form.name.data
        exam.max_mark = form.max_mark.data
        exam.date = form.date.data
        db.session.commit()
        flash('Your exam has been updated!', 'success')
        return redirect(url_for('courses'))
    elif request.method == 'GET':
        form.name.data = exam.name
        form.max_mark.data = exam.max_mark
        form.date.data = exam.date
    return render_template('new_exam.html', title='Update Exam',
                           form=form, legend='Update Exam')


@app.route("/exam/<int:exam_id>/delete", methods=['POST'])
@login_required
def delete_exam(exam_id):
    exam = Exam.query.get_or_404(exam_id)
    if exam.course.teacher != current_user:
        abort(403)
    db.session.delete(exam)
    db.session.commit()
    flash('Your exam has been deleted!', 'success')
    return redirect(url_for('courses'))


@app.route("/exam/<int:exam_id>/mark", methods=['GET', 'POST'])
@login_required
def mark_exam(exam_id):

    exam = Exam.query.get_or_404(exam_id)
    classes = exam.course.classes

    student_number = 0
    for xclass in classes:
        student_number += len(xclass.students)

    class LocalForm(ExamMarkForm):pass
    LocalForm.marks = FieldList(StringField('Mark'), min_entries=student_number)
    form = LocalForm()

    mark_fields = {}
    c = 0
    for xclass in classes:
        mark_fields[xclass.id] = []
        students = xclass.students

        print(xclass.class_name)
        for student in xclass.students:
            print(student.name)
            form.marks[c].label.text = student.name + "'s Mark"
            mark_fields[xclass.id].append([student, form.marks[c]])
            c += 1

    if form.validate_on_submit():
        students = []
```

```
        for xclass in classes:
            students = students + xclass.students

        marks = []
        for i, mark in enumerate(form.marks.data):

            if mark:
                marks.append((students[i], mark))

        print(marks)

        for mark in marks:
            exam_mark = ExamMark(mark=mark[1],
                                 exam_id=exam_id,
                                 student_id=mark[0].id)

            print(exam_mark)
            db.session.add(exam_mark)

        db.session.commit()
        flash('Exam has been marked!', 'success')
        return redirect(url_for('dash'))

    else:
        print(form.errors)

    if exam.course.teacher != current_user:
        abort(403)

    return render_template('mark_exam.html', title='Mark Exam',
                           form=form, classes=classes, mark_fields=mark_fields, legend='Mark Exam')




# COURSES

@app.route("/new_course", methods=['GET', 'POST'])
@login_required
def new_course():
    form = CourseForm()
    form.grade_system.choices = [(0, "A*-U"),
                                 (1, "A-U"),
                                 (2, "A-E"),
                                 (3, "9-1"),
                                 (4, "A-F"),
                                 (5, "A+-F")]
    if form.validate_on_submit():
        new_course = Course(name=form.name.data,
                            start_date=form.start_date.data,
                            year_num=form.year_num.data,
                            grading_system="".join([str(form.grade_system.data), form.grade_system.choices[form.grade_system.data]
[1]]),
                            teacher=current_user)
        db.session.add(new_course)
        db.session.commit()
        flash('New course has been added!', 'success')
        return redirect(url_for('courses'))

    return render_template('new_course.html', title='New Course', form=form, legend="New Course")


@app.route("/courses")
@login_required
def courses():
    page = request.args.get('page', 1, type=int)
```

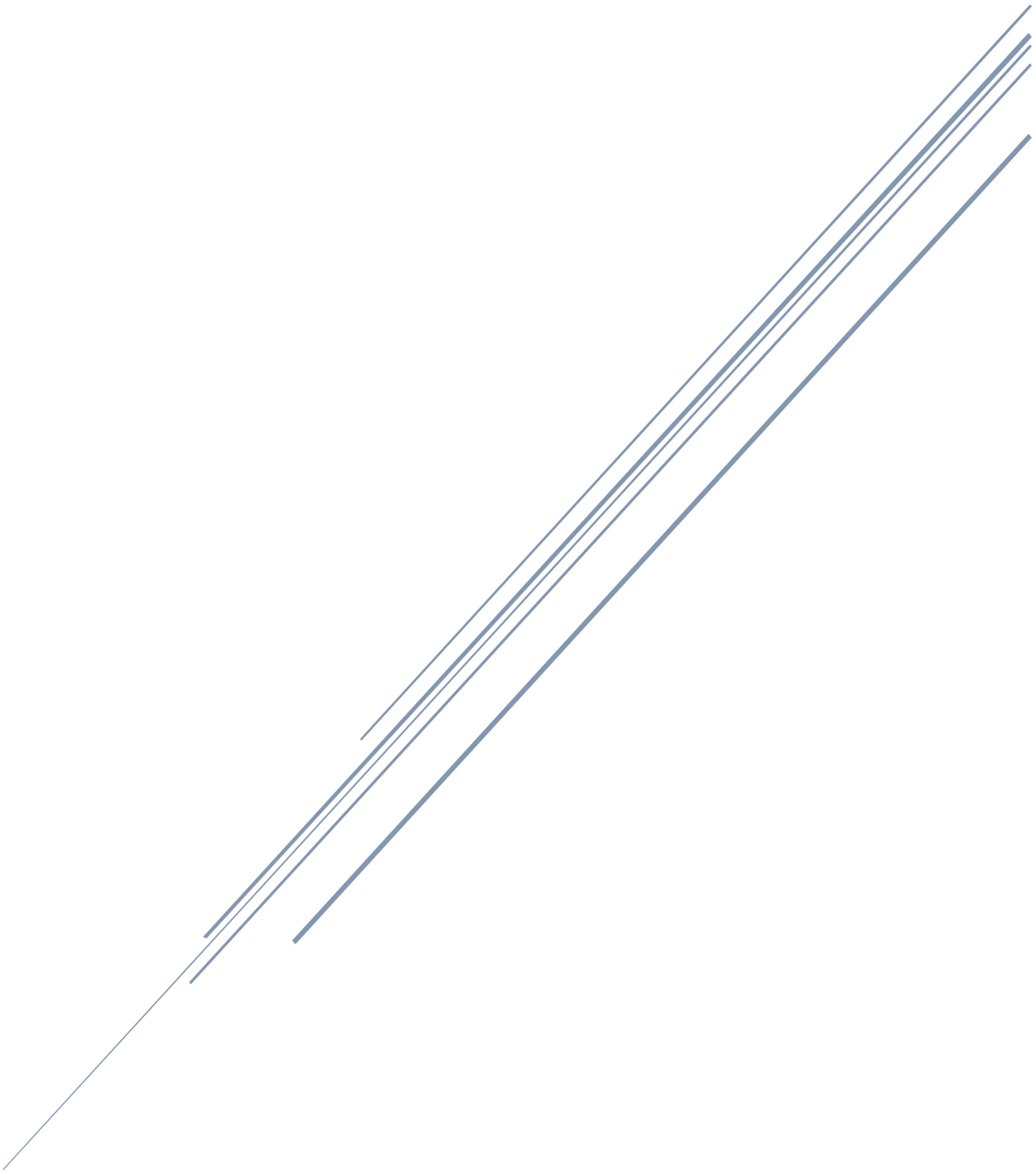
```
    courses = Course.query.filter_by(teacher=current_user)

    courses = courses.paginate(page=page, per_page=5)

    return render_template('all_courses.html', title="My Courses", courses=courses, user=current_user)


@app.route("/course/<int:course_id>/update", methods=['GET', 'POST'])
@login_required
def update_course(course_id):
    course = Course.query.get_or_404(course_id)
    if course.teacher != current_user:
        abort(403)
    form = CourseForm()
    form.grade_system.choices = [(0, "A*-U"),
                                 (1, "A-U"),
                                 (2, "A-E"),
                                 (3, "9-1"),
                                 (4, "A-F"),
                                 (5, "A+-F")]
    if form.validate_on_submit():
        course.name = form.name.data
        course.start_date = form.start_date.data
        course.year_num = form.year_num.data
        course.grading_system = "".join([str(form.grade_system.data), form.grade_system.choices[form.grade_system.data][1]])
        db.session.commit()
        flash('Your course has been updated!', 'success')
        return redirect(url_for('courses'))
    elif request.method == 'GET':
        form.name.data = course.name
        form.start_date.data = course.start_date
        form.year_num.data = course.year_num
        form.grade_system.data = int(course.grading_system[0])
    return render_template('new_course.html', title='Update Course',
                           form=form, legend='Update Course')


@app.route("/course/<int:course_id>/delete", methods=['POST'])
@login_required
def delete_course(course_id):
    course = Course.query.get_or_404(course_id)
    if course.teacher != current_user:
        abort(403)

    for topic in course.topics:
        db.session.delete(topic)

    for exam in course.exams:
        db.session.delete(exam)

    for xclass in course.classes:
        db.session.delete(xclass)

    db.session.delete(course)
    db.session.commit()
    flash('Your course has been deleted!', 'success')
    return redirect(url_for('courses'))

# TOPICS

@app.route("/new_topic", methods=['GET', 'POST'])
@login_required
def new_topic():
    form = TopicForm()

    courses = Course.query.filter_by(teacher=current_user).all()
```

```
    form.course_id.choices = [(course.id, course.name) for num, course in enumerate(courses)]

    if not len(courses):
        flash(Markup('To add a topic you must first <a href="' + url_for('new_course') + '">add a course</a>.'), 'danger')

    # If user doesn't request for a course, then enable the select box and set the result as the course_id...
    from datetime import date, timedelta

    try:
        course_id = request.args['course_id']
        form.course_id.value = course_id

        course = Course.query.get_or_404(course_id)
        minDate = course.start_date.strftime("%Y-%m-%d")

        all_block_dates = []
        for topic in course.topics:
            sdate = topic.start_date   # start date
            edate = topic.end_date   # end date
            delta = edate - sdate
            block_dates = [(sdate + timedelta(days=i)).strftime("%Y-%m-%d") for i in range(delta.days + 1)]
            all_block_dates = all_block_dates + block_dates

        all_block_dates = sorted(list(set(all_block_dates)))

        print(all_block_dates)


    except KeyError:
        form.course_id.enabled = True
        course_id = form.course_id.data

    if form.validate_on_submit():

        course = Course.query.get_or_404(course_id)
        if course.teacher != current_user:
            abort(403)


        if form.begin_date.data < form.end_date.data:

            conflicts = 0
            for topic in course.topics:
                if (form.begin_date.data < topic.start_date.date() and form.end_date.data < topic.start_date.date()) or
(form.begin_date.data > topic.end_date.date() and form.end_date.data > topic.end_date.date()):
                    pass
                else:
                    conflicts += 1
                    flash("Topic dates conflict with the active period of another topic! Change BOTH dates to before
"+topic.start_date.strftime("%d %b %Y")+" or after "+topic.end_date.strftime("%d %b %Y") + " to resolve the conflict.", "danger")

            if not conflicts:
                if form.begin_date.data < course.start_date.date():
                    flash("Topic cannot begin before the course it's a part of does!", "danger")
                else:

                    new_topic = Topic(name=form.name.data,
                                      start_date=form.begin_date.data,
                                      end_date=form.end_date.data,
                                     course=course)

                    db.session.add(new_topic)
                    db.session.commit()
                    flash(f'New topic has been added to {course.name}!', 'success')

                    next_page = request.args.get('next')
```

```
            return redirect(next_page) if next_page else redirect(url_for('topics'))
        else:
            flash("The topic cannot end before it begins!", "danger")

    return render_template('new_topic.html',
                           title='New Topic',
                           form=form,
                           legend="New Topic",
                           min_date=minDate,
                           blocked_dates=all_block_dates)


@app.route("/topics", methods=['GET', 'POST'])
@login_required
def topics():
    page = request.args.get('page', 1, type=int)
    courses = Course.query.filter_by(teacher=current_user)


    s_form = SearchForm()
    srch = 0
    search_results = []
    if s_form.validate_on_submit():
        print("Search For:", s_form.search_query.data)

        course_ids = [course.id for course in courses.all()]

        search_results = Topic.query.filter(Topic.name.like('%' + s_form.search_query.data + '%'))
        search_results = search_results.filter(Topic.course_id.in_(course_ids))
        srch = 1


    if not len(courses.all()):
        flash(Markup('To add a topic you must first <a href="' + url_for('new_course') + '">add a course</a>.'), 'danger')

    courses = courses.paginate(page=page, per_page=5)


    return render_template('all_topics.html',
                           title="My Topics",
                           courses=courses,
                           user=current_user,
                           today=datetime.now(),
                           s_form=s_form,
                           search_results=search_results,
                           srch=srch)


@app.route("/topic/<int:topic_id>")
@login_required
def topic(topic_id):
    topic = Topic.query.get_or_404(topic_id)
    if topic.course.teacher != current_user:
        abort(403)
    return render_template('single_topic.html', title=topic.name, topic=topic)


@app.route("/topic/<int:topic_id>/update", methods=['GET', 'POST'])
@login_required
def update_topic(topic_id):
    topic = Topic.query.get_or_404(topic_id)
    if topic.course.teacher != current_user:
        abort(403)
    form = TopicForm()
    form.course_id.enabled = True
```

```
    courses = Course.query.filter_by(teacher=current_user).all()
    form.course_id.choices = [(course.id, course.name) for num, course in enumerate(courses)]

    if form.validate_on_submit():
        course = Course.query.get_or_404(form.course_id.data)

        if form.begin_date.data < form.end_date.data:

            conflicts = 0
            for t in course.topics:
                if t.id == topic.id:
                    pass
                elif (form.begin_date.data < t.start_date.date() and form.end_date.data < t.start_date.date()) or
(form.begin_date.data > t.end_date.date() and form.end_date.data > t.end_date.date()):
                    pass
                else:
                    conflicts += 1
                    flash("Topic dates conflict with the active period of another topic! Change BOTH dates to before
"+t.start_date.strftime("%d %b %Y")+" or after "+t.end_date.strftime("%d %b %Y") + " to resolve the conflict.", "danger")

            if not conflicts:
                if form.begin_date.data < course.start_date.date():
                    flash("Topic cannot begin before the course it's a part of does!", "danger")
                else:
                    topic.name = form.name.data
                    topic.start_date = form.begin_date.data
                    topic.end_date = form.end_date.data
                    topic.course_id = form.course_id.data
                    db.session.commit()
                    flash('Your topic has been updated!', 'success')
                    return redirect(url_for('topics'))
        else:
            flash("The topic cannot end before it begins!", "danger")

    elif request.method == 'GET':
        form.name.data = topic.name
        print()
        form.begin_date.data = topic.start_date
        form.end_date.data = topic.end_date
        form.course_id.data = topic.course.id
    return render_template('new_topic.html', title='Update Topic',
                           form=form, legend='Update Topic')


@app.route("/topic/<int:topic_id>/delete", methods=['POST'])
@login_required
def delete_topic(topic_id):
    topic = Topic.query.get_or_404(topic_id)
    if topic.course.teacher != current_user:
        abort(403)

    for test in topic.tests:
        db.session.delete(test)

    for hmwk in topic.homeworks:
        db.session.delete(hmwk)

    db.session.delete(topic)
    db.session.commit()
    flash('Your topic has been deleted!', 'success')
    return redirect(url_for('courses'))
```

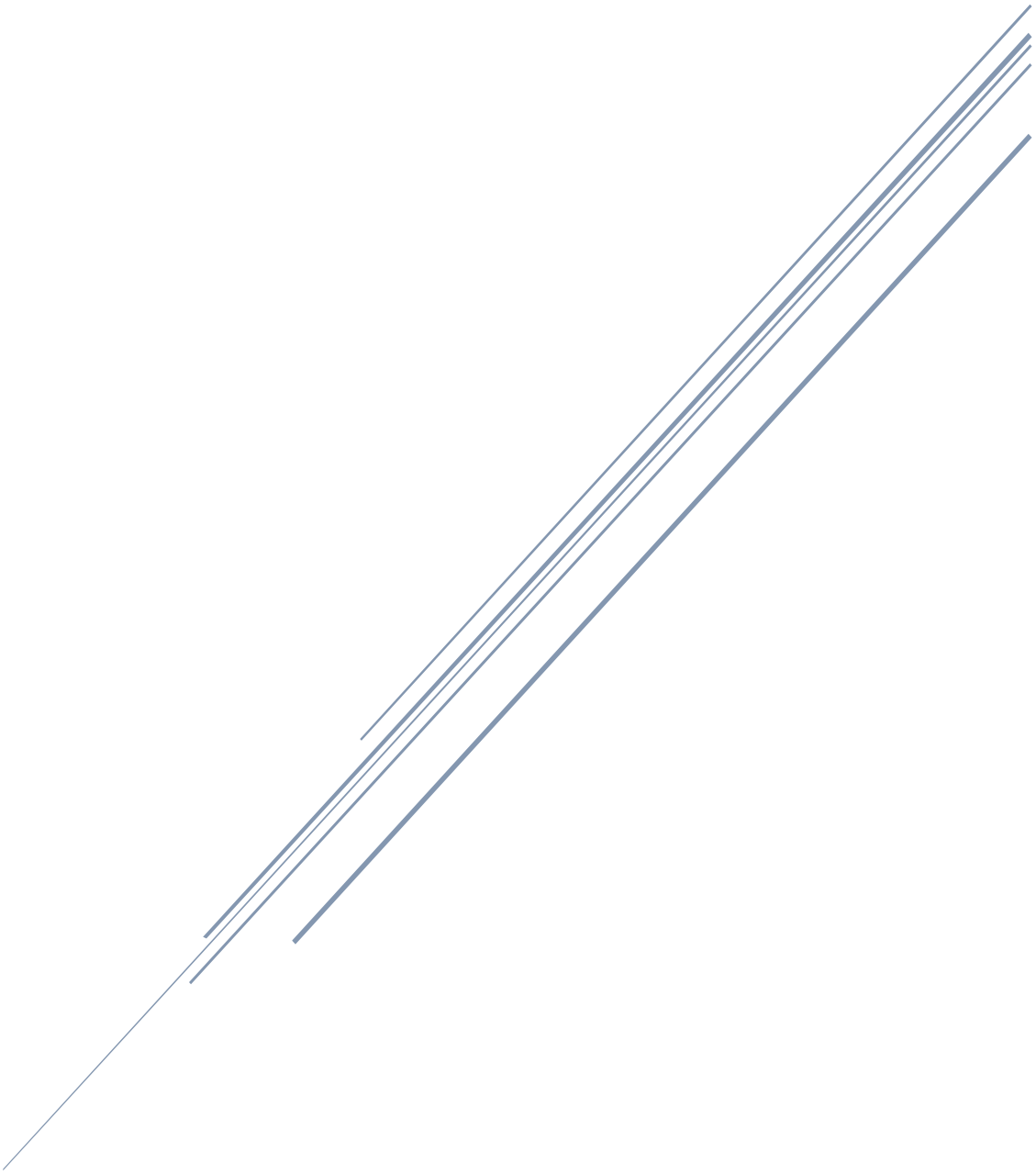
# dashboard_charts.js



```
//This is a module I built that is used on the front end to display all the graphs on the dashboard and student pages.

function random_rgb() {
  // Generates random colours...
  var o = Math.round, r = Math.random, s = 255;
  return 'rgb(' + o(r()*s) + ',' + o(r()*s) + ',' + o(r()*s) + ')';
}


function generateGraphColours(num) {
  // Initializing arrays...
  var borderColours = []
  var backgroundColours = []

  // Adding RGB and RGBA (translucent) values for each colour.
  for (i = 0; i < num; i++) {
    var col = random_rgb();
    borderColours.push(col);
    backgroundColours.push(col.replace(')', ', 0.25)').replace('rgb', 'rgba'));
  }

  // Returning values as an object, its syntactically nicer.
  return {
    borderColours: borderColours,
    backgroundColours: backgroundColours
  }
}


function buildChartDataset(data, type) {
  // Initializing "datasets" and generating graph colours.
  var datasets = []

  if (type == "bar") {
    // Generating dataset for all data needed in the graph.
    for (datum of data) {
      console.log(datum.data);
      var cols = generateGraphColours(7);
      var dataset = {
        "label": datum.label,
        "fill": false,
        "borderWidth": 1,
        "borderColor": cols.borderColours,
        "backgroundColor": cols.backgroundColours,
        "data": datum.data
      }
      datasets.push(dataset);
    }
    return datasets;
  } else if (type == "line") {
    console.log("LINE")

    var cols = generateGraphColours(7);
    var colours = cols.borderColours;

    // Generating dataset for all data needed in the graph.
    x = 0
    for (datum of data) {
      console.log(datum.data);
      var dataset = {
        "label": datum.label,
        "fill": false,
        "borderWidth": 1,
        "borderColor": colours[x],
        "backgroundColor": colours[x],
        "data": datum.data
```

```
      }
      datasets.push(dataset);
      x = x + 1
    }
    return datasets;
  }
}


function changeChart(elmtId, type, labels, data) {

  resetCanvas(elmtId);

  // Grabing the elements we need from the HTML.
  var canvas = document.getElementById(elmtId);
  var context = canvas.getContext('2d');

  // Building correctly stuctured datasets out of the data.
  var datasets = buildChartDataset(data, type);

  // Building the chart.
  var chart = new Chart(context,
    {
      "type": type,
      "data":{
        "labels": labels,
        "datasets": datasets
      },
      "options": {
        "scales": {
          "yAxes": [{"ticks": {"beginAtZero": true}}]
        }
      }
    });
}


function resetCanvas(elmtId) {
  elmtParent = $('#'+elmtId).parent().attr('id');
  $('#'+elmtId).remove();
  $('#'+elmtParent).append('<canvas id="'+elmtId+'"><canvas>');
};
```

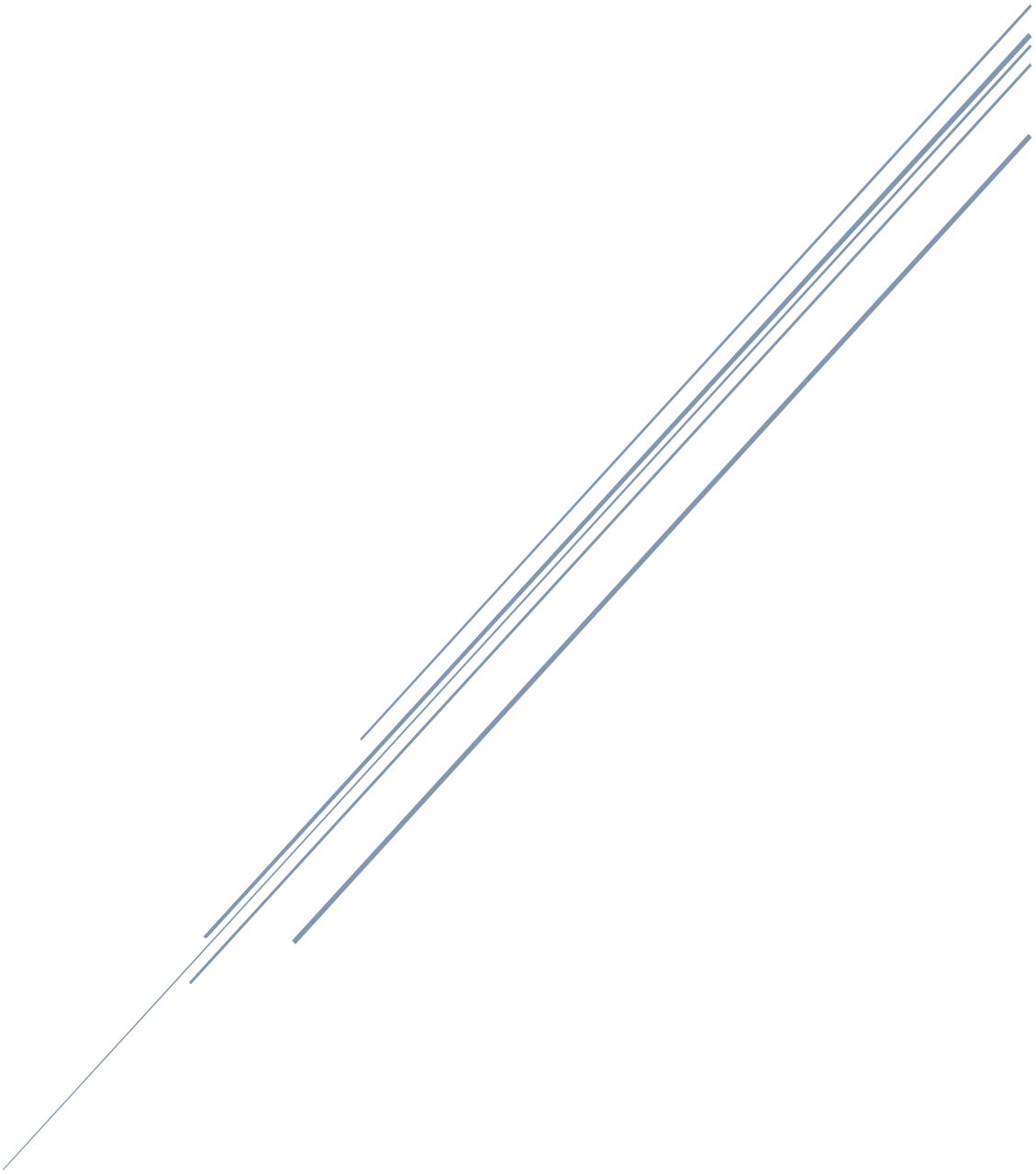
# main.css



```css
body {
  background: #fafafa;
  color: #333333;
  margin-top: 5rem;
}

h1, h2, h3, h4, h5, h6 {
  color: #444444;
}

.bg-steel {
  background-color: #2E4252;
}

.site-header .navbar-nav .nav-link {
  color: #cbd5db;
}

.site-header .navbar-nav .nav-link:hover {
  color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
  font-weight: 500;
}

.nav-img {
  height: 40px;
  width: 40px;
}

.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;

  /* Ensure that huge words dont break the styles */
  word-wrap:break-word;
  word-break:break-word;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}

.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
```

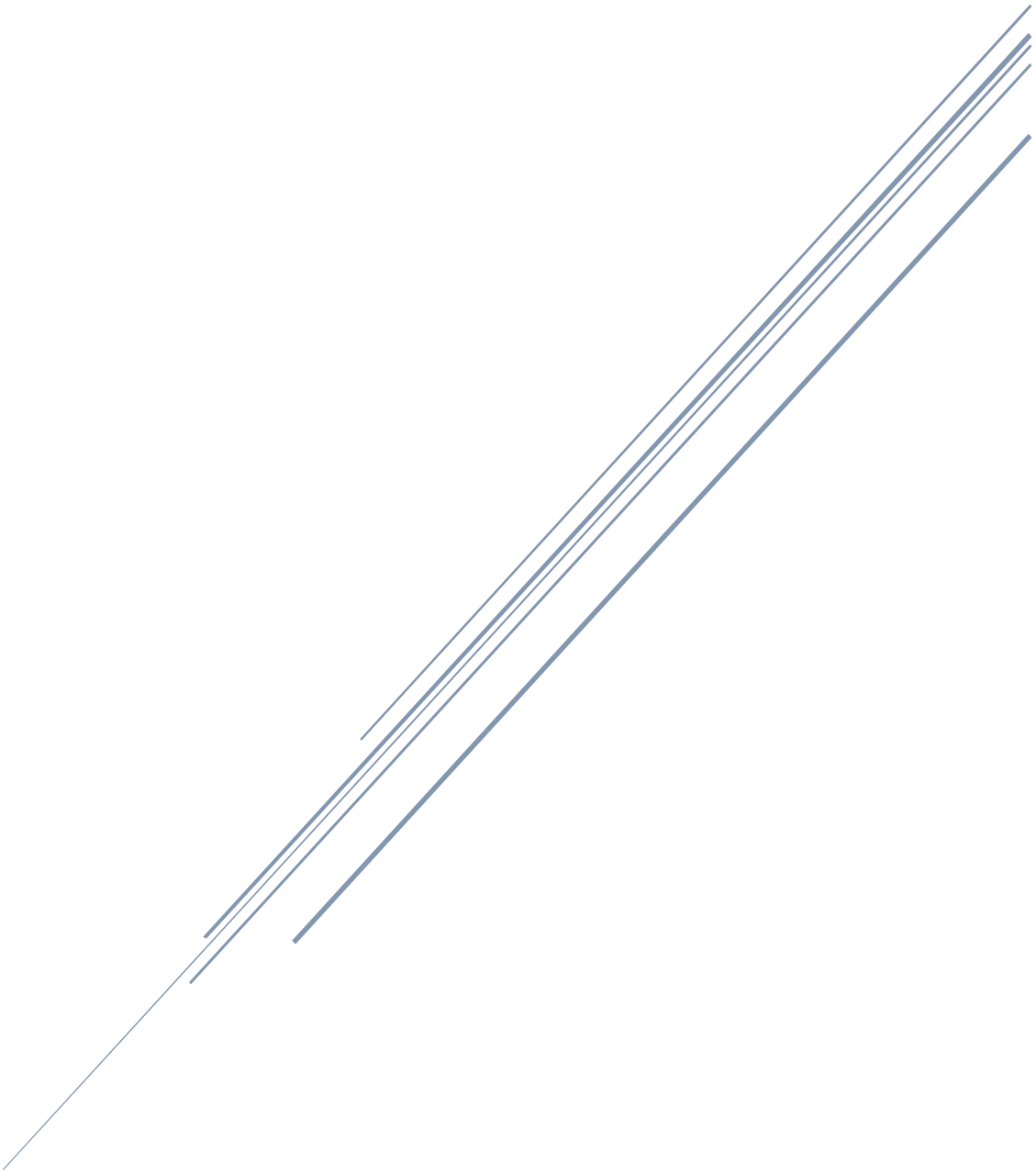
```
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}

.account-img {
  height: 125px;
  width: 125px;
  margin-right: 20px;
  margin-bottom: 16px;
}

.account-heading {
  font-size: 2.5rem;
}


/* Ribbon Stuff */
.parent {
  overflow: hidden;
  position: relative;
}
.ribbon {
  margin: 0;
  padding: 0;
  background: #17a2b8;
  color:white;
  padding:0.5em 0;
  position: absolute;
  top:0;
  right:0;
  transform: translateX(30%) translateY(0%) rotate(45deg);
  transform-origin: top left;
}
.ribbon:before,
.ribbon:after {
  content: '';
  position: absolute;
  top:0;
  margin: 0 -1px;
  width: 100%;
  height: 100%;
  background: #17a2b8;
}
.ribbon:before {
  right:100%;
}
.ribbon:after {
  left:100%;
}
```

# about.html



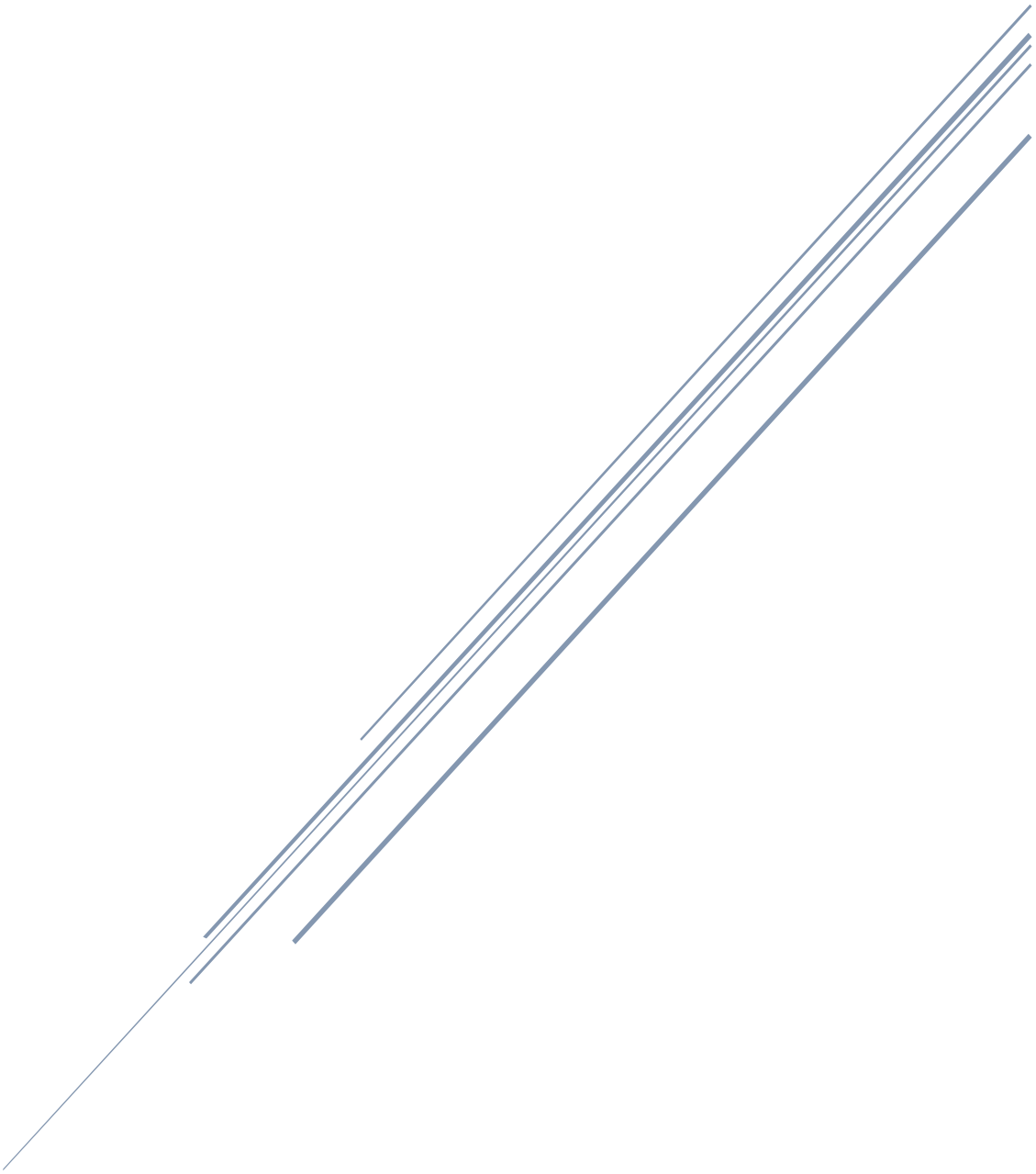
```
{% extends "layout_content.html" %}
{% block content %}
    <h1>About Student<font color="red"><i>Track</i></font></h1>
    <p>Student<font color="red"><i>Track</i></font> is a program designed to aid teachers in organising and managing their courses, students and classes. Please read the "Setup and Installation" section to get started.</p>

    <h3>Getting Started</h3>
    <p>To get started on Student<font color="red"><i>Track</i></font> it is recommended that you first go to the <a href="{{ url_for('courses') }}">Courses</a> page and add a course. After this you can add topics to this course, and add and assign classes to take this course. Once you have this initial setup done you can add some upcoming homeworks and tests so we can remind you when their scheduled time approaches. Student<font color="red"><i>Track</i></font> can also track the performance of classes over time, if you record your students' progress in Student<font color="red"><i>Track</i></font> you can view statistics on recent performance, monitor completion of homeworks and tests, and get general reminders on the <a href="{{ url_for('dash') }}">Dashboard</a>.</p>

    <h3>Coursework Mention</h3>
    <p>This website is a coursework project.</p>

    <h3>Setup and Installation</h3>
    <h5>For Testing and Assessment</h5>
    <ol>
    <li> Ensure python 3.6 or greater is installed.</li>
    <li> Open CMD on Windows, or a terminal on Linux.</li>
    <li> Navigate to the prototypes directory. (By default this document is contained in it.)</li>
    <li> If you wish to use a virtual enviroment run `python -m venv ./venv` and then `./venv/Scripts/activate.bat` on Windows, or `./venv/Scripts/activate` on Linux.</li>
    <li> Run `pip install -r requirements.txt` to install the requirements for this program.</li>
    <li> Run `python setup-database.py` to initialize the database.</li>
    <li> Run `python run.py` to run the server.</li>
    <li> Open a browser and navagate to `localhost:5000`.</li>
    </ol>

{% endblock content %}
```

# acc_login.html



```
{% extends "layout_content.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Log In</legend>
                <div class="form-group">
                    {{ form.email.label(class="form-control-label") }}
                    {% if form.email.errors %}
                        {{ form.email(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.email.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.email(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-group">
                    {{ form.password.label(class="form-control-label") }}
                    {% if form.password.errors %}
                        {{ form.password(class="form-control form-control-lg is-invalid") }}
                        <div class="invalid-feedback">
                            {% for error in form.password.errors %}
                                <span>{{ error }}</span>
                            {% endfor %}
                        </div>
                    {% else %}
                        {{ form.password(class="form-control form-control-lg") }}
                    {% endif %}
                </div>
                <div class="form-check">
                    {{ form.remember(class="form-check-input") }}
                    {{ form.remember.label(class="form-check-label") }}
                </div>
            </fieldset>
            <div class="form-group">
                {{ form.submit(class="btn btn-outline-info") }}
                <small class="text-muted ml-2">
                    <a href="{{ url_for('reset_request') }}">Forgot Password?</a>
                </small>
            </div>
        </form>
    </div>
    <div class="border-top pt-3">
        <small class="text-muted">
            Need An Account? <a class="ml-2" href="{{ url_for('register') }}">Sign Up Now</a>
        </small>
    </div>
{% endblock content %}
```

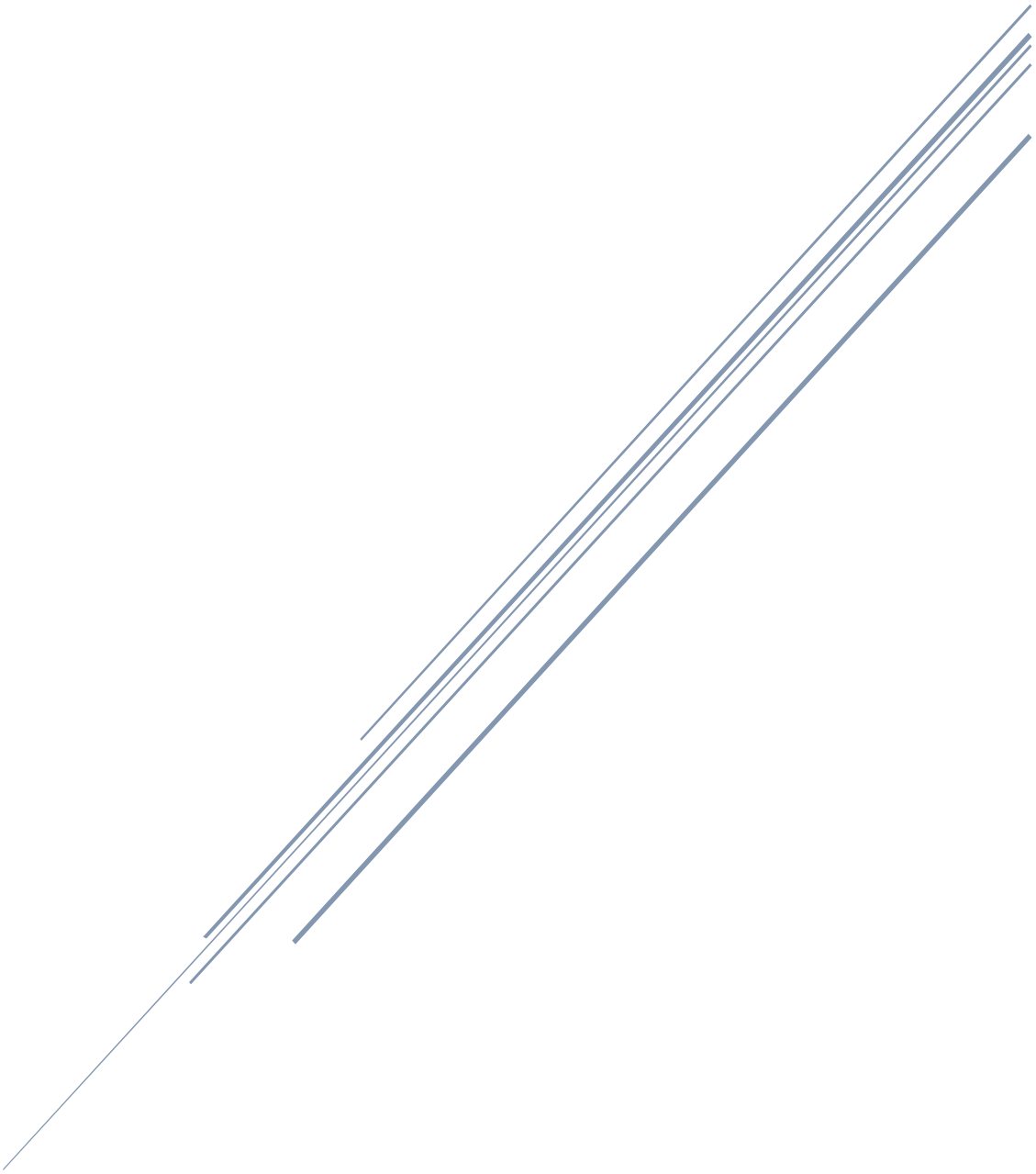
# acc_register.html



```
{% extends "layout_content.html" %}
{% block content %}
  <div class="content-section">
    <form method="POST" action="">
      {{ form.hidden_tag() }}
      <fieldset class="form-group">
        <legend class="border-bottom mb-4">Join Today</legend>
        <div class="form-group">
          {{ form.username.label(class="form-control-label") }}

          {% if form.username.errors %}
            {{ form.username(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
              {% for error in form.username.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.username(class="form-control form-control-lg") }}
          {% endif %}
        </div>
        <div class="form-group">
          {{ form.email.label(class="form-control-label") }}
          {% if form.email.errors %}
            {{ form.email(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
              {% for error in form.email.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.email(class="form-control form-control-lg") }}
          {% endif %}
        </div>
        <div class="form-group">
          {{ form.password.label(class="form-control-label") }}
          {% if form.password.errors %}
            {{ form.password(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
              {% for error in form.password.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.password(class="form-control form-control-lg") }}
          {% endif %}
        </div>
        <div class="form-group">
          {{ form.confirm_password.label(class="form-control-label") }}
          {% if form.confirm_password.errors %}
            {{ form.confirm_password(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
              {% for error in form.confirm_password.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.confirm_password(class="form-control form-control-lg") }}
          {% endif %}
        </div>
      </fieldset>
      <div class="form-group">
        {{ form.submit(class="btn btn-outline-info") }}
      </div>
    </form>
```

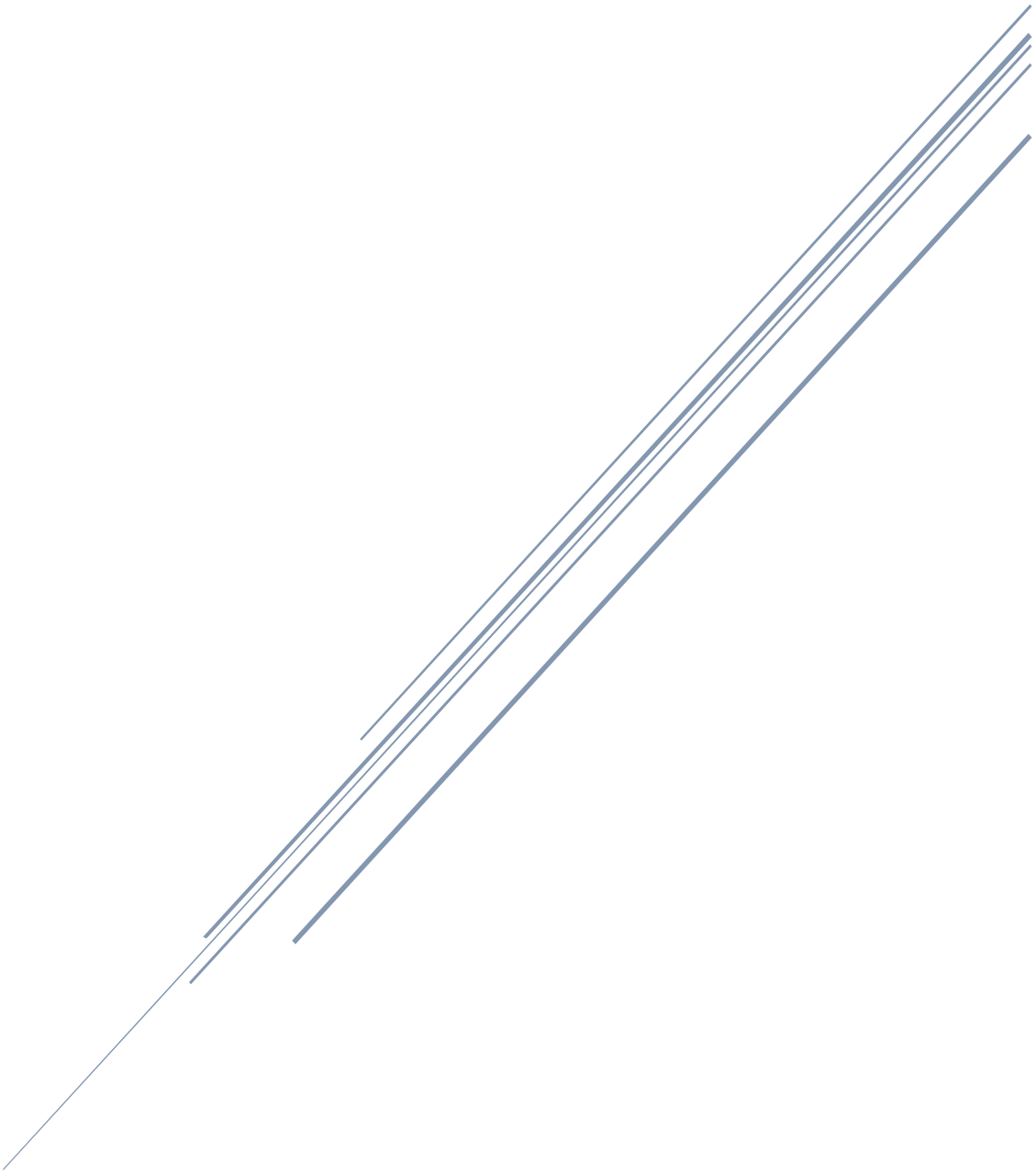
```
    </div>
    <div class="border-top pt-3">
        <small class="text-muted">
            Already Have An Account? <a class="ml-2" href="{{ url_for('login') }}">Sign In</a>
        </small>
    </div>
{% endblock content %}
```

# account.html



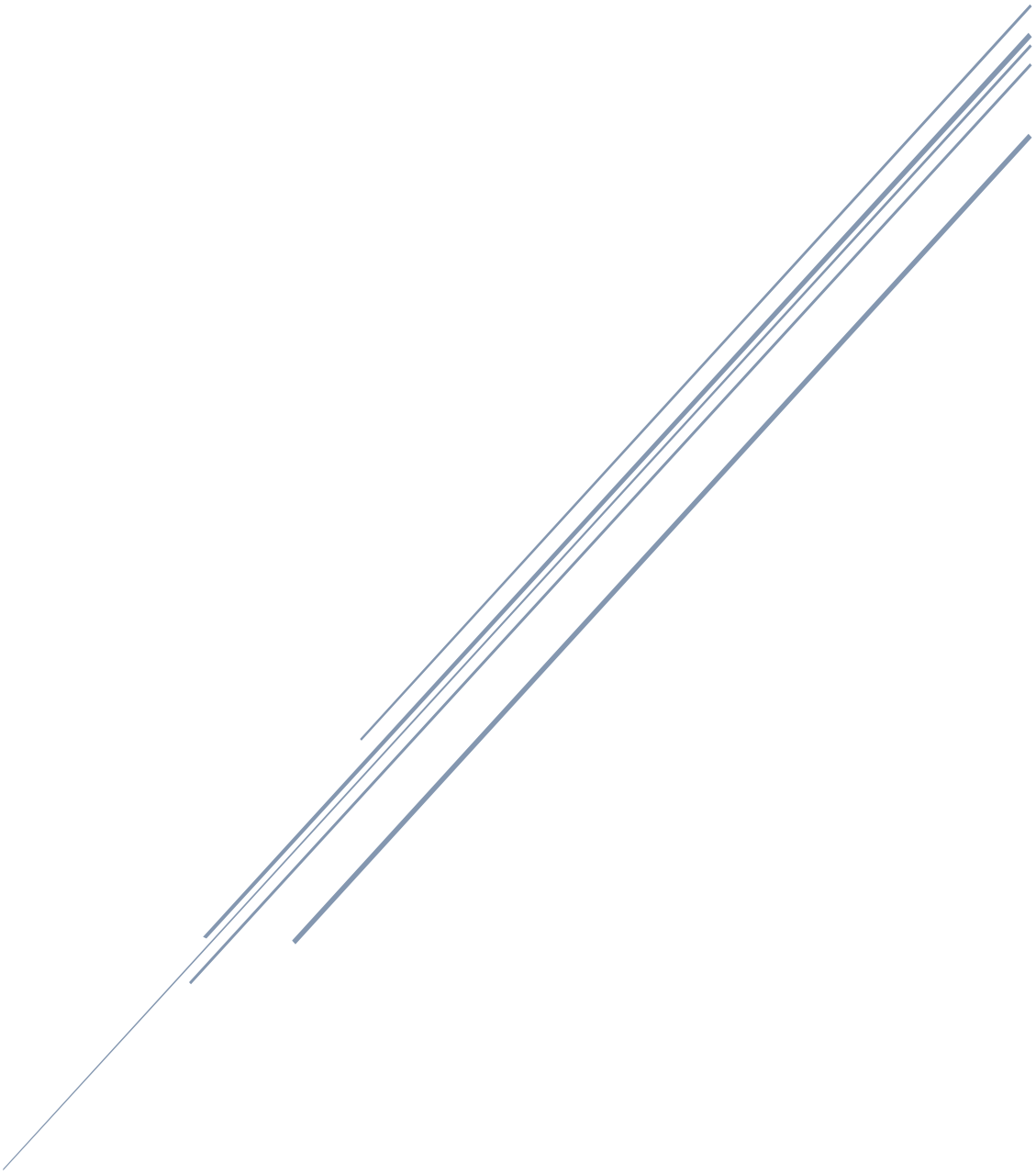
```
{% extends "layout_content.html" %}
{% block content %}
  <div class="content-section">
    <div class="media">
      <img class="rounded-circle account-img" src="{{ image_file }}">
      <div class="media-body">
        <h2 class="account-heading">{{ current_user.username }}</h2>
        <p class="text-secondary">{{ current_user.email }}</p>
      </div>
    </div>
      <form method="POST" action="" enctype="multipart/form-data">
        {{ form.hidden_tag() }}
        <fieldset class="form-group">
          <legend class="border-bottom mb-4">Account Info</legend>
          <div class="form-group">
            {{ form.username.label(class="form-control-label") }}

            {% if form.username.errors %}
              {{ form.username(class="form-control form-control-lg is-invalid") }}
              <div class="invalid-feedback">
                {% for error in form.username.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.username(class="form-control form-control-lg") }}
            {% endif %}
          </div>
          <div class="form-group">
            {{ form.email.label(class="form-control-label") }}
            {% if form.email.errors %}
              {{ form.email(class="form-control form-control-lg is-invalid") }}
              <div class="invalid-feedback">
                {% for error in form.email.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.email(class="form-control form-control-lg") }}
            {% endif %}
          </div>
          <div class="form-group">
            {{ form.picture.label() }}
            {{ form.picture(class="form-control-file") }}
            {% if form.picture.errors %}
              {% for error in form.picture.errors %}
                <span class="text-danger">{{ error }}</span></br>
              {% endfor %}
            {% endif %}
          </div>
        </fieldset>
        <div class="form-group">
          {{ form.submit(class="btn btn-outline-info") }}
        </div>
      </form>
  </div>
{% endblock content %}
```

# add_student_to_class.html



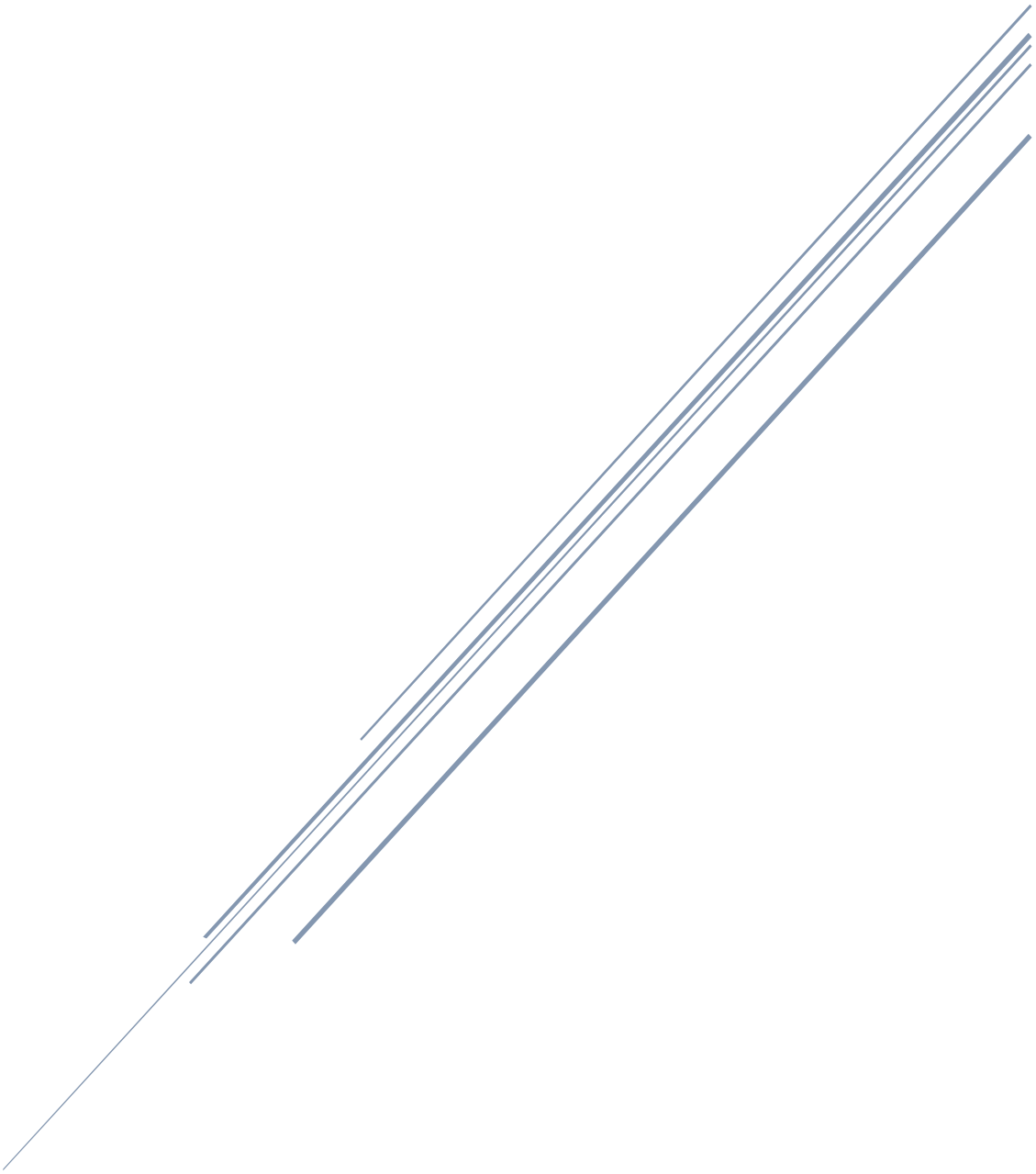
```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.class_id.label(class="form-control-label") }}
        {% if form.class_id.errors %}
          {{ form.class_id(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.class_id.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.class_id(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

# all_classes.html



```
{% extends "layout_content.html" %}
{% block content %}
  <h1 class="mb-3" style="display: inline-block;">My Classes ({{ classes.total }})</h1>
  <a class="btn btn-secondary btn-sm ml-4 mb-3" style="display: inline-block;" href="{{ url_for('new_class') }}">Add Class</a>
  {% for class in classes.items %}
    <article class="media content-section parent">

      <div class="ribbon"><a style="color:white;" href="{{ url_for('courses', course_id=class.course.id) }}">{{ class.course.name }}</a></div>

      <div class="media-body">
        <div class="article-metadata">
          <small class="text-muted mr-2">Beginning: {{ class.class_starting_date.strftime('%d %b %Y') }}</small>
        </div>
        <h2><a class="article-title" href="{{ url_for('xclass', class_id=class.id) }}">{{ class.class_name }}</a></h2>
        <div class="mt-3 mb-0">
          <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('xclass', class_id=class.id) }}">Manage</a>
          <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('class_report', class_id=class.id) }}">Get Report</a>
          <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_class', class_id=class.id) }}">Update</a>
          <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-target="#deleteModal{{ class.id }}">Delete</button>
        </div>
      </div>
    </article>
    <!-- Modal -->
    <div class="modal fade" id="deleteModal{{ class.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
      <div class="modal-dialog" role="document">
        <div class="modal-content">

          <div class="modal-header">
            <h5 class="modal-title" id="deleteModalLabel">Delete Class?</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <p class="modal-body"><font color="red">Warning: All this Classes students will also be deleted if this class is the only class they are assigned to!</font></p>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <form action="{{ url_for('delete_class', class_id=class.id) }}" method="POST">
              <input class="btn btn-danger" type="submit" value="Delete">
            </form>
          </div>
        </div>
      </div>
    </div>
  {% endfor %}
  {% for page_num in classes.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
    {% if page_num %}
      {% if classes.page == page_num %}
        <a class="btn btn-info mb-4" href="{{ url_for('classes', page=page_num) }}">{{ page_num }}</a>
      {% else %}
        <a class="btn btn-outline-info mb-4" href="{{ url_for('classes', page=page_num) }}">{{ page_num }}</a>
      {% endif %}
    {% else %}
      ...
    {% endif %}
  {% endfor %}

{% endblock content %}
```

# all_courses.html



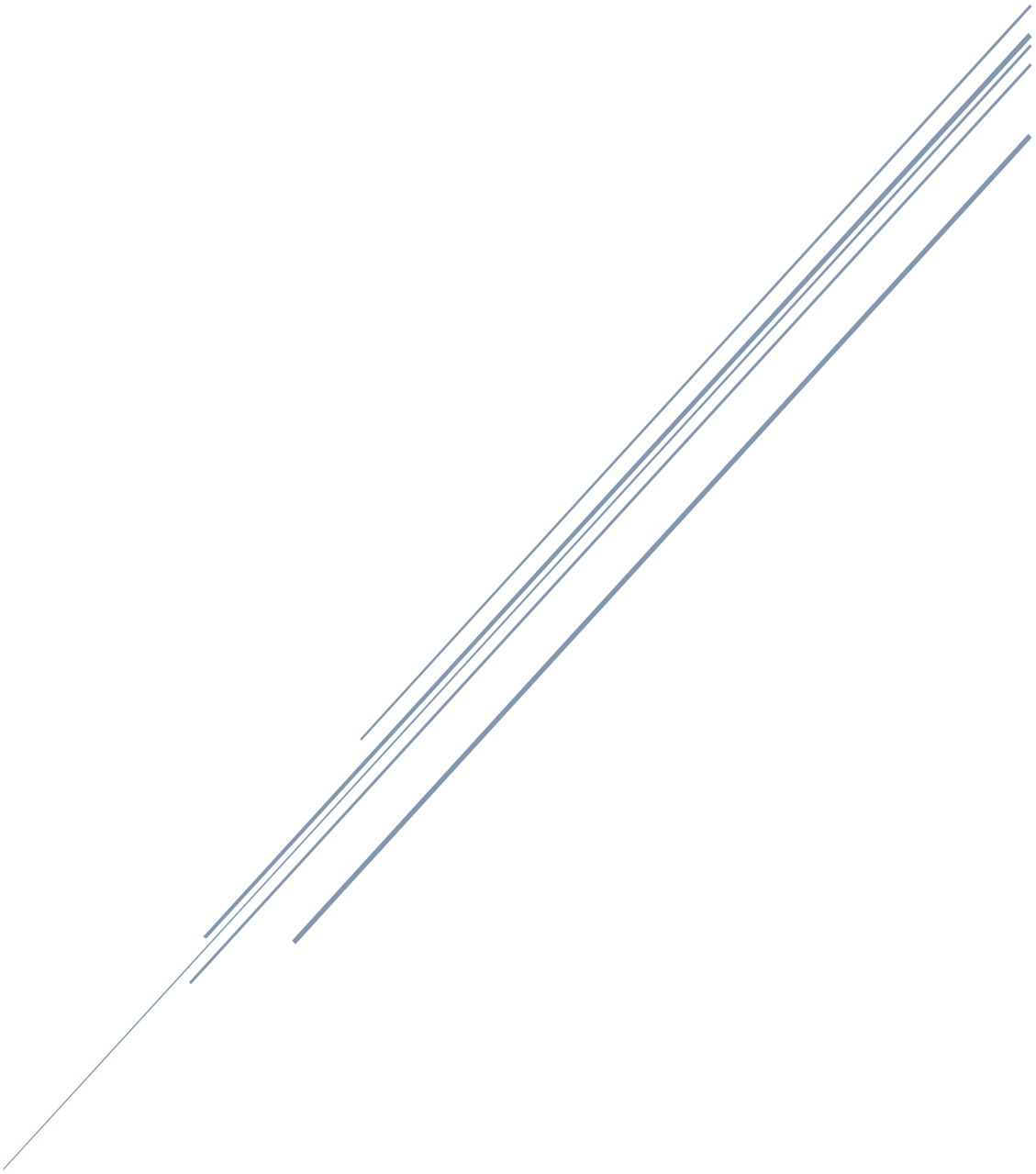
```
{% extends "layout_content.html" %}
{% block content %}
  <h1 class="mb-3" style="display: inline-block;">My Courses ({{ courses.total }})</h1>
  <a class="btn btn-secondary btn-sm ml-4 mb-3" style="display: inline-block;" href="{{ url_for('new_course') }}">Add
Course</a>
  {% for course in courses.items %}
    <article class="media content-section">
      <div class="media-body">

        <div style="display: inline-block">
          <h2><a class="article-title" href="{{ url_for('courses', course_id=course.id) }}">{{ course.name }}</a></h2>
        </div>

        <div style="display: inline-block; float:right;">
          <!-- <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('courses', course_id=course.id) }}">Manage</a> -
->
          <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('new_topic', course_id=course.id, next='courses')
}}">Add Topic</a>
          <a class="btn btn-secondary btn-sm mt-1 ml-1 mb-1" href="{{ url_for('new_exam', course_id=course.id, next='courses')
}}">Add Exam</a>
          <div style="display: inline-block;" class="dropdown show ml-1">
            <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class='fas fa-bars'></i>
            </a>
            <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
              <a class="dropdown-item" href="{{ url_for('update_course', course_id=course.id) }}">Update Details</a>
              <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModal{{ course.id }}">Delete</button>
            </div>
          </div>
          <!-- <a class="btn btn-secondary btn-sm mt-1 ml-1 mb-1" href="{{ url_for('update_course', course_id=course.id)
}}">Update Details</a>
          <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModal{{ course.id }}">Delete</button> -->
        </div>

        <div class="mb-3">
          <p class="m-0">Duration: {{ course.start_date.strftime('%b %Y') }}-{{
course.start_date.replace(year=course.start_date.year + course.year_num).strftime('%y') }} ({{ course.year_num }} Years)</p>
          <p class="m-0">Grading System: {{ course.grading_system[1:] }}</p>

          {% if course.classes|length > 0 %}
          <p>Classes Enrolled:
          {% for xclass in course.classes %}
          <br>- {{ xclass.class_name }}
          {% endfor %}
          </p>
          {% else %}
          <p>Classes Enrolled: None</p>
          {% endif %}

        </div>


        {% if course.topics %}
          <h5>Topics</h5>
          {% for topic in course.topics %}
            <article class="content-section" style="background:#eeeeee;">
              <div style="display: inline-block;">
              <a href="{{ url_for('topic', topic_id=topic.id) }}"><h6 class="mt-2">{{ topic.name }}</h6></a>
              </div>
              <div style="display: inline-block;" class="float-right">

                <div style="display: inline-block;" class="dropdown show">
```

```
                    <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                      <i class='fas fa-bars'></i>
                    </a>
                    <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                      <a class="dropdown-item" href="{{ url_for('topic', topic_id=topic.id) }}">Manage</a>
                      <a class="dropdown-item" href="{{ url_for('update_topic', topic_id=topic.id) }}">Update Details</a>
                      <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModalT{{ topic.id }}">Delete</button>
                    </div>
                  </div>
                </div>
              </article>
              <!-- Modal -->
              <div class="modal fade" id="deleteModalT{{ topic.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
                <div class="modal-dialog" role="document">
                  <div class="modal-content">
                    <div class="modal-header">
                      <h5 class="modal-title" id="deleteModalLabel">Delete Topic?</h5>
                      <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                      </button>
                    </div>
                    <p class="modal-body"><font color="red">Warning: All this Topic's homeworks and tests will also be
deleted!</font></p>
                    <div class="modal-footer">
                      <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                      <form action="{{ url_for('delete_topic', topic_id=topic.id) }}" method="POST">
                        <input class="btn btn-danger" type="submit" value="Delete">
                      </form>
                    </div>
                  </div>
                </div>
              </div>
            {% endfor %}
          {% endif %}


          {% if course.exams %}
            <h5>Exams</h5>
            {% for exam in course.exams %}
              <article class="content-section" style="background:#eeeeee;">
                <div style="display: inline-block;">
                <a href="{{ url_for('courses', course_id=course.id) }}"><h6 class="mt-2">{{ exam.name }}</h6></a>
                </div>
                <div style="display: inline-block;" class="float-right">
                  <!-- <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('courses', course_id=course.id)
}}">Manage</a> -->
                  <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_exam', exam_id=exam.id) }}">Mark
Exam</a>
                  <div style="display: inline-block;" class="dropdown show">
                    <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                      <i class='fas fa-bars'></i>
                    </a>
                    <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                      <a class="dropdown-item" href="{{ url_for('courses', course_id=course.id) }}">Manage</a>
                      <a class="dropdown-item" href="{{ url_for('update_exam', exam_id=exam.id) }}">Update Details</a>
                      <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModalE{{ exam.id }}">Delete</button>
                    </div>
                  </div>
                  <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_exam', exam_id=exam.id) }}">Update
```

```
Details</a>
                <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModalE{{ exam.id }}">Delete</button> -->
              </div>
            </article>
            <!-- Modal -->
            <div class="modal fade" id="deleteModalE{{ exam.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
              <div class="modal-dialog" role="document">
                <div class="modal-content">
                  <div class="modal-header">
                    <h5 class="modal-title" id="deleteModalLabel">Delete Exam?</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                      <span aria-hidden="true">&times;</span>
                    </button>
                  </div>
                  <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                    <form action="{{ url_for('delete_exam', exam_id=exam.id) }}" method="POST">
                      <input class="btn btn-danger" type="submit" value="Delete">
                    </form>
                  </div>
                </div>
              </div>
            </div>
          {% endfor %}
        {% endif %}


      </div>
    </article>
    <!-- Modal -->
    <div class="modal fade" id="deleteModal{{ course.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel"
aria-hidden="true">
      <div class="modal-dialog" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="deleteModalLabel">Delete Course?</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <p class="modal-body"><font color="red">Warning: All this Course's topics, classes and students will also be
deleted!</font></p>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <form action="{{ url_for('delete_course', course_id=course.id) }}" method="POST">
              <input class="btn btn-danger" type="submit" value="Delete">
            </form>
          </div>
        </div>
      </div>
    </div>
  {% endfor %}
  {% for page_num in courses.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
    {% if page_num %}
      {% if courses.page == page_num %}
        <a class="btn btn-info mb-4" href="{{ url_for('courses', page=page_num) }}">{{ page_num }}</a>
      {% else %}
        <a class="btn btn-outline-info mb-4" href="{{ url_for('courses', page=page_num) }}">{{ page_num }}</a>
      {% endif %}
    {% else %}
      ...
    {% endif %}
```

```
    {% endfor %}



{% endblock content %}
```

# all_students.html

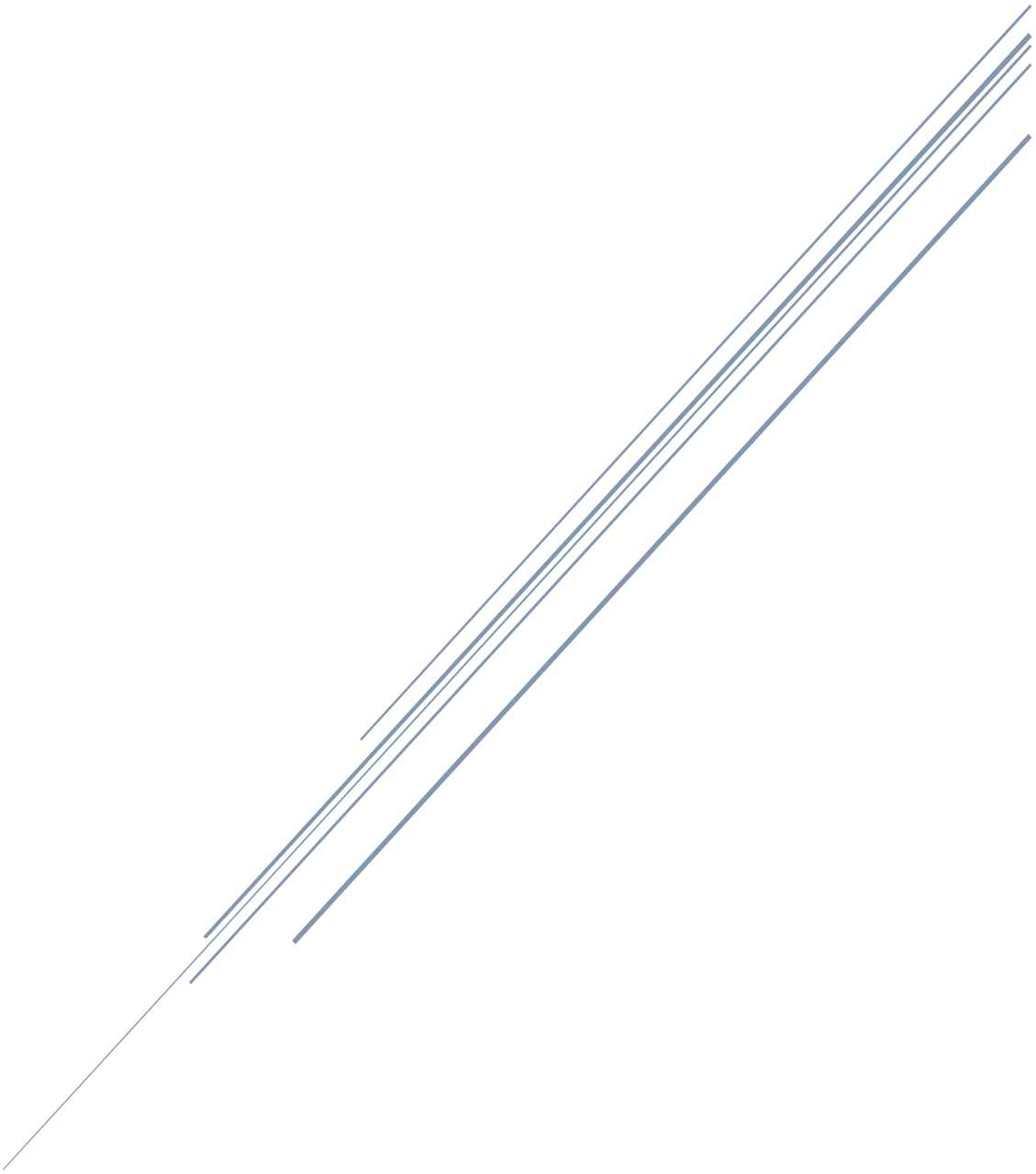


```
{% extends "layout_content.html" %}
{% block content %}



<form method="POST" action="">
    {{ s_form.hidden_tag() }}
    <div class="form-group" style="display:flex;">
        {% if s_form.search_query.errors %}
            {{ s_form.search_query(class="form-control form-control-lg is-invalid", style="flex:90%;", placeholder="Search
Students...") }}
            <div class="invalid-feedback">
                {% for error in s_form.search_query.errors %}
                    <span>{{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ s_form.search_query(class="form-control form-control-lg", style="flex:90%;", placeholder="Search Students...") }}
        {% endif %}
        {{ s_form.submit(class="btn btn-light", style="flex:10%;border: 1px solid #dddddd;") }}
    </div>
</form>
  {% if srch %}
      {% if search_results.all()|length > 0 %}
          <h3>Search Results</h3>
          {% for student in search_results %}
              <article class="media content-section">
                <div class="media-body">
                  <!--<div class="article-metadata">
                    <small class="text-muted mr-2">Beginning: \{\{ class.class_starting_date.strftime('%d %b %Y') \}\}</small>
                    <small class="text-muted">Week No: x</small>
                  </div>-->
                  <div style="display: inline-block;">
                    <h2><a class="article-title" href="{{ url_for('student', student_id=student.id) }}">{{ student.name }}</a></h2>
                  </div>
                  <div style="display: inline-block;float:right;">
                    <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('student_report', student_id=student.id) }}">Get
Report</a>
                    <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('add_to_class', student_id=student.id)
}}">Add To Class</a>
                    <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('remove_from_class', student_id=student.id)
}}">Remove From Class</a>

                    <div style="display: inline-block;" class="dropdown show">
                      <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <i class='fas fa-bars'></i>
                      </a>
                      <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                        <a class="dropdown-item" href="{{ url_for('student', student_id=student.id) }}">Manage</a>
                        <a class="dropdown-item" href="{{ url_for('update_student', student_id=student.id) }}">Update Details</a>
                        <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModal{{ student.id }}">Delete</button>
                      </div>
                    </div>
                    <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_student', student_id=student.id)
}}">Update Details</a>
                    <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-target="#deleteModal{{
student.id }}">Delete</button> -->
                </div>
              </div>
              </article>
              <!-- Modal -->
              <div class="modal fade" id="deleteModal{{ student.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
```

```
            <div class="modal-dialog" role="document">
              <div class="modal-content">
                <div class="modal-header">
                  <h5 class="modal-title" id="deleteModalLabel">Delete Student?</h5>
                  <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                  </button>
                </div>
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                  <form action="{{ url_for('delete_student', student_id=student.id) }}" method="POST">
                    <input class="btn btn-danger" type="submit" value="Delete">
                  </form>
                </div>
              </div>
            </div>
          </div>
        {% endfor %}

      {% else %}
        <h3>No results found!</h3>
      {% endif %}
    {% endif %}









  <h1 class="mb-3" style="display: inline-block;">My Students ({{ students|length }})</h1>
  <a class="btn btn-secondary btn-sm ml-4 mb-3" style="display: inline-block;" href="{{ url_for('new_student') }}">Add
Student</a>
  {% for student in students %}
    <article class="media content-section">
      <div class="media-body">
        <!--<div class="article-metadata">
          <small class="text-muted mr-2">Beginning: \{\{ class.class_starting_date.strftime('%d %b %Y') \}\}</small>
          <small class="text-muted">Week No: x</small>
        </div>-->
        <div style="display: inline-block;">
          <h2><a class="article-title" href="{{ url_for('student', student_id=student.id) }}">{{ student.name }}</a></h2>
          <!-- <p class="mb-1">[Summary of student.]</p> -->
        </div>
        <div style="display: inline-block;float:right;">
          <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('student_report', student_id=student.id) }}">Get
Report</a>
          <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('add_to_class', student_id=student.id) }}">Add To
Class</a>
          <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('remove_from_class', student_id=student.id)
}}">Remove From Class</a>

          <div style="display: inline-block;" class="dropdown show">
            <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class='fas fa-bars'></i>
            </a>
            <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
              <a class="dropdown-item" href="{{ url_for('student', student_id=student.id) }}">Manage</a>
              <a class="dropdown-item" href="{{ url_for('update_student', student_id=student.id) }}">Update Details</a>
              <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModal{{ student.id }}">Delete</button>
            </div>
          </div>
```

```
          <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_student', student_id=student.id) }}">Update Details</a>
          <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-target="#deleteModal{{ student.id }}">Delete</button> -->
        </div>
      </div>
    </article>
    <!-- Modal -->
    <div class="modal fade" id="deleteModal{{ student.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
      <div class="modal-dialog" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="deleteModalLabel">Delete Student?</h5>
            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <form action="{{ url_for('delete_student', student_id=student.id) }}" method="POST">
              <input class="btn btn-danger" type="submit" value="Delete">
            </form>
          </div>
        </div>
      </div>
    </div>
  {% endfor %}

  <!-- &% for page_num in students.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
    &% if page_num %}
      &% if students.page == page_num %}
        <a class="btn btn-info mb-4" href="&& url_for('students', page=page_num) }}">&& page_num }}</a>
      &% else %}
        <a class="btn btn-outline-info mb-4" href="&& url_for('students', page=page_num) }}">&& page_num }}</a>
      &% endif %}
    &% else %}
      ...
    &% endif %}
  &% endfor %} -->

{% endblock content %}
```

# all_topics.html



```
{% extends "layout_content.html" %}
{% block content %}

<form method="POST" action="">
    {{ s_form.hidden_tag() }}
    <div class="form-group" style="display:flex;">
        {% if s_form.search_query.errors %}
            {{ s_form.search_query(class="form-control form-control-lg is-invalid", style="flex:90%;", placeholder="Search Topics...")
}}
            <div class="invalid-feedback">
                {% for error in s_form.search_query.errors %}
                    <span>{{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ s_form.search_query(class="form-control form-control-lg", style="flex:90%;", placeholder="Search Topics...") }}
        {% endif %}
        {{ s_form.submit(class="btn btn-light", style="flex:10%;border: 1px solid #dddddd;") }}
    </div>
</form>
  {% if srch %}
      {% if search_results.all()|length > 0 %}
          <h3>Search Results</h3>
          {% for topic in search_results %}
              <article class="media content-section">
                <div class="media-body">

                  <div style="display: inline-block">
                    <h2><a class="article-title" href="{{ url_for('topic', topic_id=topic.id) }}">{{ topic.name }}</a></h2>
                  </div>

                  <div style="display: inline-block; float:right;">
                    <div style="display: inline-block;" class="dropdown show">
                      <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <i class='fas fa-plus'></i>
                      </a>
                      <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                        <a class="dropdown-item" href="{{ url_for('new_homework', topic_id=topic.id) }}">Add Homework</a>
                        <a class="dropdown-item" href="{{ url_for('new_test', topic_id=topic.id) }}">Add Test</a>
                      </div>
                    </div>

                    <div style="display: inline-block;" class="dropdown show">
                      <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <i class='fas fa-bars'></i>
                      </a>
                      <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                        <a class="dropdown-item" href="{{ url_for('topic', topic_id=topic.id) }}">Manage</a>
                        <a class="dropdown-item" href="{{ url_for('update_topic', topic_id=topic.id) }}">Update Details</a>
                        <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModal{{ topic.id }}">Delete</button>
                      </div>
                    </div>
                    <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_topic', topic_id=topic.id) }}">Update
Details</a>
                    <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModal{{ topic.id }}">Delete</button> -->
                  </div>

                  <p>{{ topic.start_date.strftime('%b %Y') }}-{{ topic.end_date.strftime('%b %Y') }} ({{ topic.end_date-topic.start_date }}
TIme)</p>

                  {% if topic.homeworks|length > 0 %}
```

```
            <h5>Homeworks</h5>
            {% for homework in topic.homeworks %}
              <article class="content-section" style="background:#eeeeee;">
                <div style="display: inline-block;">
                <h6 class="mt-2">{{ homework.name }}</h6>
                </div>
                <div style="display: inline-block;" class="float-right">
                  <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_homework',
homework_id=homework.id) }}">Mark Homework</a>

                <div style="display: inline-block;" class="dropdown show">
                  <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <i class='fas fa-bars'></i>
                  </a>
                  <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                    <a class="dropdown-item" href="{{ url_for('homework', homework_id=homework.id) }}">Manage</a>
                    <a class="dropdown-item" href="{{ url_for('update_homework', homework_id=homework.id) }}">Update
Details</a>
                    <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModalHH{{ homework.id }}">Delete</button>
                  </div>
                </div>
                <!-- <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('homework',
homework_id=homework.id) }}">Manage</a>
                <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_homework',
homework_id=homework.id) }}">Mark Homework</a>
                <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_homework',
homework_id=homework.id) }}">Update Details</a>
                <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModalHH{{ homework.id }}">Delete</button> -->
                <!-- Modal -->
                <div class="modal fade" id="deleteModalHH{{ homework.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
                  <div class="modal-dialog" role="document">
                    <div class="modal-content">
                      <div class="modal-header">
                        <h5 class="modal-title" id="deleteModalLabel">Delete Homework?</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                          <span aria-hidden="true">&times;</span>
                        </button>
                      </div>
                      <p class="modal-body"><font color="red">Warning: All this Topics tests and homeworks will also be
deleted!</font></p>

                      <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                        <form action="{{ url_for('delete_homework', homework_id=homework.id) }}" method="POST">
                          <input class="btn btn-danger" type="submit" value="Delete">
                        </form>
                      </div>
                    </div>
                  </div>
                </div>
              </article>
            {% endfor %}
          {% endif %}


          {% if topic.tests|length > 0 %}
            <h5>Tests</h5>
            {% for test in topic.tests %}
              <article class="content-section" style="background:#eeeeee;">
```

```
                    <div style="display: inline-block;">
                    <h6 class="mt-2">{{ test.name }}</h6>
                    </div>
                    <div style="display: inline-block;" class="float-right">
                      <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_test', test_id=test.id) }}">Mark
Test</a>

                      <div style="display: inline-block;" class="dropdown show">
                        <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                          <i class='fas fa-bars'></i>
                        </a>
                        <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                          <a class="dropdown-item" href="{{ url_for('test', test_id=test.id) }}">Manage</a>
                          <a class="dropdown-item" href="{{ url_for('update_test', test_id=test.id) }}">Update Details</a>
                          <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModalTTT{{ test.id }}">Delete</button>
                        </div>
                      </div>
                      <!-- <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('test', test_id=test.id) }}">Manage</a>
                      <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_test', test_id=test.id) }}">Update
Details</a>
                      <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModalTT{{ test.id }}">Delete</button> -->
                    </div>
                    <!-- Modal -->
                    <div class="modal fade" id="deleteModalTT{{ test.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
                      <div class="modal-dialog" role="document">
                        <div class="modal-content">
                          <div class="modal-header">
                            <h5 class="modal-title" id="deleteModalLabel">Delete Test?</h5>
                            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                              <span aria-hidden="true">&times;</span>
                            </button>
                          </div>
                          <p class="modal-body"><font color="red">Warning: All this Topics tests and homeworks will also be
deleted!</font></p>

                          <div class="modal-footer">
                            <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                            <form action="{{ url_for('delete_test', test_id=test.id) }}" method="POST">
                              <input class="btn btn-danger" type="submit" value="Delete">
                            </form>
                          </div>
                        </div>
                      </div>
                    </div>
                  </article>
                {% endfor %}
              {% endif %}


            </div>
          </article>
          <!-- Modal -->
          <div class="modal fade" id="deleteModal{{ topic.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel"
aria-hidden="true">
            <div class="modal-dialog" role="document">
              <div class="modal-content">
                <div class="modal-header">
                  <h5 class="modal-title" id="deleteModalLabel">Delete Topic?</h5>
                  <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
```

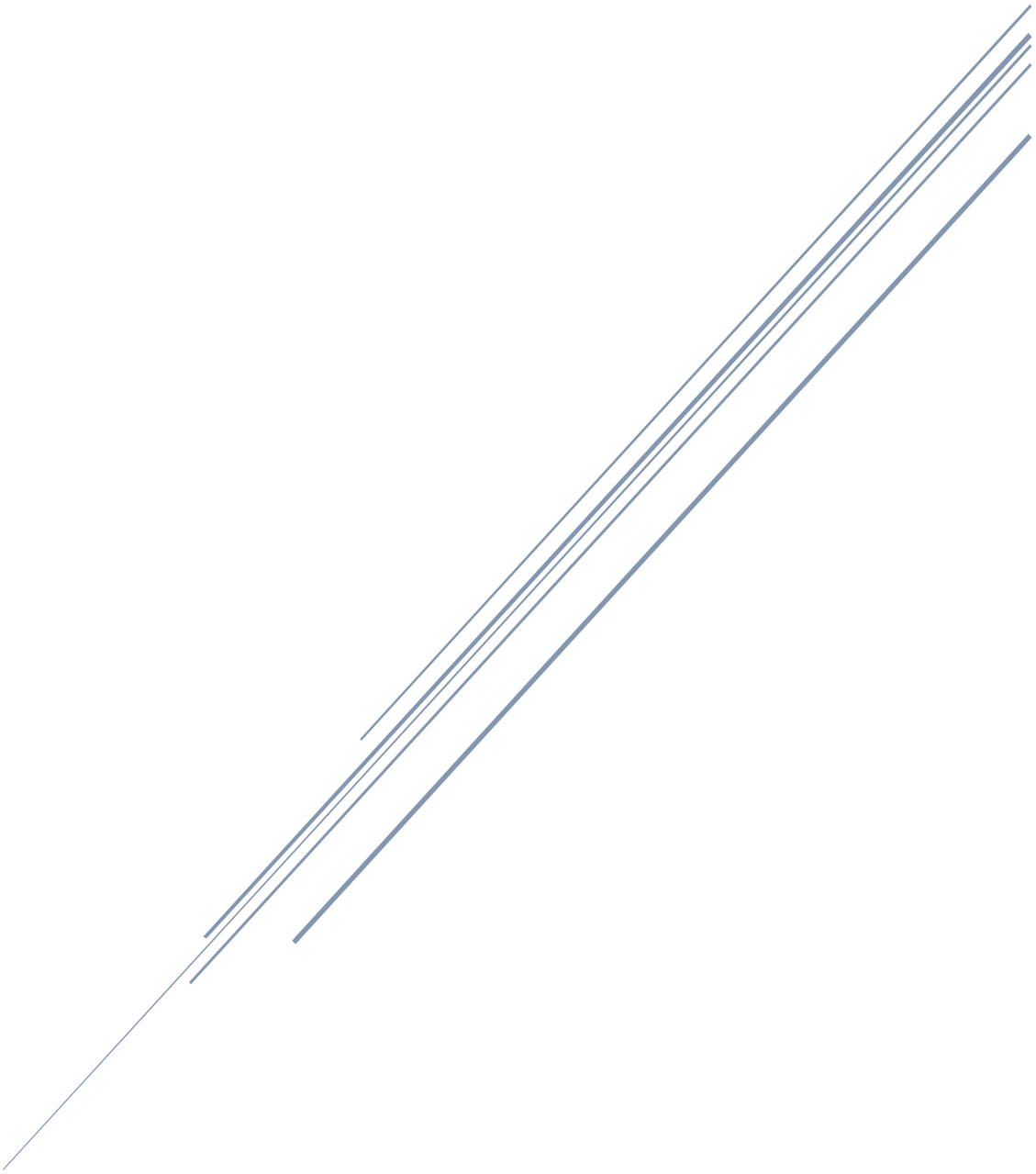
```
                </button>
              </div>
              <p class="modal-body"><font color="red">Warning: All this Topics tests and homeworks will also be
deleted!</font></p>

              <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                <form action="{{ url_for('delete_topic', topic_id=topic.id) }}" method="POST">
                  <input class="btn btn-danger" type="submit" value="Delete">
                </form>
              </div>
            </div>
          </div>
        </div>
        {% endfor %}

    {% else %}
        <h3>No results found!</h3>
    {% endif %}
 {% endif %}


{% for course in courses.items %}
   <!-- <br>
   <br> -->
   <h3 class="mb-3" style="display: inline-block;">{{ course.name }} Topics</h3>
   <a class="btn btn-secondary btn-sm ml-4 mb-3" style="display: inline-block;" href="{{ url_for('new_topic', course_id=course.id)
}}">Add Topic</a>



   {% for topic in course.topics %}
      <article class="media content-section">
        <div class="media-body">

          <div style="display: inline-block">
            <h2><a class="article-title" href="{{ url_for('topic', topic_id=topic.id) }}">{{ topic.name }}</a></h2>
          </div>

          <div style="display: inline-block; float:right;">

            <div style="display: inline-block;" class="dropdown show">
              <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                <i class='fas fa-plus'></i>
              </a>
              <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                <a class="dropdown-item" href="{{ url_for('new_homework', topic_id=topic.id) }}">Add Homework</a>
                <a class="dropdown-item" href="{{ url_for('new_test', topic_id=topic.id) }}">Add Test</a>
              </div>
            </div>

            <div style="display: inline-block;" class="dropdown show">
              <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                <i class='fas fa-bars'></i>
              </a>
              <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                <a class="dropdown-item" href="{{ url_for('topic', topic_id=topic.id) }}">Manage</a>
                <a class="dropdown-item" href="{{ url_for('update_topic', topic_id=topic.id) }}">Update Details</a>
                <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModal{{ topic.id }}">Delete</button>
              </div>
            </div>
```

```
        <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_topic', topic_id=topic.id) }}">Update Details</a>
        <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-target="#deleteModal{{ topic.id }}">Delete</button> -->
      </div>

      <p>
      {{ topic.start_date.strftime('%d %b %Y') }} - {{ topic.end_date.strftime('%d %b %Y') }}
      {% if topic.start_date < today and topic.end_date > today  %}
      <font color="white" style="background:green;">Current Topic</font>
      {% endif %}
      ({{ (topic.end_date-topic.start_date).days }} Days)
      </p>

      {% if topic.homeworks|length > 0 %}
        <h5>Homeworks</h5>
        {% for homework in topic.homeworks %}
          <article class="content-section" style="background:#eeeeee;">
            <div style="display: inline-block;">
            <a href="{{ url_for('homework', homework_id=homework.id) }}"><h6 class="mt-2">{{ homework.name }}</h6></a>
            </div>
            <div style="display: inline-block;" class="float-right">

              <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_homework', homework_id=homework.id) }}">Mark Homework</a>

              <div style="display: inline-block;" class="dropdown show">
                <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button" id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                  <i class='fas fa-bars'></i>
                </a>
                <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                  <a class="dropdown-item" href="{{ url_for('homework', homework_id=homework.id) }}">Manage</a>
                  <a class="dropdown-item" href="{{ url_for('update_homework', homework_id=homework.id) }}">Update Details</a>
                  <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-target="#deleteModalHH{{ homework.id }}">Delete</button>
                </div>
              </div>
              <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_homework', homework_id=homework.id) }}">Update Details</a>
              <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-target="#deleteModalHH{{ homework.id }}">Delete</button> -->
            </div>
            <!-- Modal -->
            <div class="modal fade" id="deleteModalHH{{ homework.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
              <div class="modal-dialog" role="document">
                <div class="modal-content">
                  <div class="modal-header">
                    <h5 class="modal-title" id="deleteModalLabel">Delete Homework?</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                      <span aria-hidden="true">&times;</span>
                    </button>
                  </div>
                  <p class="modal-body"><font color="red">Warning: All Students marks for this homework will also be deleted!</font></p>

                  <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                    <form action="{{ url_for('delete_homework', homework_id=homework.id) }}" method="POST">
                      <input class="btn btn-danger" type="submit" value="Delete">
                    </form>
                  </div>
```

```
                </div>
              </div>
            </div>
          </article>
        {% endfor %}
      {% endif %}


      {% if topic.tests|length > 0 %}
        <h5>Tests</h5>
        {% for test in topic.tests %}
          <article class="content-section" style="background:#eeeeee;">
            <div style="display: inline-block;">
            <a href="{{ url_for('test', test_id=test.id) }}"><h6 class="mt-2">{{ test.name }}</h6></a>
            </div>
            <div style="display: inline-block;" class="float-right">
              <a class="btn btn-primary btn-sm mt-1 mb-1 mr-1" href="{{ url_for('mark_test', test_id=test.id) }}">Mark Test</a>

              <div style="display: inline-block;" class="dropdown show">
                <a class="btn btn-secondary btn-sm mt-1 mb-1 mr-1 dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                  <i class='fas fa-bars'></i>
                </a>
                <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
                  <a class="dropdown-item" href="{{ url_for('test', test_id=test.id) }}">Manage</a>
                  <a class="dropdown-item" href="{{ url_for('update_test', test_id=test.id) }}">Update Details</a>
                  <button type="button" class="dropdown-item btn-danger" style="color:red;" data-toggle="modal" data-
target="#deleteModalTTT{{ test.id }}">Delete</button>
                </div>
              </div>
              <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_test', test_id=test.id) }}">Update
Details</a>
              <button type="button" class="btn btn-danger btn-sm ml-1 mb-1 mt-1 mr-0" data-toggle="modal" data-
target="#deleteModalTTT{{ test.id }}">Delete</button> -->
            </div>
            <!-- Modal -->
            <div class="modal fade" id="deleteModalTTT{{ test.id }}" tabindex="-1" role="dialog" aria-
labelledby="deleteModalLabel" aria-hidden="true">
              <div class="modal-dialog" role="document">
                <div class="modal-content">
                  <div class="modal-header">
                    <h5 class="modal-title" id="deleteModalLabel">Delete Test?</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                      <span aria-hidden="true">&amp;times;</span>
                    </button>
                  </div>
                  <p class="modal-body"><font color="red">Warning: All Students marks for this test will also be
deleted!</font></p>

                  <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                    <form action="{{ url_for('delete_test', test_id=test.id) }}" method="POST">
                      <input class="btn btn-danger" type="submit" value="Delete">
                    </form>
                  </div>
                </div>
              </div>
            </div>
          </article>
        {% endfor %}
      {% endif %}


    </div>
```

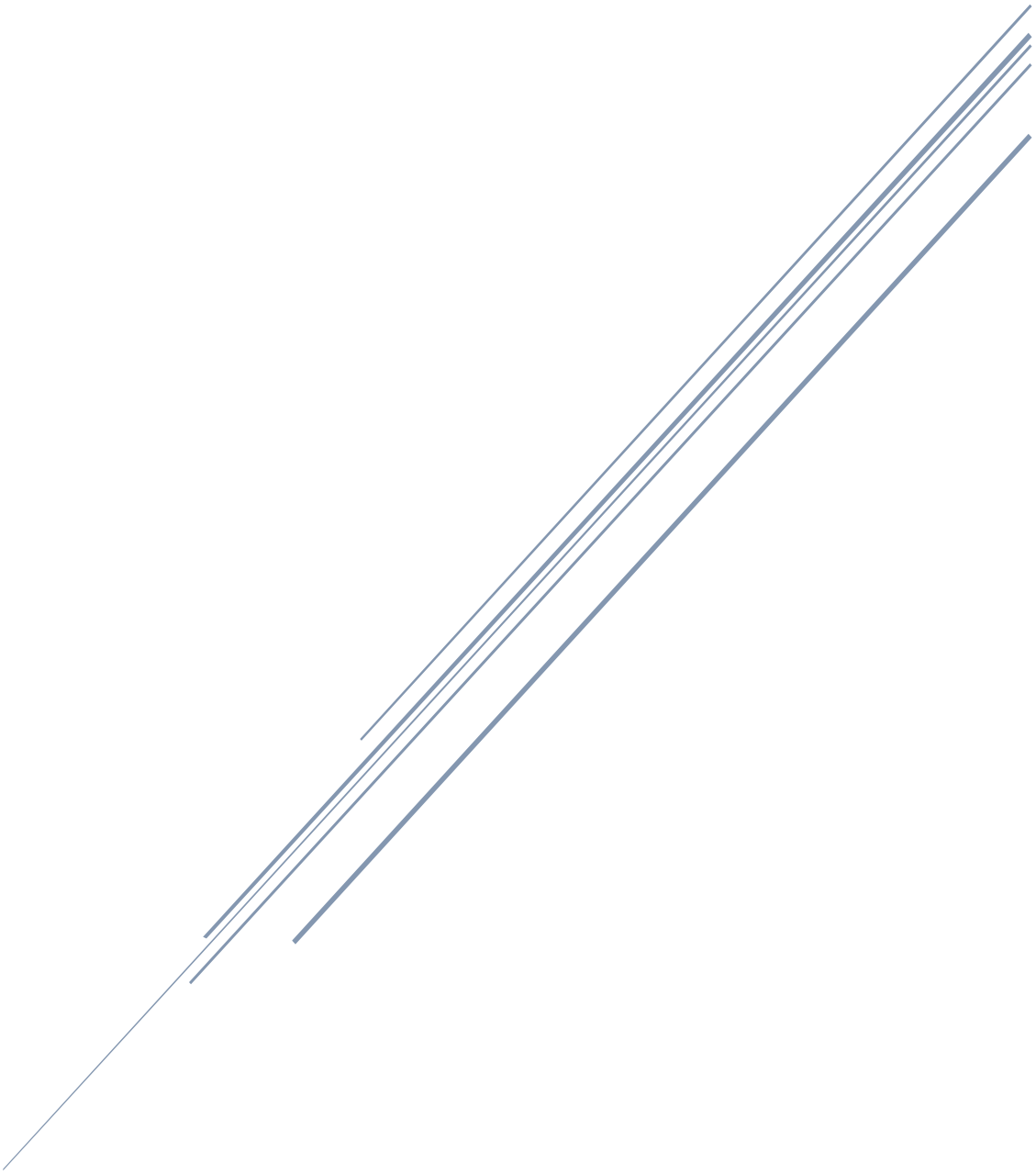
```
      </article>
      <!-- Modal -->
      <div class="modal fade" id="deleteModal{{ topic.id }}" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-
hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="deleteModalLabel">Delete Topic?</h5>
              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
              </button>
            </div>
            <p class="modal-body"><font color="red">Warning: All this Topics tests and homeworks will also be deleted!</font>
</p>

            <div class="modal-footer">
              <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
              <form action="{{ url_for('delete_topic', topic_id=topic.id) }}" method="POST">
                <input class="btn btn-danger" type="submit" value="Delete">
              </form>
            </div>
          </div>
        </div>
      </div>
    {% endfor %}

    {% endfor %}

    <br>

    {% for page_num in courses.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
      {% if page_num %}
        {% if courses.page == page_num %}
          <a class="btn btn-info mb-4" href="{{ url_for('topics', page=page_num) }}">{{ page_num }}</a>
        {% else %}
          <a class="btn btn-outline-info mb-4" href="{{ url_for('topics', page=page_num) }}">{{ page_num }}</a>
        {% endif %}
      {% else %}
        ...
      {% endif %}
    {% endfor %}

{% endblock content %}
```

# all_user_posts.html



```
{% extends "layout_content.html" %}
{% block content %}
    <h1 class="mb-3">Posts by {{ user.username }} ({{ posts.total }})</h1>
    {% for post in posts.items %}
        <article class="media content-section">
          <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + post.author.image_file) }}">
          <div class="media-body">
            <div class="article-metadata">
              <a class="mr-2" href="{{ url_for('user_posts', username=post.author.username) }}">{{ post.author.username }}</a>
              <small class="text-muted">{{ post.date_posted.strftime('%d %b %Y - %H:%M') }}</small>
            </div>
            <h2><a class="article-title" href="{{ url_for('post', post_id=post.id) }}">{{ post.title }}</a></h2>
            <p class="article-content">{{ post.content }}</p>
          </div>
        </article>
    {% endfor %}
    {% for page_num in posts.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
      {% if page_num %}
        {% if posts.page == page_num %}
          <a class="btn btn-info mb-4" href="{{ url_for('user_posts', username=user.username, page=page_num) }}">{{ page_num
}}</a>
        {% else %}
          <a class="btn btn-outline-info mb-4" href="{{ url_for('user_posts', username=user.username, page=page_num) }}">{{
page_num }}</a>
        {% endif %}
      {% else %}
        ...
      {% endif %}
    {% endfor %}
{% endblock content %}
```

# dash.html



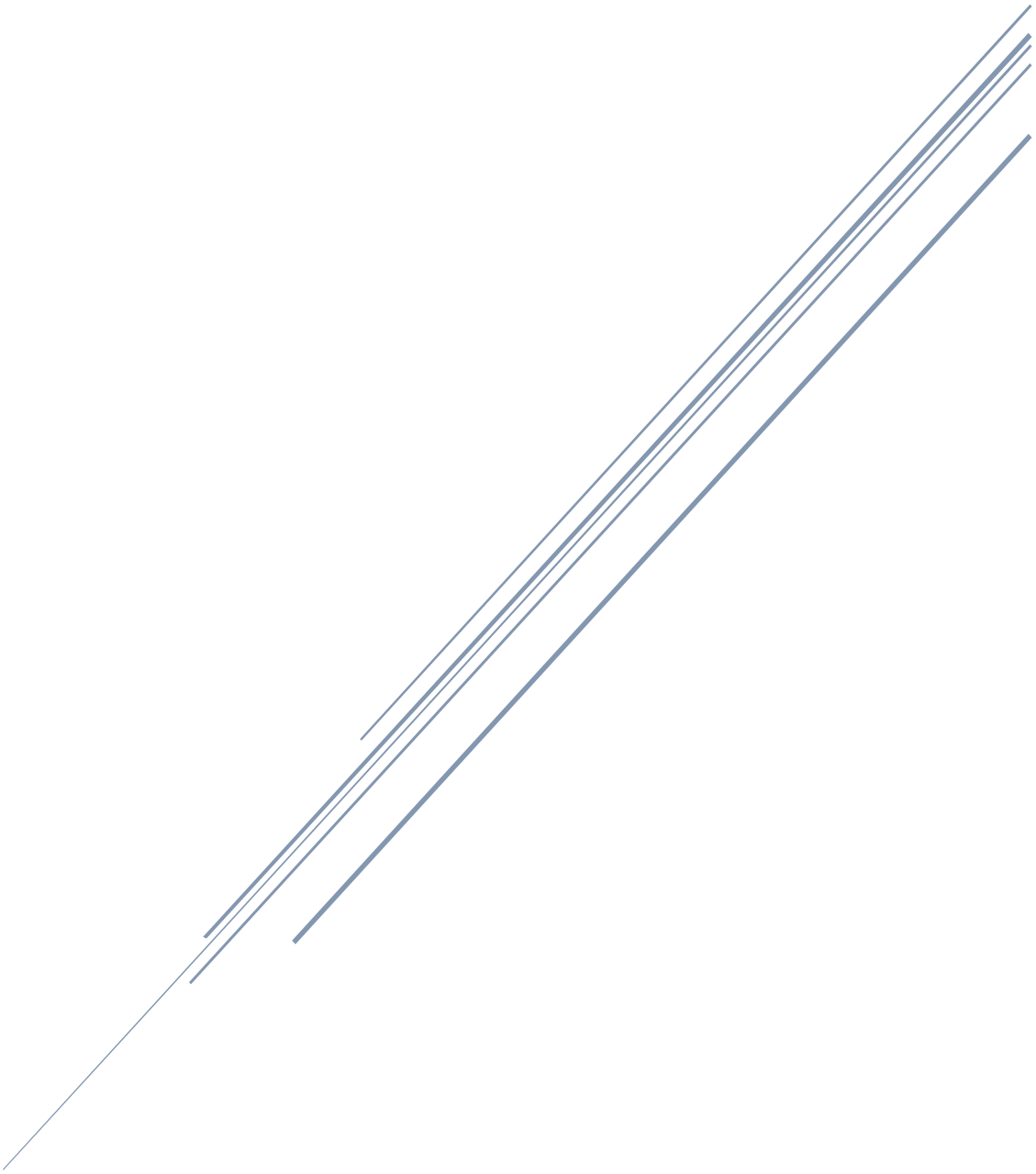
```
{% extends "layout_dashboard.html" %}
{% block content %}

{% if cur_tops|length %}

<div class="row">
 <div class="col-md-6">
  <div class="content-section">
   {% if cur_tops|length %}
     <h2 class="article-title">Current Topic{% if cur_tops|length > 1 %}s{% endif %}</h2>
     {% for cur_top in cur_tops %}
       <h5 class="article-content">{{ cur_top[0].name }}: {{ cur_top[1].name }}</h5>
     {% endfor %}
   {% endif %}
  </div>

  <div class="content-section">
   <h2 class="article-title">Homeworks / Tests</h2>
   {% if upcoming_homeworks|length %}
     <h5 class="article-content m-0 mb-2">Upcoming Homeworks:</h5>
     {% for up_hmwk in upcoming_homeworks %}
       <p class="article-content">- {{ up_hmwk[0].name }} (Due {{ up_hmwk[0].due_date.strftime('%d %b %Y') }})</p>
     {% endfor %}
   {% endif %}

   {% if due_homeworks|length %}
     <h5 class="article-content m-0 mb-2">Homeworks Due:</h5>
     {% for due_hmwk in due_homeworks %}
      <div class="d-block">
       <div style="display: inline-block; float:left;">
        <p class="article-content mb-2">- {{ due_hmwk[0].name }} (Due {{ due_hmwk[0].due_date.strftime('%d %b %Y')
}}</p>
       </div>
       <div style="display: inline-block; float:right;">
        <a class="btn btn-sm btn-primary"  href="{{ url_for('mark_homework', homework_id=due_hmwk[0].id) }}">Mark
Homework</a>
       </div>
       <p>)</p>
      </div>
     {% endfor %}
   {% endif %}


   {% if due_tests|length %}
     <h5 class="article-content m-0 mb-2">Tests Due:</h5>
     {% for due_test in due_tests %}
      <div class="d-block">
       <div style="display: inline-block; float:left;">
        <p class="article-content mb-2">- {{ due_test[0].name }} (Due {{ due_test[0].date.strftime('%d %b %Y') }}</p>
       </div>
       <div style="display: inline-block; float:right;">
        <a class="btn btn-sm btn-primary"  href="{{ url_for('mark_test', test_id=due_test[0].id) }}">Mark Test</a>
       </div>
       <p>)</p>
      </div>
     {% endfor %}
   {% endif %}
  </div>

  <div class="content-section">
   <h2 class="article-title">Homework Submissions</h2>
   {% if due_homeworks|length %}
   {% for due_hmwk in due_homeworks %}
   <h5 class="article-content mt-2">{{ due_hmwk[0].name }} (Due {{ due_hmwk[0].due_date.strftime('%d %b %Y') }})</h5>
   {% for xclass in cur_top_classes %}
```

```
    <p class="article-content m-0 mt-2"><b>{{ xclass.class_name }}</b></p>
    {% for student in xclass.students %}

    {% if homework_marks[due_hmwk[0].id][student.id] %}
      {% if homework_marks[due_hmwk[0].id][student.id][0].late %}
        <p class="article-content mb-0" style="background: orange;">- {{ student.name }}</p>
      {% else  %}
        <p class="article-content mb-0" style="background: #339933;">- {{ student.name }}</p>
      {% endif %}
    {% else %}
      <!-- <p class="article-content mb-0" style="background: #dd4444;">- {{ student.name }}</p> -->
      <div style="background: #dd4444;">
       <div style="display: inline-block;">
        <p class="article-content mb-2" >- {{ student.name }}</p>
       </div>
       <div style="display: inline-block; float:right;">

        <form style="display: inline-block; float:right;" method="POST" action="/submit_homework">
         <input name="homework_id" required="" type="hidden" value="{{ due_hmwk[0].id }}">
         <input name="student_id" required="" type="hidden" value="{{ student.id }}">
         <input style="display: inline-block; float:left;"  id="mark" name="mark" required="" type="int" value="">
         <input style="display: inline-block; float:left;"  id="date_handed_in" name="date_handed_in" required="" type="date"
value="">
         <input style="display: inline-block; float:right;" id="submit" name="submit" type="submit" value="Submit">
         <!-- <a style="display: inline-block; float:right;" type="submit"><font color="#000077">HAND IN</font></a> -->
        </form>
       </div>
      </div>
    {% endif %}
    {% endfor %}
    {% endfor %}
    {% endfor %}
    {% endif %}
  </div>
 </div>



 <div class="col-md-6">
  <div id="graph-container" class="content-section">

    <div class="row">
     <div class="col-md-7">
      <h2 class="article-title">Recent Performance</h2>
     </div>

     <div class="col-md-5 form-group">
      <label for="datasetChoice">Choose Dataset</label>
      <select class="form-control" id="datasetChoice">
       <option>Avg Class Performance Last Test</option>
       <option>Avg Class Performance By Assignment</option>
       <!-- <option>Avg Class Performance Over Time</option>
       <option>Avg Class Performance By Test</option>
       <option>5</option> -->
      </select>
     </div>
    </div>



    <canvas id="performanceGraph"></canvas>

    </div>
    <div class="content-section">
     <h2 class="article-title">Latest Test</h2>
```

```
      {% if due_tests|length %}
      {% for due_test in due_tests %}
      <h5 class="article-content mt-2">{{ due_test[0].name }} (Due {{ due_test[0].date.strftime('%d %b %Y') }})</h5>
      {% for xclass in cur_top_classes %}
      <p class="article-content m-0 mt-2"><b>{{ xclass.class_name }}</b></p>
      {% for student in xclass.students %}

      {% if test_marks[due_test[0].id][student.id] %}
        {% if test_marks[due_test[0].id][student.id][0].late %}
          <p class="article-content mb-0" style="background: orange;">- (Late) {{ student.name }}</p>
        {% else  %}
          <p class="article-content mb-0" style="background: #339933;">- {{ student.name }}</p>
        {% endif %}
      {% else %}
        <!-- <p class="article-content mb-0" style="background: #dd4444;">- {{ student.name }}</p> -->
        <div style="background: #dd4444;">
          <div style="display: inline-block;">
            <p class="article-content mb-2" >- {{ student.name }}</p>
          </div>
          <div style="display: inline-block; float:right;">
            <form style="display: inline-block; float:right;" method="POST" action="/submit_test">
              <input name="test_id" required="" type="hidden" value="{{ due_test[0].id }}">
              <input name="student_id" required="" type="hidden" value="{{ student.id }}">
              <input style="display: inline-block; float:left;"  id="mark" name="mark" required="" type="int" value="">
              <input style="display: inline-block; float:left;"  id="date_handed_in" name="date_handed_in" required=""
type="date" value="">
              <input style="display: inline-block; float:right;" id="submit" name="submit" type="submit" value="Submit">
              <!-- <a style="display: inline-block; float:right;" type="submit"><font color="#000077">HAND IN</font></a> -->
            </form>
          </div>
        </div>
      {% endif %}
      {% endfor %}
      {% endfor %}
      {% endfor %}
      {% endif %}
    </div>
  </div>
 </div>
{% else %}
<div class="d-flex justify-content-center">
 <span class="align-middle"><h1>You do not have any current topics.</h1>
 <h3><a class="justify-content-center" href="{{ url_for('topics') }}">Create one on the topics page!</a></h3></span>
</div>
{% endif %}

<!-- Linking all the scripts! -->
<script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
<script type="text/javascript" src="{{ url_for('static', filename='js/dashboard_charts.js') }}"></script>
<script type="text/javascript">

 // Setting graph to load onload.
 $(document).ready(function() {
   changeChart("performanceGraph", "bar",
            {{ class_perf.labels | safe }},
            {{ class_perf.datasets | safe }});
 });

 // Changing graph upon users' selection.
 $('#datasetChoice').on("change",function() {
    if ($(this).val() == "Avg Class Performance Last Test") {
      changeChart("performanceGraph", "bar",
              {{ class_perf.labels | safe }},
```

```
            {{ class_perf.datasets | safe }});

    } else if ($(this).val() == "Avg Class Performance Over Time") {
      changeChart("performanceGraph", "line",
            {{ classes_perf_time.labels | safe }},
            {{ classes_perf_time.datasets | safe }});

    } else if ($(this).val() == "Avg Class Performance By Assignment") {
      changeChart("performanceGraph", "line",
            {{ classes_perf_per_test.labels | safe }},
            {{ classes_perf_per_test.datasets | safe }});


    }
    else {
      console.log($(this).val());
    }
  });

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

# home.html

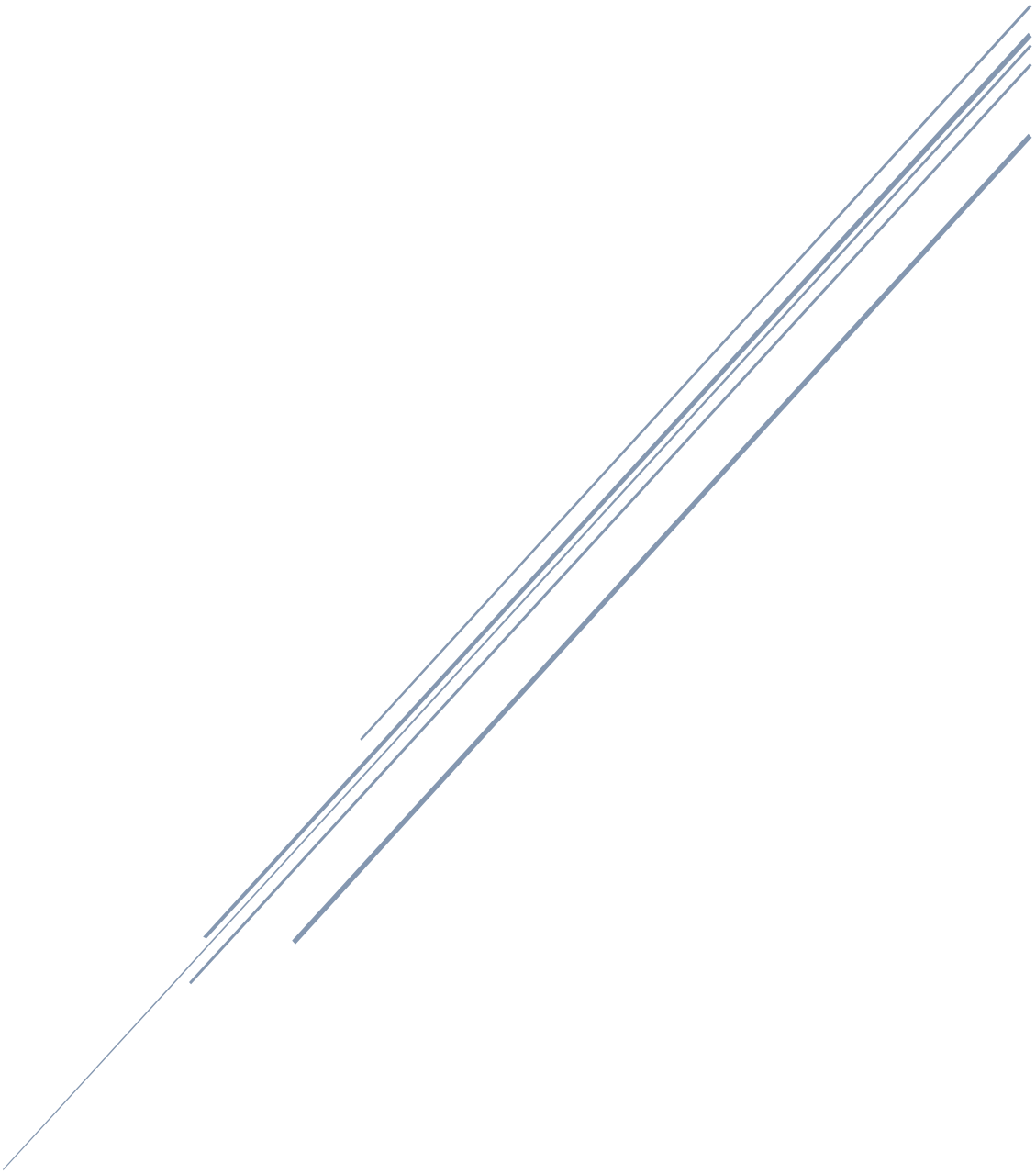


```
{% extends "layout_content.html" %}
{% block content %}

 <form method="POST" action="">
    {{ s_form.hidden_tag() }}
    <div class="form-group" style="display:flex;">
       {% if s_form.search_query.errors %}
          {{ s_form.search_query(class="form-control form-control-lg is-invalid", style="flex:90%;", placeholder="Search
Forum...") }}
          <div class="invalid-feedback">
             {% for error in s_form.search_query.errors %}
                <span>{{ error }}</span>
             {% endfor %}
          </div>
       {% else %}
          {{ s_form.search_query(class="form-control form-control-lg", style="flex:90%;", placeholder="Search Forum...") }}
       {% endif %}
       {{ s_form.submit(class="btn btn-light", style="flex:10%;border: 1px solid #dddddd;") }}
    </div>
 </form>
  {% if srch %}
     {% if search_results.all()|length > 0 %}
        <h4>Post Search Results</h4>
        {% for post in search_results %}
        <article  class="content-section">
          <div class="media mb-4">
            <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + post.author.image_file) }}">
            <div class="media-body">
              <div class="article-metadata">
                <a class="mr-2" href="{{ url_for('user_posts', username=post.author.username) }}">{{ post.author.username }}</a>
                <small class="text-muted">{{ post.date_posted.strftime('%d %b %Y - %H:%M') }}</small>
              </div>
              <h2><a class="article-title" href="{{ url_for('post', post_id=post.id) }}">{{ post.title }}</a></h2>
              <p class="article-content">{{ post.content }}</p>
            </div>
          </div>
          <form method="POST" action="{{ url_for('add_comment', post_id=post.id) }}">
            {{ form.hidden_tag() }}
            <legend class="border-bottom">{{ legend }}</legend>
            {{ form.comment.label(class="form-control-label", style="font-size:14px;") }}
            <div class="media form-group">

              {% if form.comment.errors %}
                {{ form.comment(class="form-control form-control-lg is-invalid", style="height: 37px;font-size:13px;") }}
                <div class="invalid-feedback">
                  {% for error in form.comment.errors %}
                    <span>{{ error }}</span>
                  {% endfor %}
                </div>
              {% else %}
                {{ form.comment(class="form-control form-control-lg", style="height: 37px;font-size:13px;") }}
              {% endif %}

              {{ form.submit(class="btn-sm btn-outline-info ml-2", style="font-size:13px;") }}
            </div>
          </form>
        </article>
        {% endfor %}


     {% else %}
        <h4>No post results found!</h4>
     {% endif %}

        {% if search_user_results.all()|length > 0 %}
```

```
            <h4>User Search Results</h4>
            {% for userR in search_user_results %}
               <article class="media content-section">
                    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + userR.image_file) }}">
                    <h2><a href="{{ url_for('user_posts', username=userR.username) }}">{{ userR.username }}</a></h2>
               </article>

            {% endfor %}

            {% else %}
               <h4>No user results found!</h4>
            {% endif %}
            <hr>

   {% endif %}

<h3>Teacher Forum</h3>
  {% for post in posts.items %}



   <article  class="content-section">

     <div class="media mb-4">
      <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + post.author.image_file) }}">
      <div class="media-body">
       <div class="article-metadata">
        <a class="mr-2" href="{{ url_for('user_posts', username=post.author.username) }}">{{ post.author.username }}</a>
        <small class="text-muted">{{ post.date_posted.strftime('%d %b %Y - %H:%M') }}</small>
       </div>
       <h2><a class="article-title" href="{{ url_for('post', post_id=post.id) }}">{{ post.title }}</a></h2>
       <p class="article-content">{{ post.content }}</p>
      </div>
     </div>

     <form method="POST" action="{{ url_for('add_comment', post_id=post.id) }}">
      {{ form.hidden_tag() }}
      <legend class="border-bottom">{{ legend }}</legend>
      {{ form.comment.label(class="form-control-label", style="font-size:14px;") }}
      <div class="media form-group">

        {% if form.comment.errors %}
         {{ form.comment(class="form-control form-control-lg is-invalid", style="height: 37px;font-size:13px;") }}
         <div class="invalid-feedback">
          {% for error in form.comment.errors %}
           <span>{{ error }}</span>
          {% endfor %}
         </div>
        {% else %}
         {{ form.comment(class="form-control form-control-lg", style="height: 37px;font-size:13px;") }}
        {% endif %}

        {{ form.submit(class="btn-sm btn-outline-info ml-2", style="font-size:13px;") }}
       </div>

     </form>
   </article>
  {% endfor %}
  {% for page_num in posts.iter_pages(left_edge=1, right_edge=1, left_current=1, right_current=2) %}
   {% if page_num %}
    {% if posts.page == page_num %}
     <a class="btn btn-info mb-4" href="{{ url_for('home', page=page_num) }}">{{ page_num }}</a>
    {% else %}
     <a class="btn btn-outline-info mb-4" href="{{ url_for('home', page=page_num) }}">{{ page_num }}</a>
    {% endif %}
```

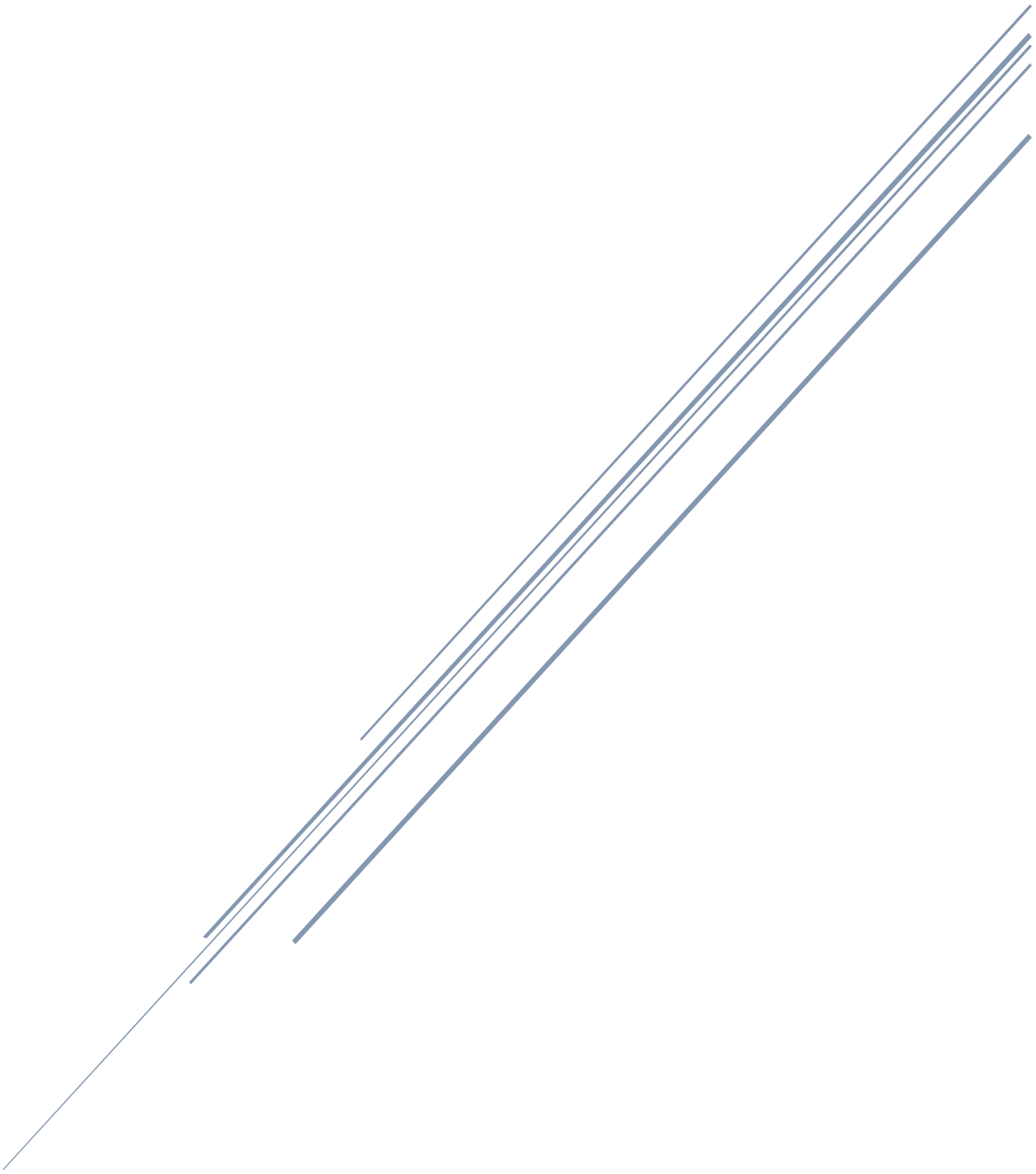
```
    {% else %}
      ...
    {% endif %}
  {% endfor %}
{% endblock content %}
```

# layout_content.html



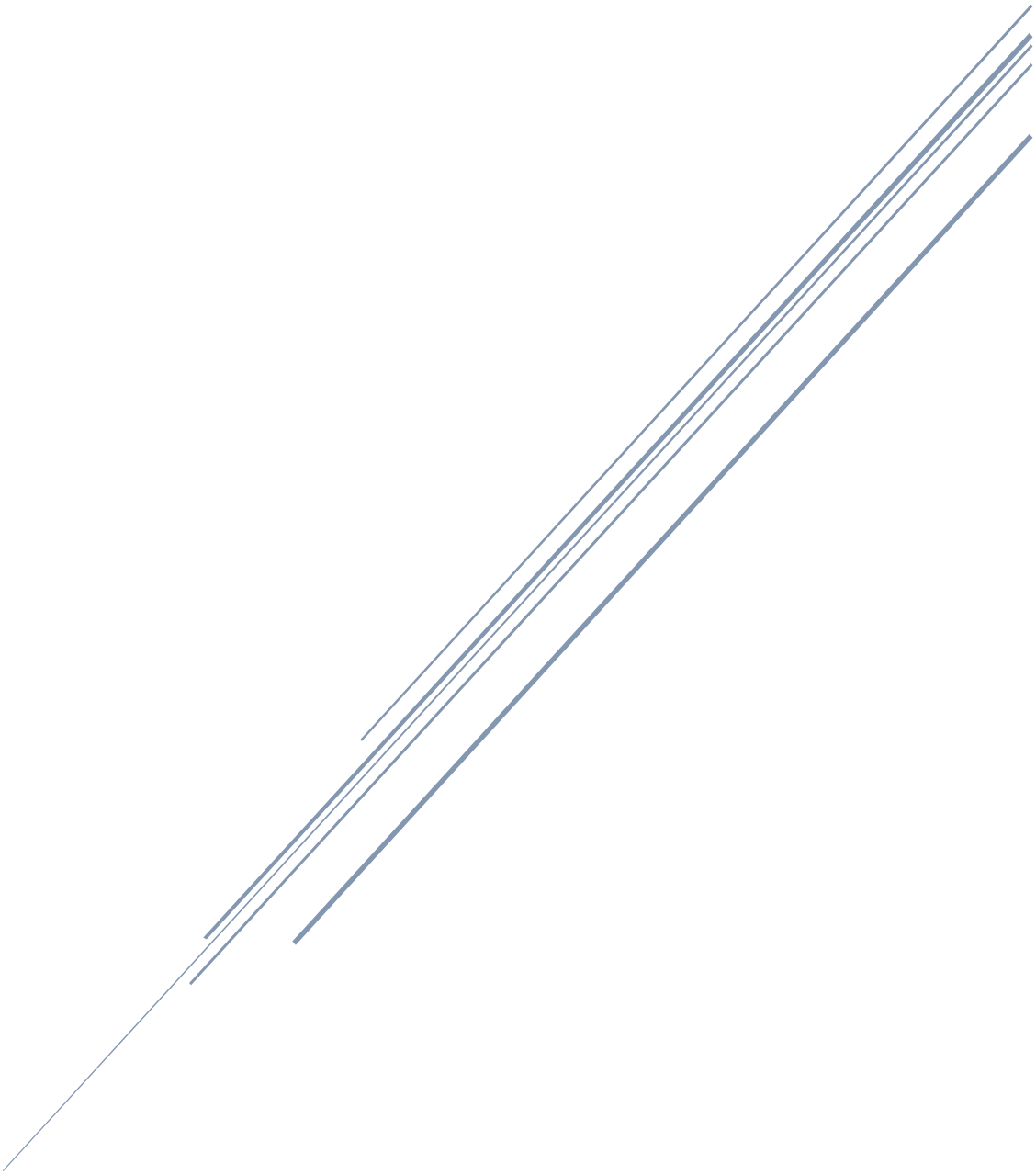
```
{% extends "layout_site_main.html" %}
{% block sublayout %}
<main role="main" class="container">
 <div class="row">
  <div class="col-md-8">
   {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
     {% for category, message in messages %}
      <div class="alert alert-{{ category }}">
       {{ message }}
      </div>
     {% endfor %}
    {% endif %}
   {% endwith %}
   {% block content %}{% endblock %}
  </div>
  <div class="col-md-4">
   <div class="content-section">
    <h3>Build 3.8.9</h3>
    <p class='text-muted'>Welcome to build 3.8.9 of <font color="black">Student</font><font color="red"><em>Track</em>
</font>! Get started by heading to the <a href="{{ url_for('courses') }}">Courses</a> page and add your first course, or make a
post on the <a href="{{ url_for('home') }}">Teachers Forum</a>.
     <!-- <ul class="list-group">
      <li class="list-group-item list-group-item-light"><a href="/register">Register as a user.</a></li>
      <li class="list-group-item list-group-item-light"><a href="/login">Login as a user.</a></li>
      <li class="list-group-item list-group-item-light"><a href="/new_class">Create a class.</a></li>
      <li class="list-group-item list-group-item-light"><a href="/new_student">Add students to a class.</a></li>
      <li class="list-group-item list-group-item-light"><a href="/account">View and edit all your, and your students, details.</a>
</li>
     </ul> -->
    </p>
   </div>
  </div>
 </div>
</main>
{% endblock sublayout %}
```

# layout_dashboard.html



```
{% extends "layout_site_main.html" %}
{% block sublayout %}
<main role="main" class="container" style="max-width: 80%;">
 <div class="row">
  <div class="col-md">
   {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
     {% for category, message in messages %}
      <div class="alert alert-{{ category }}">
       {{ message }}
      </div>
     {% endfor %}
    {% endif %}
   {% endwith %}
   {% block content %}{% endblock %}
  </div>
 </div>
</main>
{% endblock sublayout %}
```

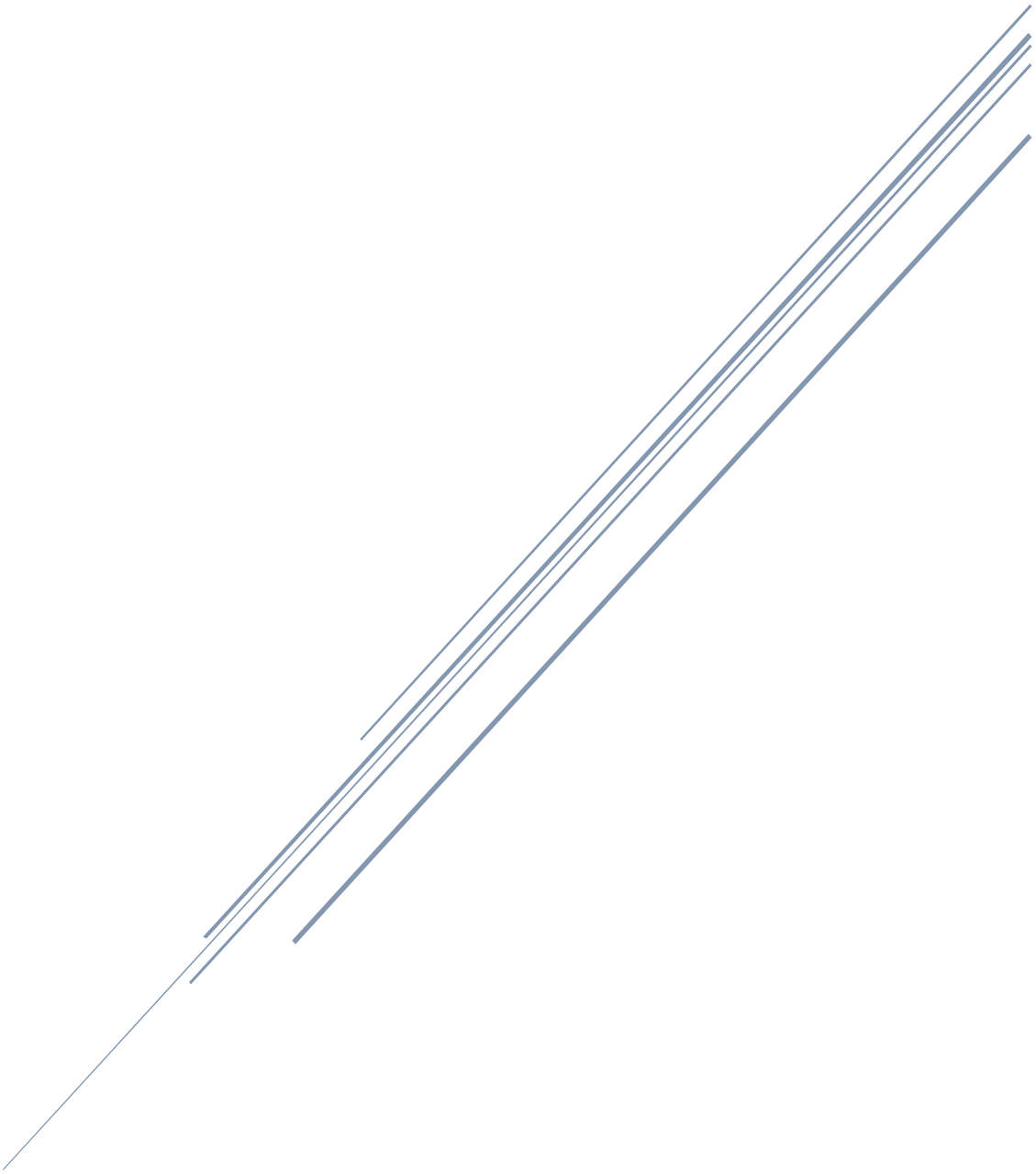
# layout_site_main.html



```
<!DOCTYPE html>
<html>
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Emergency Favicon (If script at bottom doesnt load.) -->
    <link rel="icon" type="image/png" href="https://img.icons8.com/ios/32/000000/math-book.png">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/main.css') }}">

    {% if title %}
        <title>StudentTrack - {{ title }}</title>
    {% else %}
        <title>StudentTrack</title>
    {% endif %}
</head>
<body>
    <header class="site-header">
      <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
        <div class="container">
          <a class="navbar-brand mr-4" href="/">Student<font color="red"><em>Track</em></font></a>
          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-
controls="navbarToggle" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarToggle">

            <!-- Navbar Left Side -->
            <div class="navbar-nav mr-auto">
              {% if current_user.is_authenticated %}
              <a class="nav-item nav-link" href="{{ url_for('dash') }}">Dashboard</a>
              <a class="nav-item nav-link" href="{{ url_for('courses') }}">Courses</a>
              <a class="nav-item nav-link" href="{{ url_for('topics') }}">Topics</a>
              <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown1" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                  Students
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown1">
                  <a class="dropdown-item" href="{{ url_for('students') }}">Students</a>
                  <a class="dropdown-item" href="{{ url_for('classes') }}">Classes</a>
                </div>
              </li>
              {% else %}
              <a class="nav-item nav-link" href="{{ url_for('home') }}">Forum</a>
              {% endif %}
              <a class="nav-item nav-link" href="{{ url_for('about') }}">About</a>
            </div>

            <!-- Navbar Right Side -->
            <div class="navbar-nav">
              {% if current_user.is_authenticated %}
              <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown1" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                  Teachers Forum
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown1">
                  <a class="dropdown-item" href="{{ url_for('home') }}">Go to Forum</a>
                  <a class="dropdown-item" href="{{ url_for('new_post') }}">Make a Post</a>
```

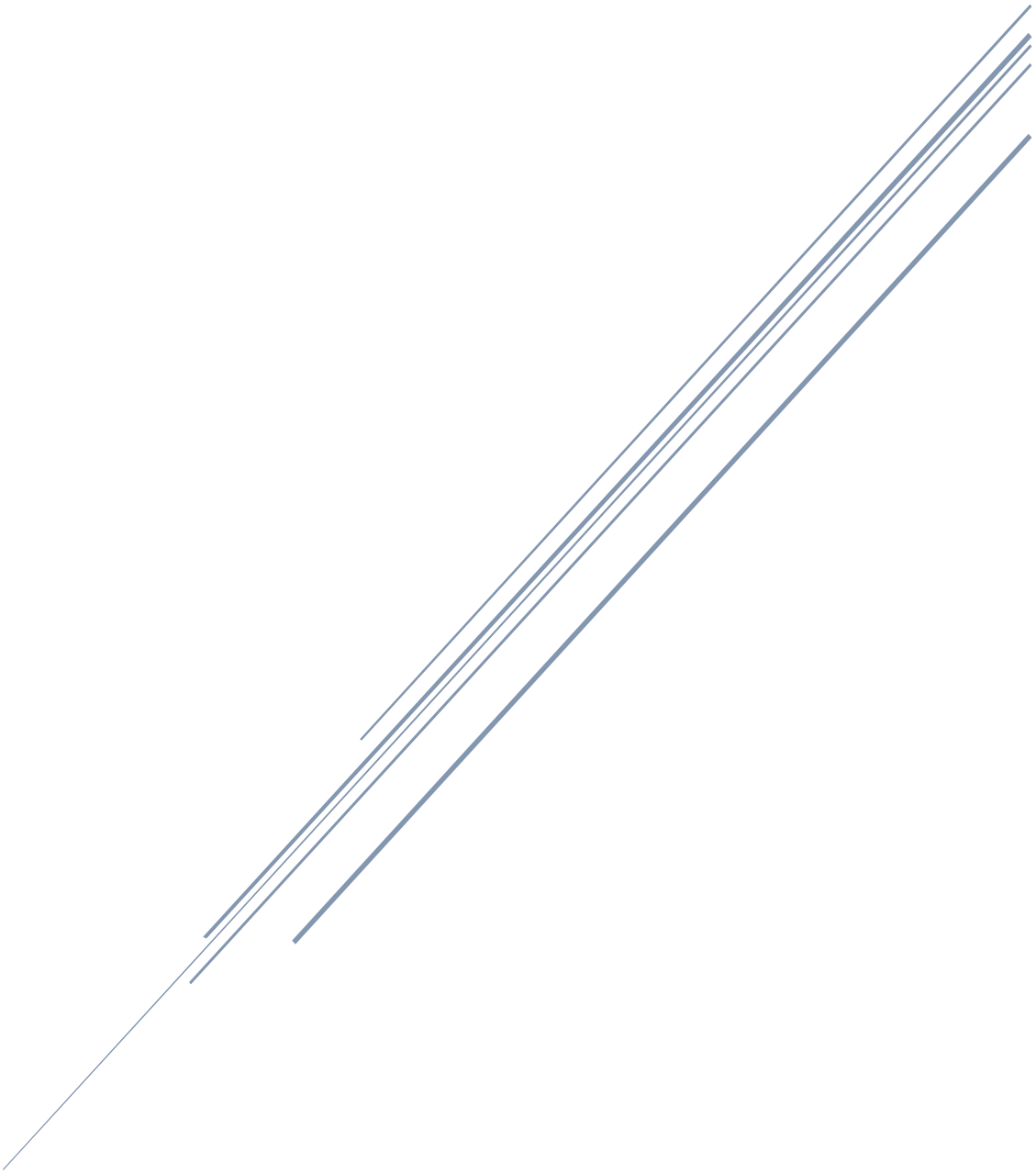
```
            </div>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle p-0" href="#" id="navbarDropdown1" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <!-- {{ current_user.username }} -->
              <img class="rounded-circle nav-img dropdown-toggle" src="{{ url_for('static', filename='profile_pics/' + current_user.image_file) }}">

            </a>
            <div class="dropdown-menu" aria-labelledby="navbarDropdown1">
              <a class="dropdown-item" href="{{ url_for('account') }}">My Account</a>
              <a class="dropdown-item" href="{{ url_for('logout') }}">Logout</a>
            </div>
          </li>
          {% else %}
          <a class="nav-item nav-link" href="{{ url_for('login') }}">Login</a>
          <a class="nav-item nav-link" href="{{ url_for('register') }}">Register</a>
          {% endif %}
        </div>
      </div>
    </div>
  </nav>
</header>
{% block sublayout %}{% endblock %}

<!-- Optional JavaScript -->
<script src='https://kit.fontawesome.com/a076d05399.js'></script>
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

<script type="text/javascript">
function applyIcon (type) {
var link = document.querySelector("link[rel*='icon']") || document.createElement('link');
link.type = 'image/x-icon';
link.rel = 'shortcut icon';

if (type === "dark") {
    link.href = "{{ url_for('static', filename='icons/favicon--dark.png') }}";
} else {
    link.href = "{{ url_for('static', filename='icons/favicon.png') }}";
}

document.getElementsByTagName('head')[0].appendChild(link);
}

var dmQuery = window.matchMedia("(prefers-color-scheme: dark)");
var lmQuery = window.matchMedia("(prefers-color-scheme: light)");

// Check on initial load if dark mode is already there. Apply the dark
// mode favicon if true.
if (dmQuery.matches) {
    applyIcon("dark");
} else {
    applyIcon("light");
}
</script>
</body>
</html>
```

# mark_exam.html



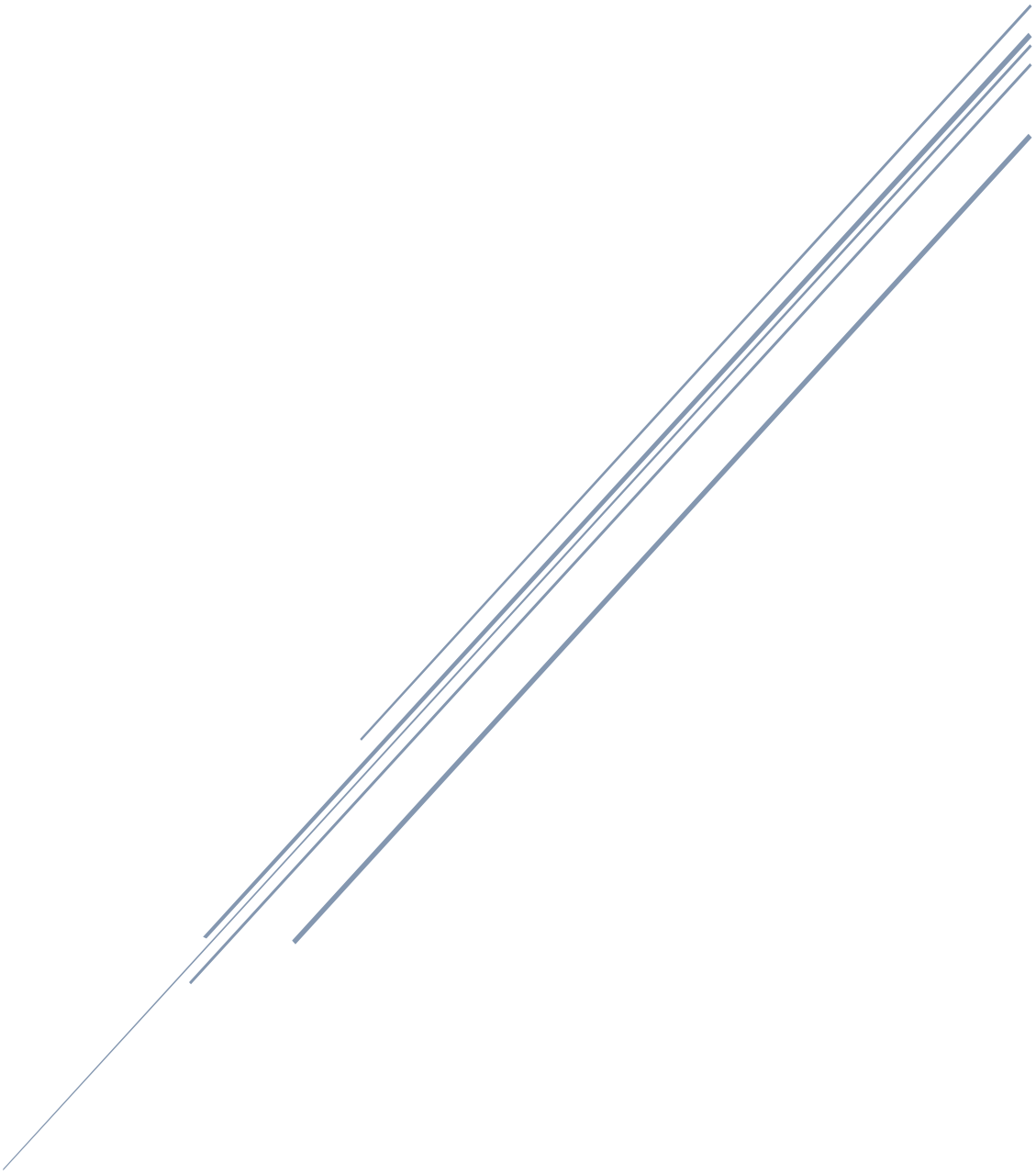
```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      {% for class in classes %}
      {% if class.students %}
      <h5 class="mb-3">{{ class.class_name }}</h5>
      {% endif %}
      {% for field in mark_fields[class.id] %}
      <div class="form-group">
        {{ field[1].label(class="form-control-label") }}
        {% if field.errors %}
          {{ field[1](class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in field.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ field[1](class="form-control form-control-lg") }}
        {% endif %}
      </div>
      {% endfor %}
      {% endfor %}
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

# mark_homework.html



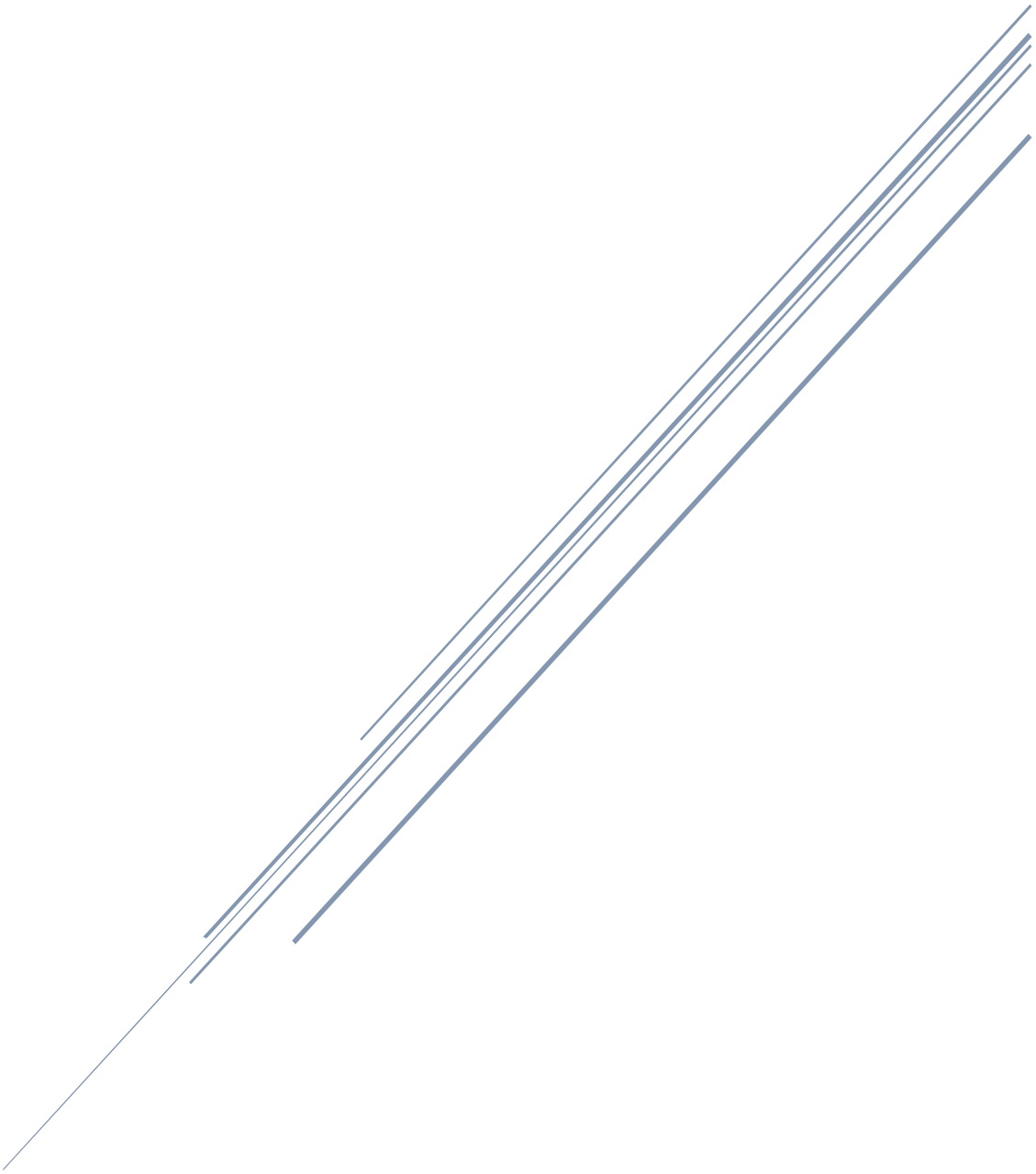
```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      {% for class in classes %}
      {% if class.students %}
      <h5 class="mb-3">{{ class.class_name }}</h5>
      {% endif %}
      {% for field in mark_fields[class.id] %}
      <div class="form-group">
        {{ field[1].label(class="form-control-label") }}
        {% if field.errors %}
          {{ field[1](class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in field.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ field[1](class="form-control form-control-lg") }}
        {% endif %}
      </div>
      {% endfor %}
      {% endfor %}
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

# mark_test.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      {% for class in classes %}
      {% if class.students %}
      <h5 class="mb-3">{{ class.class_name }}</h5>
      {% endif %}
      {% for field in mark_fields[class.id] %}
      <div class="form-group">
        {{ field[1].label(class="form-control-label") }}
        {% if field.errors %}
          {{ field[1](class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in field.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ field[1](class="form-control form-control-lg") }}
        {% endif %}
      </div>
      {% endfor %}
      {% endfor %}
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

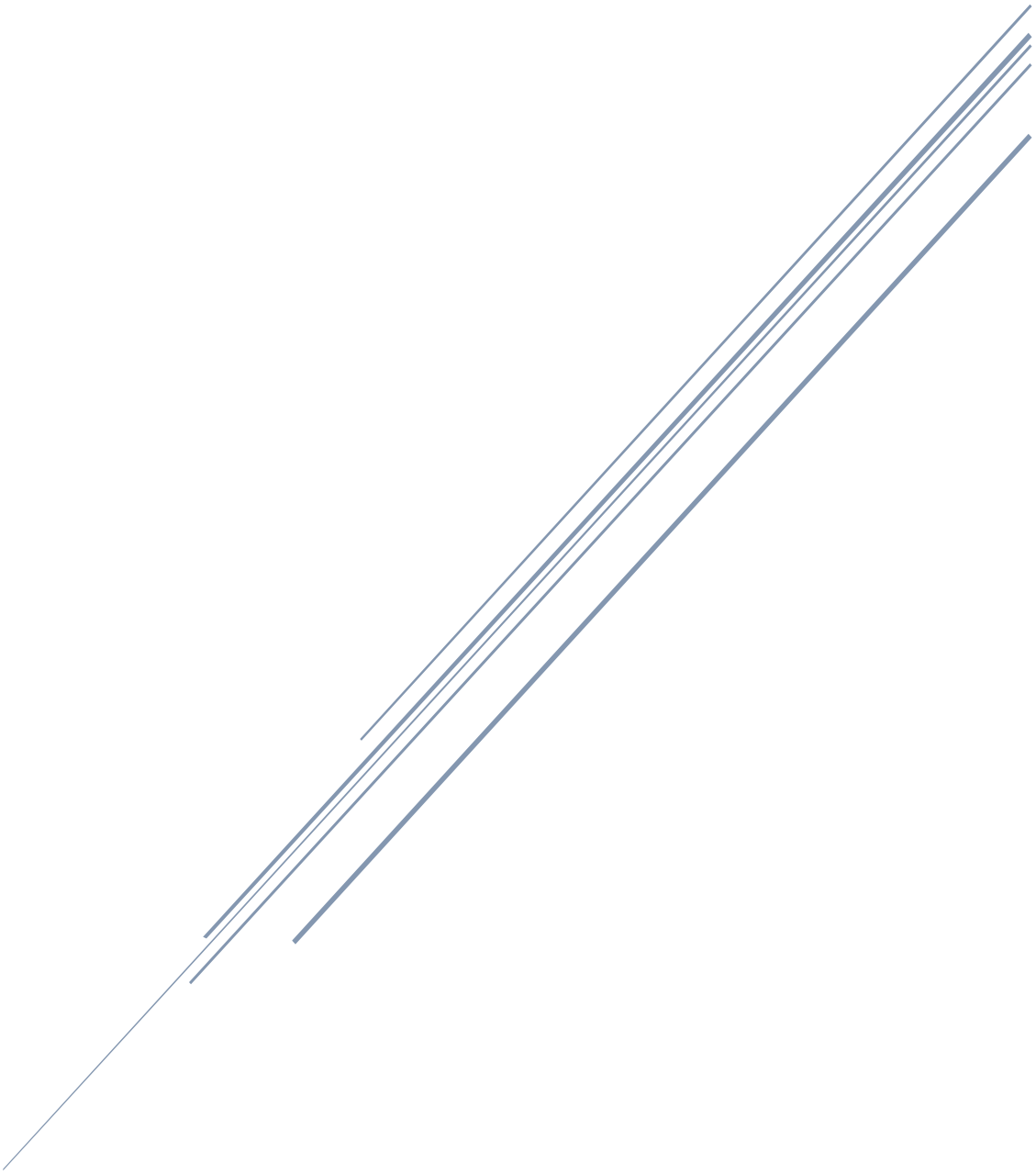
# new_class.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.class_name.label(class="form-control-label") }}
        {% if form.class_name.errors %}
          {{ form.class_name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.class_name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.class_name(class="form-control form-control-lg") }}
        {% endif %}
      </div>

      {% if form.course_id.enabled %}
      <div class="form-group">
        {{ form.course_id.label(class="form-control-label") }}
        {% if form.course_id.errors %}
          {{ form.course_id(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.course_id.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.course_id(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      {% else %}
      {{ form.course_id(class="d-none") }}
      {% endif %}

      <div class="form-group">
        {{ form.class_starting_date.label(class="form-control-label") }}
        {% if form.class_starting_date.errors %}
          {{ form.class_starting_date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.class_starting_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.class_starting_date(class="form-control form-control-lg") }}
        {% endif %}
      </div>

      <!-- Old manual date method, now replaced by wtforms html5 datefield
      <div class="form-group">
        {{ form.class_starting_date.label(class="form-control-label") }}
        {% if form.class_starting_date.errors %}
          <input class="form-control form-control-lg is-invalid" id="class_starting_date" name="class_starting_date"
type="date" value="{{ form.class_starting_date.data }}">
          <div class="invalid-feedback">
            {% for error in form.class_starting_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          <input class="form-control form-control-lg" id="class_starting_date" name="class_starting_date" type="date"
```

```
value="{{ form.class_starting_date.data }}">
                {% endif %}
            </div>-->
        </fieldset>
        <div class="form-group">
            {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
</div>
{% endblock content %}
```

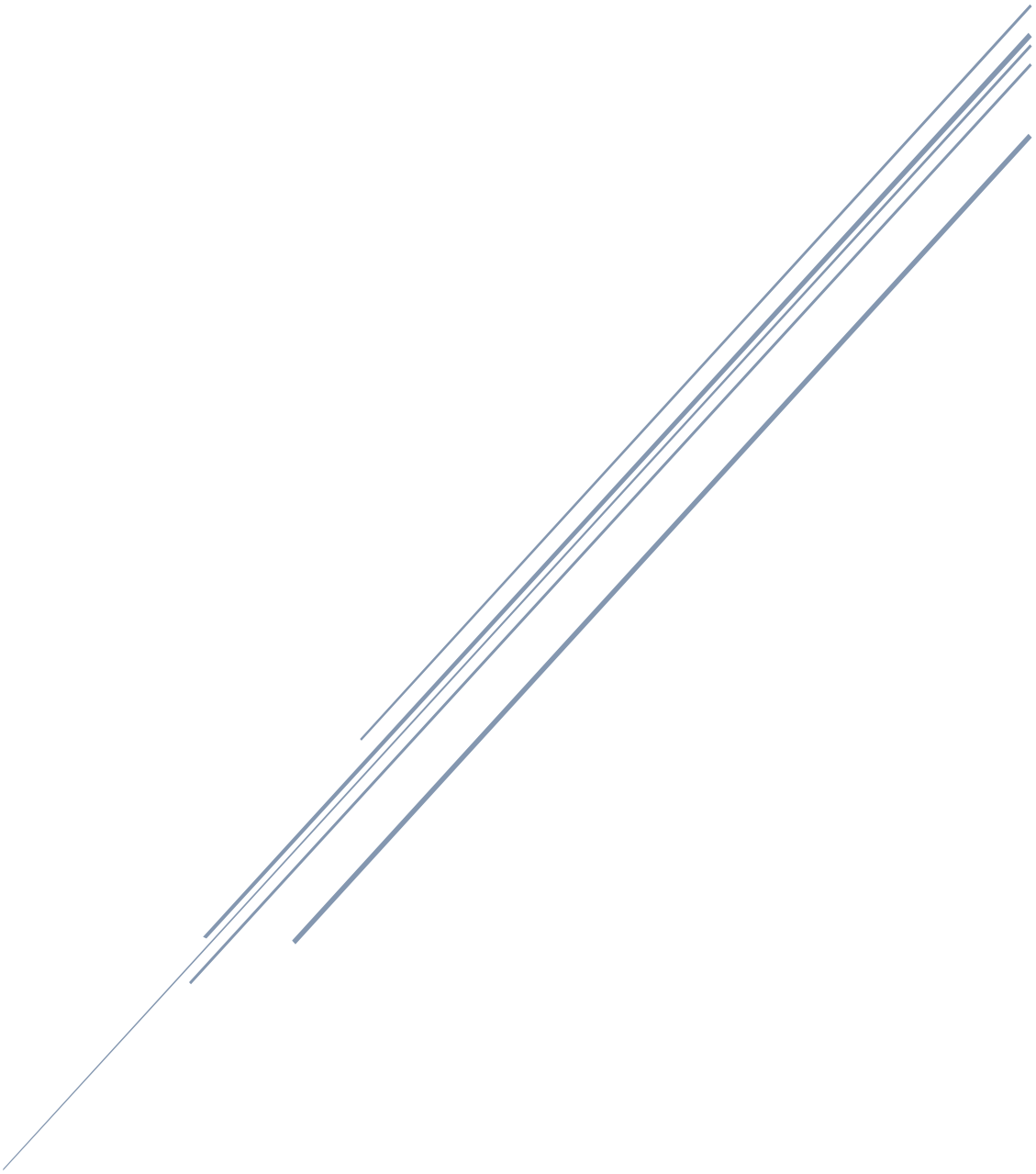
# new_course.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.start_date.label(class="form-control-label") }}
        {% if form.start_date.errors %}
          {{ form.start_date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.start_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.start_date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.year_num.label(class="form-control-label") }}
        {% if form.year_num.errors %}
          {{ form.year_num(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.year_num.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.year_num(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.grade_system.label(class="form-control-label") }}
        {% if form.grade_system.errors %}
          {{ form.grade_system(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.grade_system.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.grade_system(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
```

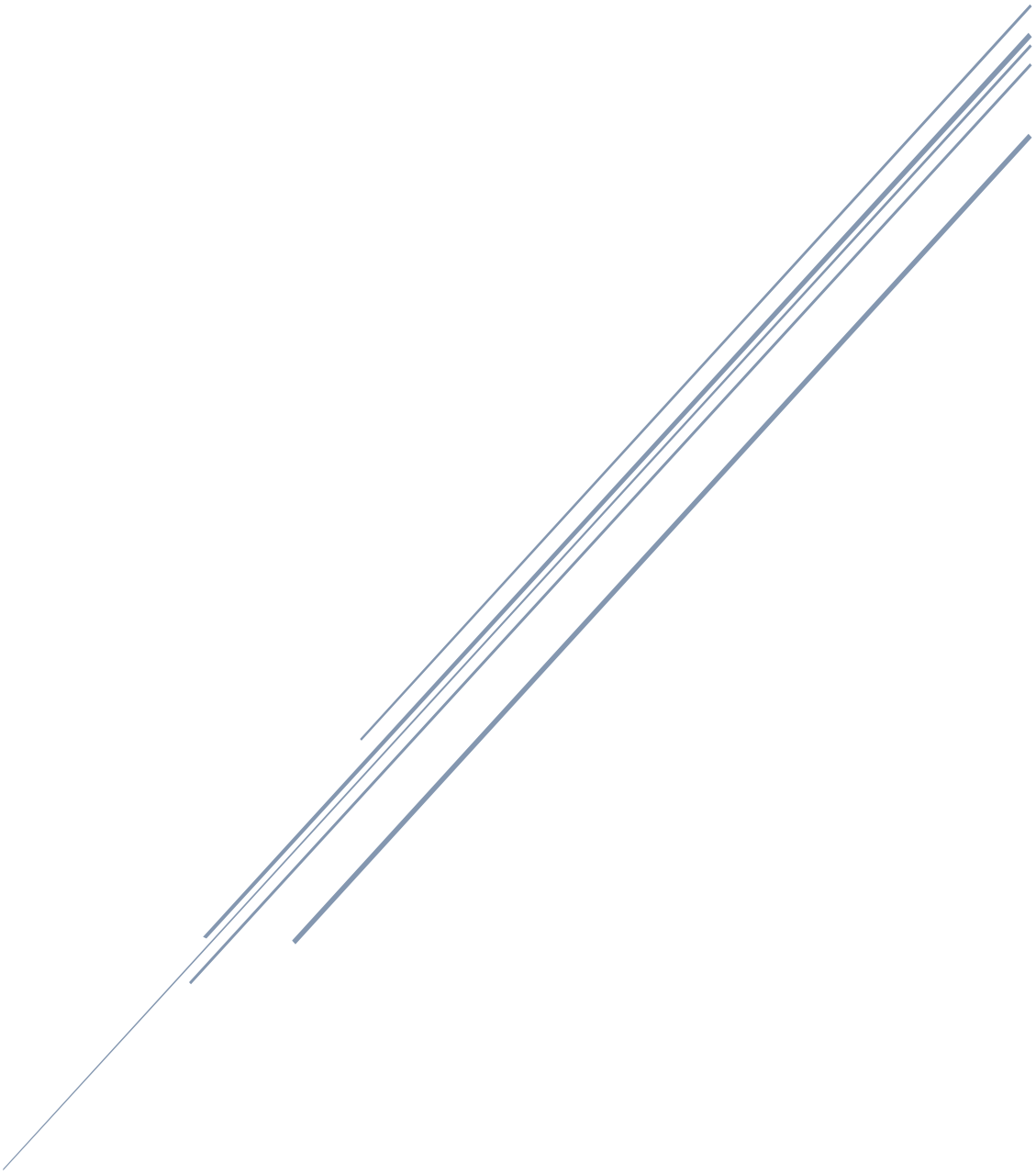
```
{% endblock content %}
```

# new_exam.html



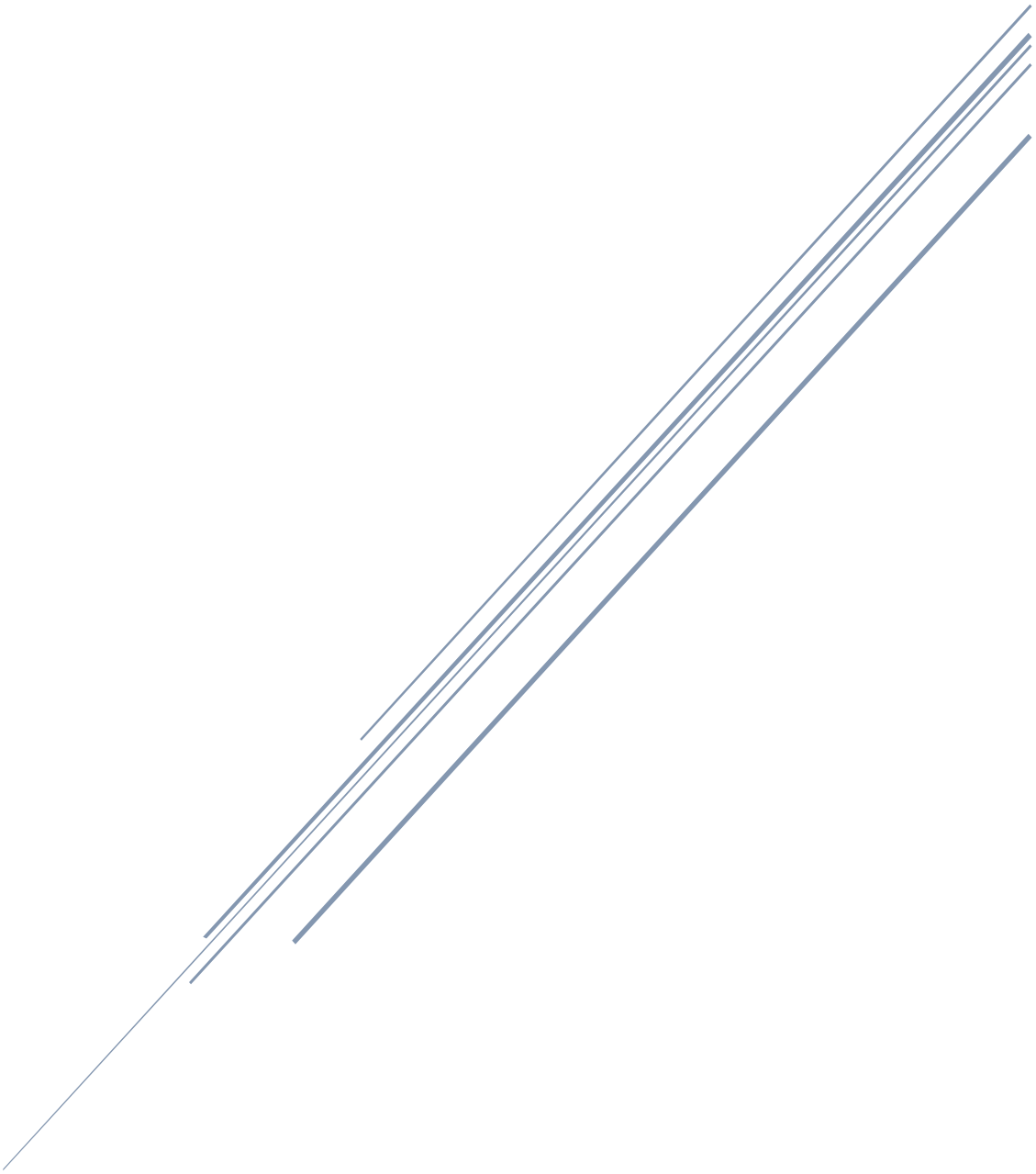
```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.max_mark.label(class="form-control-label") }}
        {% if form.max_mark.errors %}
          {{ form.max_mark(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.max_mark.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.max_mark(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.date.label(class="form-control-label") }}
        {% if form.date.errors %}
          {{ form.date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

# new_homework.html



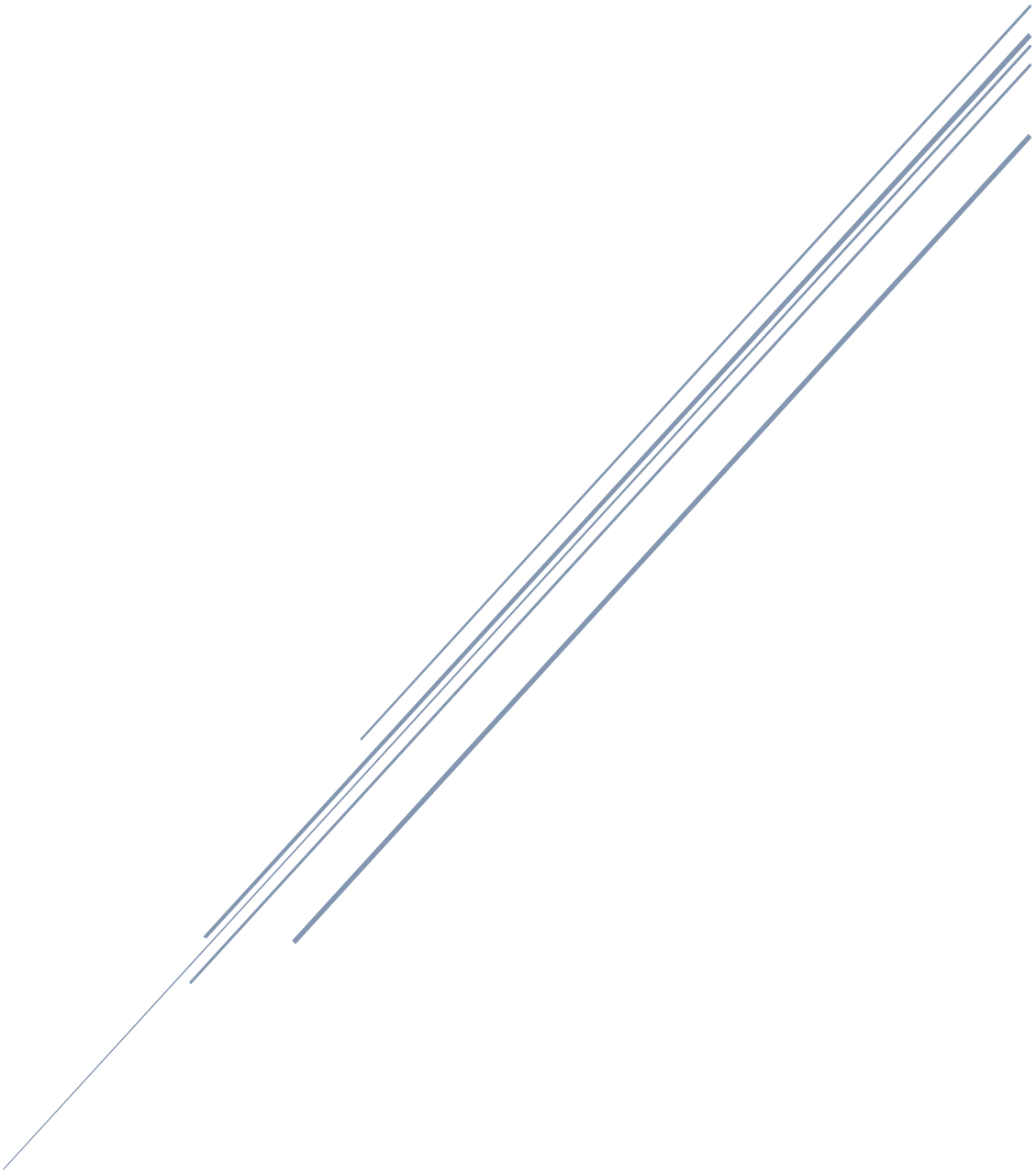
```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.max_mark.label(class="form-control-label") }}
        {% if form.max_mark.errors %}
          {{ form.max_mark(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.max_mark.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.max_mark(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.due_date.label(class="form-control-label") }}
        {% if form.due_date.errors %}
          {{ form.due_date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.due_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.due_date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

# new_post.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.title.label(class="form-control-label") }}
        {% if form.title.errors %}
          {{ form.title(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.title.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.title(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.content.label(class="form-control-label") }}
        {% if form.content.errors %}
          {{ form.content(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.content.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.content(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

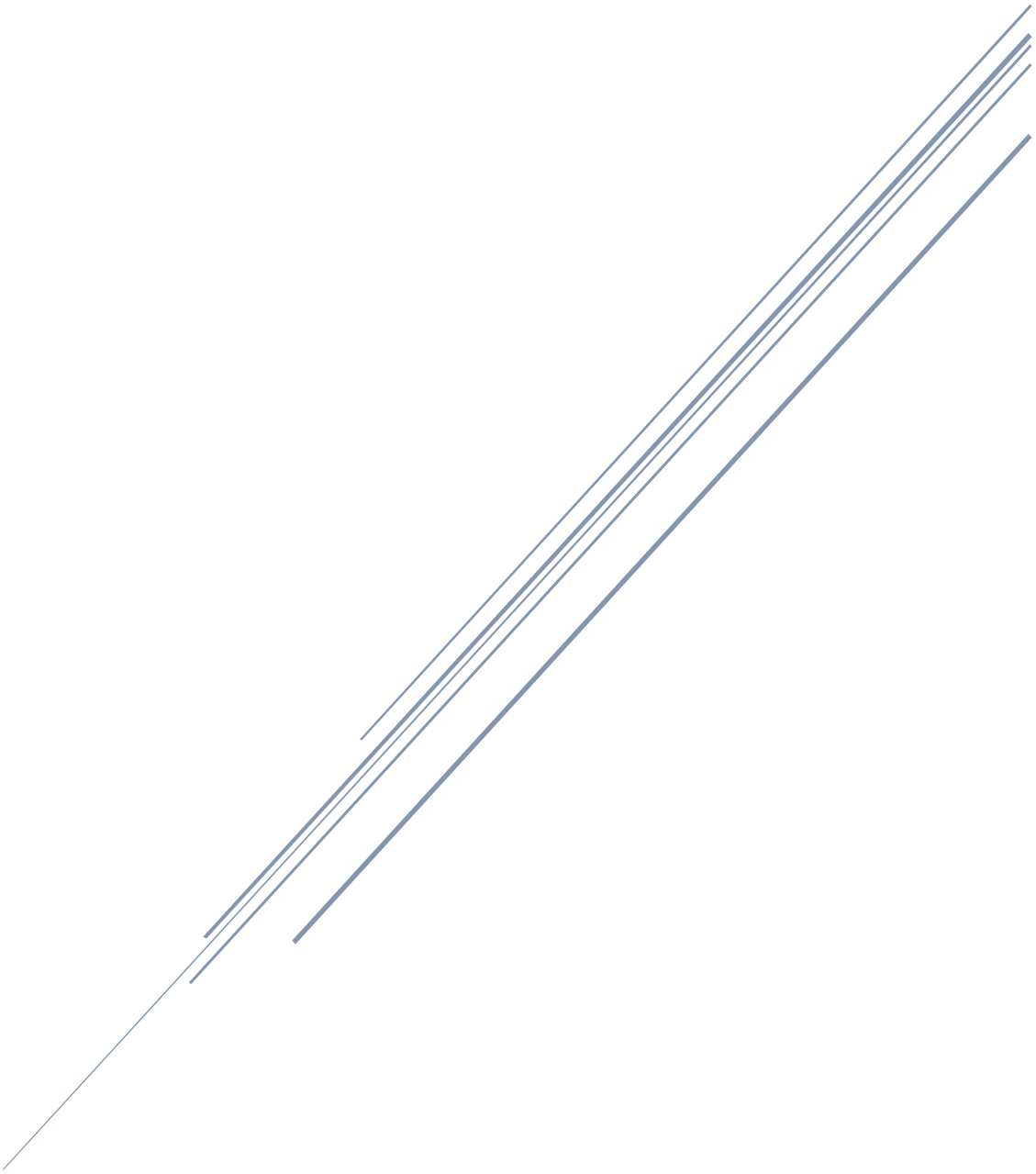
# new_student.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.class_id.label(class="form-control-label") }}
        {% if form.class_id.errors %}
          {{ form.class_id(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.class_id.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.class_id(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.email.label(class="form-control-label") }}
        {% if form.email.errors %}
          {{ form.email(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.email.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.email(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.address.label(class="form-control-label") }}
        {% if form.address.errors %}
          {{ form.address(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.address.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.address(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.parent_phone.label(class="form-control-label") }}
        {% if form.parent_phone.errors %}
          {{ form.parent_phone(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.parent_phone.errors %}
```

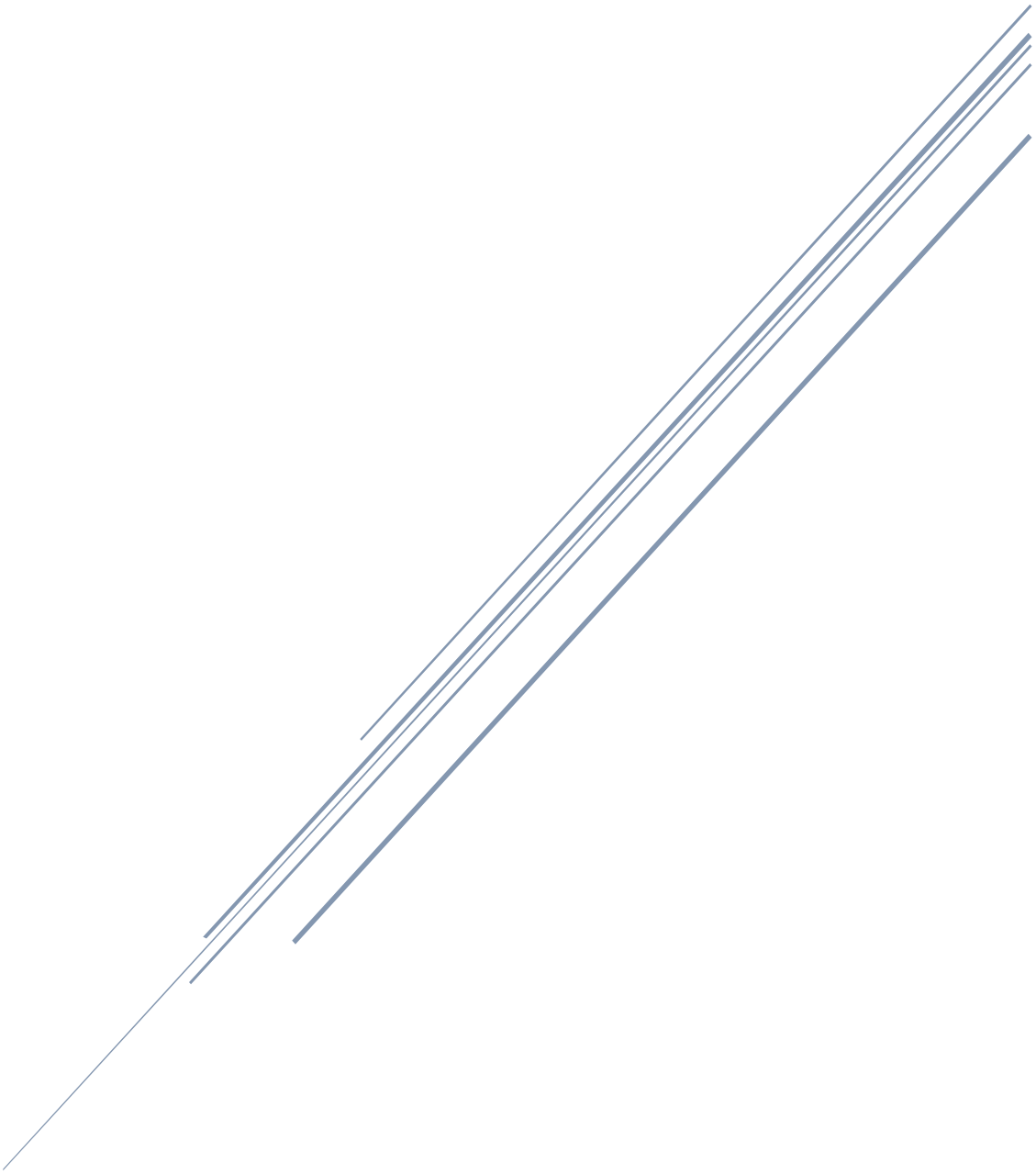
```
                    <span>{{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.parent_phone(class="form-control form-control-lg") }}
        {% endif %}
    </div>
    <div class="form-group">
        {{ form.predicted_grade.label(class="form-control-label") }}
        {% if form.predicted_grade.errors %}
            {{ form.predicted_grade(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
                {% for error in form.predicted_grade.errors %}
                    <span>{{ error }}</span>
                {% endfor %}
            </div>
        {% else %}
            {{ form.predicted_grade(class="form-control form-control-lg") }}
        {% endif %}
    </div>
  </fieldset>
  <div class="form-group">
    {{ form.submit(class="btn btn-outline-info") }}
  </div>
 </form>
</div>
{% endblock content %}
```

# new_test.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.max_mark.label(class="form-control-label") }}
        {% if form.max_mark.errors %}
          {{ form.max_mark(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.max_mark.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.max_mark(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.date.label(class="form-control-label") }}
        {% if form.date.errors %}
          {{ form.date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
    <div class="form-group">
      {{ form.submit(class="btn btn-outline-info") }}
    </div>
  </form>
</div>
{% endblock content %}
```

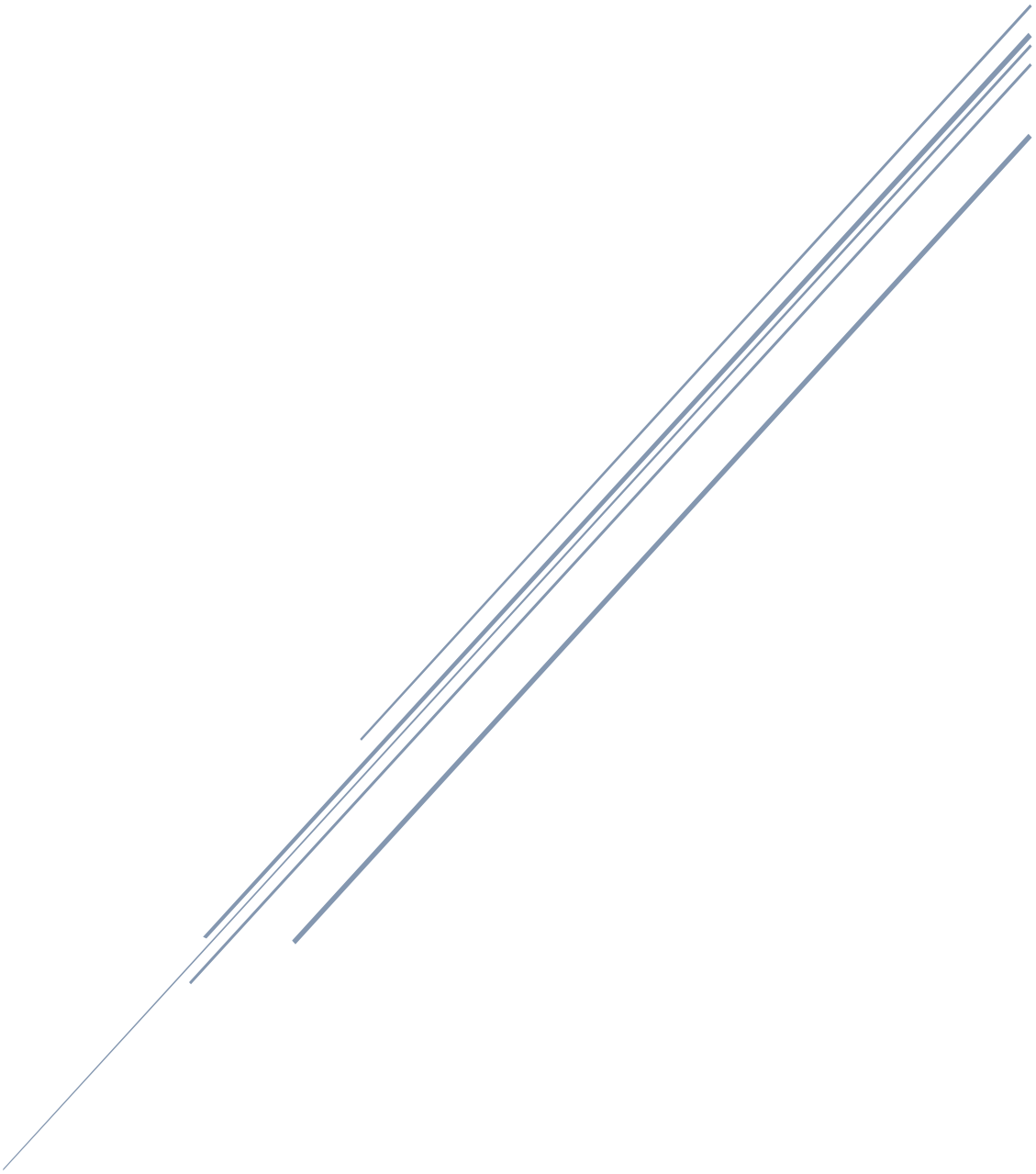
# new_topic.html



```
{% extends "layout_content.html" %}
{% block content %}
<div class="content-section">
  <form method="POST" action="">
    {{ form.hidden_tag() }}
    <fieldset class="form-group">
      <legend class="border-bottom mb-4">{{ legend }}</legend>
      <div class="form-group">
        {{ form.name.label(class="form-control-label") }}
        {% if form.name.errors %}
          {{ form.name(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.name.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.name(class="form-control form-control-lg") }}
        {% endif %}
      </div>

      {% if form.course_id.enabled %}
      <div class="form-group">
        {{ form.course_id.label(class="form-control-label") }}
        {% if form.course_id.errors %}
          {{ form.course_id(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.course_id.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.course_id(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      {% else %}
      {{ form.course_id(class="d-none") }}
      {% endif %}

      <div class="form-group">
        {{ form.begin_date.label(class="form-control-label") }}
        {% if form.begin_date.errors %}
          {{ form.begin_date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.begin_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.begin_date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
      <div class="form-group">
        {{ form.end_date.label(class="form-control-label") }}
        {% if form.end_date.errors %}
          {{ form.end_date(class="form-control form-control-lg is-invalid") }}
          <div class="invalid-feedback">
            {% for error in form.end_date.errors %}
              <span>{{ error }}</span>
            {% endfor %}
          </div>
        {% else %}
          {{ form.end_date(class="form-control form-control-lg") }}
        {% endif %}
      </div>
    </fieldset>
```

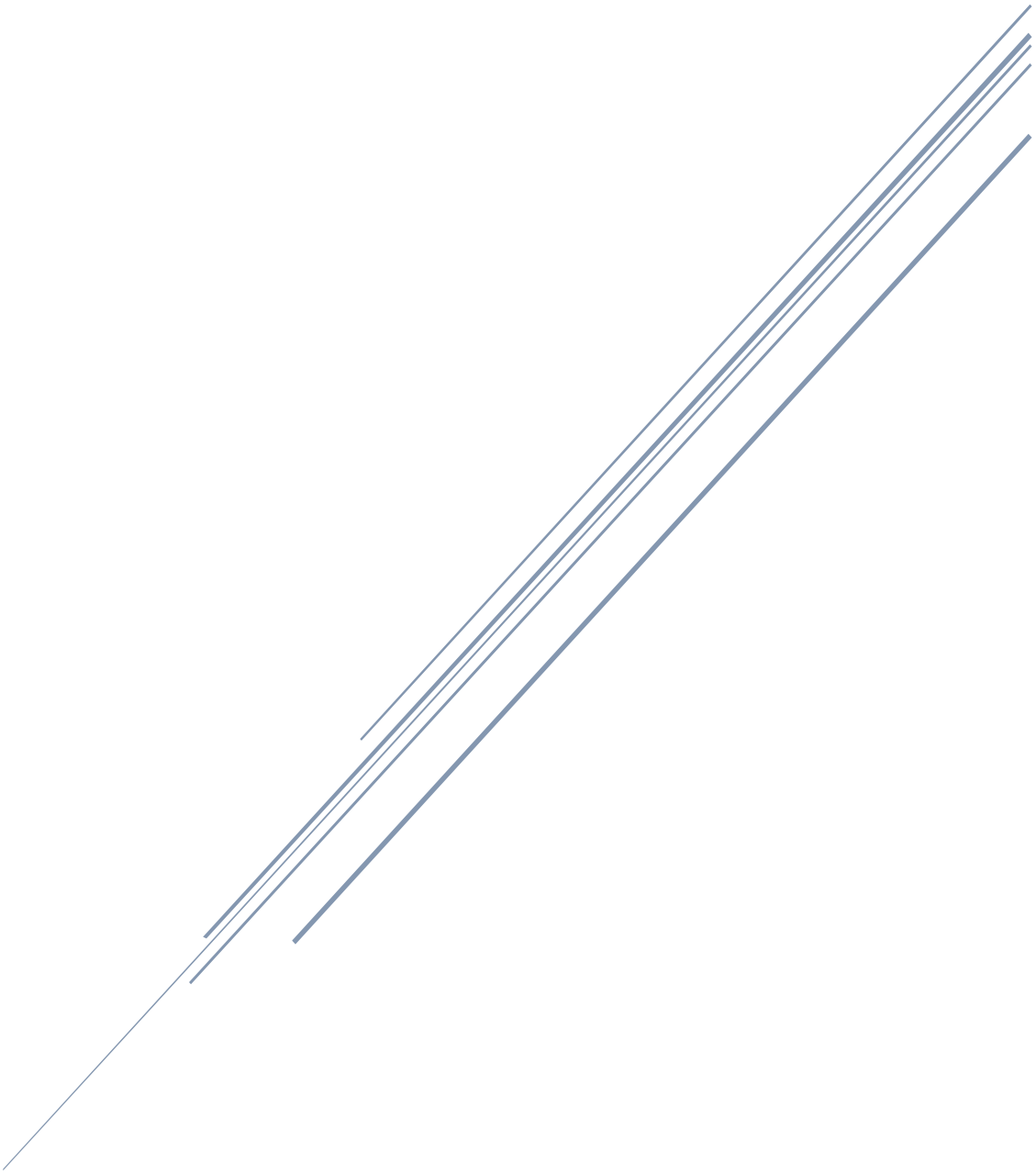
```
        <div class="form-group">
            {{ form.submit(class="btn btn-outline-info") }}
        </div>
    </form>
</div>
{% endblock content %}
```

# reset_request.html



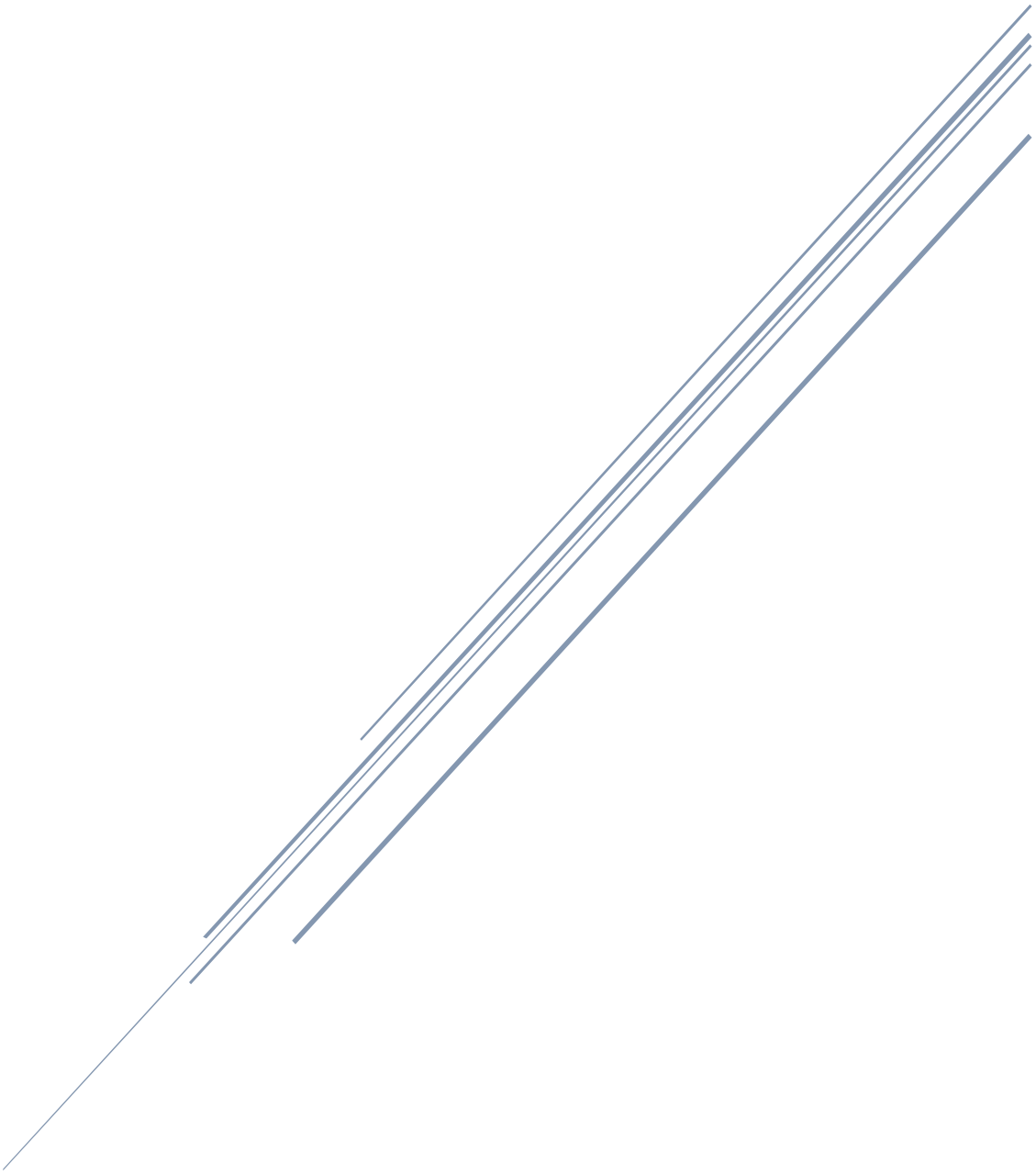
```
{% extends "layout_content.html" %}
{% block content %}
  <div class="content-section">
    <form method="POST" action="">
      {{ form.hidden_tag() }}
      <fieldset class="form-group">
        <legend class="border-bottom mb-4">Reset Password</legend>
        <div class="form-group">
          {{ form.email.label(class="form-control-label") }}
          {% if form.email.errors %}
            {{ form.email(class="form-control form-control-lg is-invalid") }}
            <div class="invalid-feedback">
              {% for error in form.email.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.email(class="form-control form-control-lg") }}
          {% endif %}
        </div>
      </fieldset>
      <div class="form-group">
        {{ form.submit(class="btn btn-outline-info") }}
      </div>
    </form>
  </div>
{% endblock content %}
```

# reset_token.html



```
{% extends "layout_content.html" %}
{% block content %}
    <div class="content-section">
      <form method="POST" action="">
          {{ form.hidden_tag() }}
          <fieldset class="form-group">
              <legend class="border-bottom mb-4">Reset Password</legend>
              <div class="form-group">
                  {{ form.password.label(class="form-control-label") }}
                  {% if form.password.errors %}
                      {{ form.password(class="form-control form-control-lg is-invalid") }}
                      <div class="invalid-feedback">
                          {% for error in form.password.errors %}
                              <span>{{ error }}</span>
                          {% endfor %}
                      </div>
                  {% else %}
                      {{ form.password(class="form-control form-control-lg") }}
                  {% endif %}
              </div>
              <div class="form-group">
                  {{ form.confirm_password.label(class="form-control-label") }}
                  {% if form.confirm_password.errors %}
                      {{ form.confirm_password(class="form-control form-control-lg is-invalid") }}
                      <div class="invalid-feedback">
                          {% for error in form.confirm_password.errors %}
                              <span>{{ error }}</span>
                          {% endfor %}
                      </div>
                  {% else %}
                      {{ form.confirm_password(class="form-control form-control-lg") }}
                  {% endif %}
              </div>
          </fieldset>
          <div class="form-group">
              {{ form.submit(class="btn btn-outline-info") }}
          </div>
      </form>
    </div>
{% endblock content %}
```

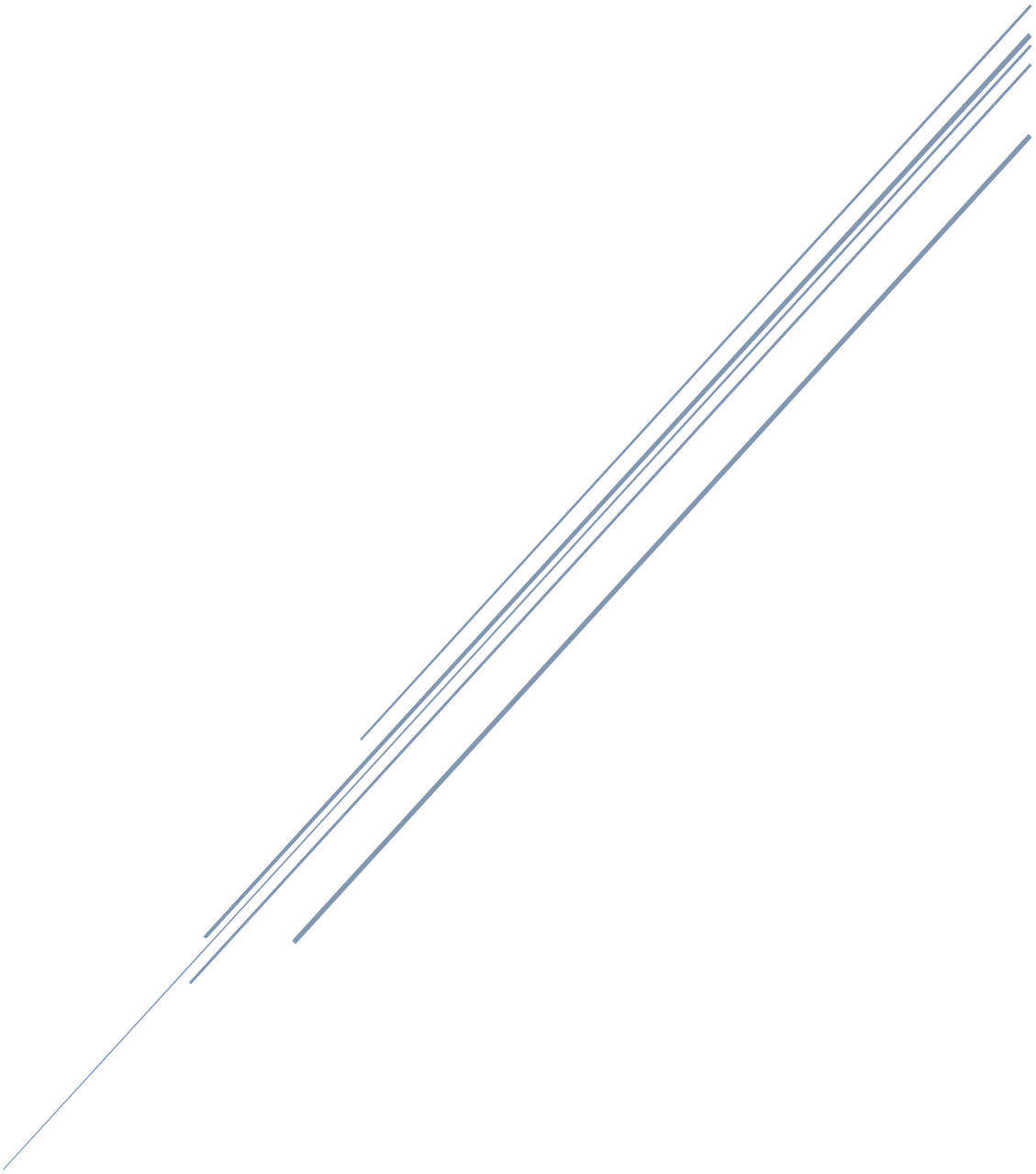
# single_class.html



```
{% extends "layout_content.html" %}
{% block content %}
  <article class="media content-section parent">

    <div class="ribbon"><a style="color:white;" href="{{ url_for('courses', course_id=xclass.course.id) }}">{{ xclass.course.name
}}</a></div>

    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + xclass.course.teacher.image_file) }}">
    <div class="media-body">
      <div class="article-metadata">
        <small class="text-muted mr-2">Beginning: {{ xclass.class_starting_date.strftime('%d %b %Y') }}</small>
      </div>
      <h2><a class="article-title" href="{{ url_for('xclass', class_id=xclass.id) }}">{{ xclass.class_name }}</a></h2>

      <h4>Performance:</h4>

      <div id="graph-container">
        <canvas id="performanceGraph"></canvas>
      </div>


      <h4>Students:</h4>

      {% for student in xclass.students %}
       <div class="pb-2"><b>â˜</b><li class="d-inline m-3">{{ student.name }}</li><a class="d-inline btn-danger btn-sm"
style="float:right;" href="{{ url_for('remove_from_class', student_id=student.id) }}">Remove</a></div>
      {% endfor %}

      <div class="mt-3 mb-0">
        <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('class_report', class_id=xclass.id) }}">Get Report</a>
        <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_class', class_id=xclass.id) }}">Update Details</a>
        <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-
target="#deleteModal">Delete</button>
      </div>
    </div>
  </article>
  <!-- Modal -->
  <div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="deleteModalLabel">Delete Class?</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
          <form action="{{ url_for('delete_class', class_id=xclass.id) }}" method="POST">
            <input class="btn btn-danger" type="submit" value="Delete">
          </form>
        </div>
      </div>
    </div>
  </div>


  <!-- Linking all the scripts! -->
  <script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
  <script type="text/javascript" src="{{ url_for('static', filename='js/dashboard_charts.js') }}"></script>
  <script type="text/javascript">
```

```
    // Setting graph to load onload.
    $(document).ready(function() {
      changeChart("performanceGraph", "line",
                {{ class_perf.labels | safe }},
                {{ class_perf.datasets | safe }});
    });
  </script>
{% endblock content %}
```

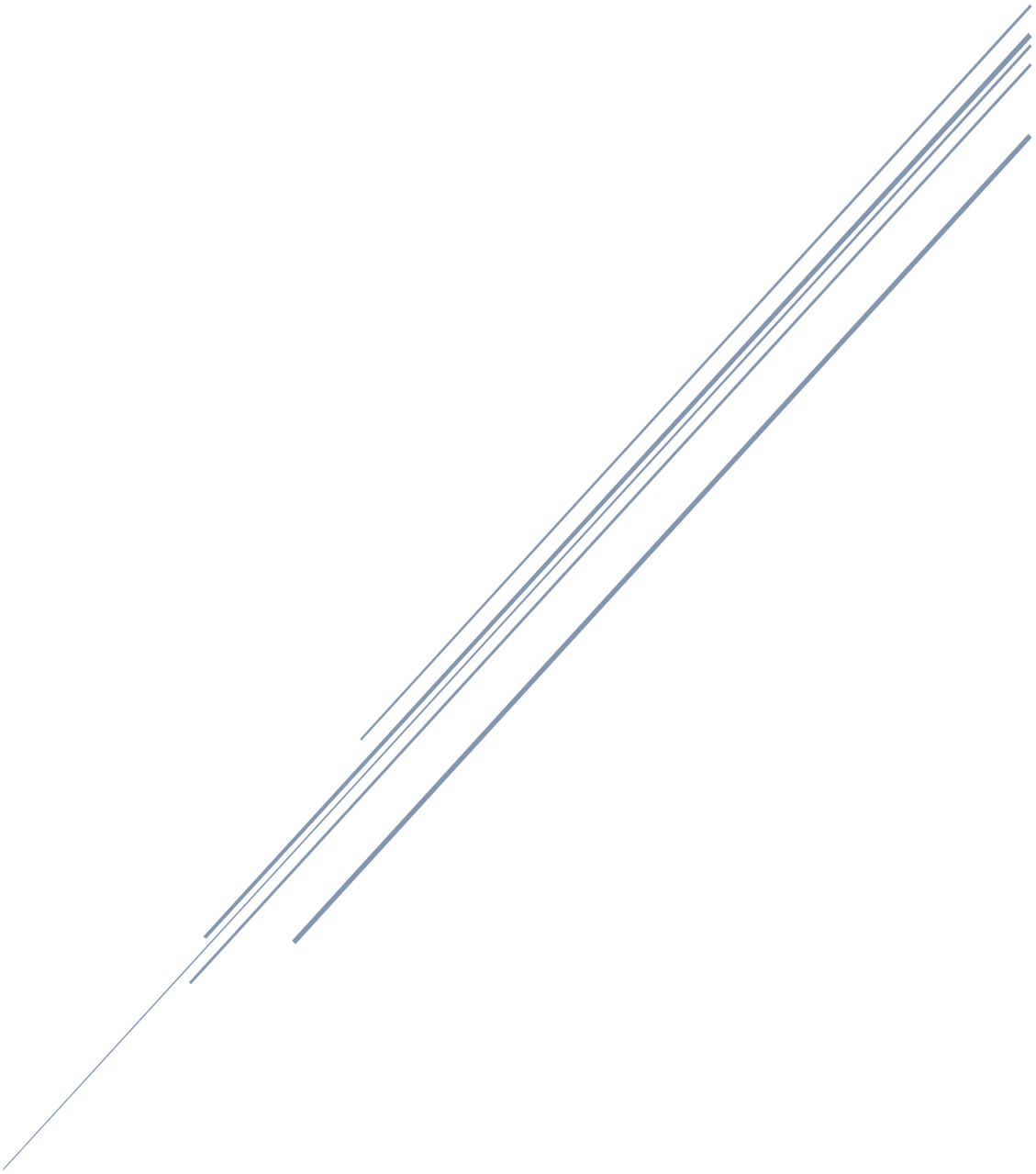
# single_homework.html



```
{% extends "layout_content.html" %}
{% block content %}
  <article class="parent media content-section">

    <div class="ribbon"><a style="color:white;" href="{{ url_for('courses', course_id=homework.topic.course.id) }}">{{
homework.topic.course.name }}</a></div>

    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/def.png') }}">
    <div class="media-body">
      <h2><a class="article-title" href="{{ url_for('homework', homework_id=homework.id) }}">{{ homework.name }}</a></h2>


      <h4 class="m-0 p-0 mb-2">Classes Assigned To</h4>
  <p class="m-0 p-0 mb-3">
      {% for class in homework.topic.course.classes %}
      - {{ class.class_name }}<br>
      {% endfor %}

      </p>

      <h4 class="mt-2">Recorded Performance</h4>

      {% if class_perf.datasets %}

      <div id="graph-container">
        <canvas id="performanceGraph"></canvas>
      </div>


      {% else %}
      <p>No data to graph.</p>
      {% endif %}

      <div class="mt-3 mb-0">
        <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_homework', homework_id=homework.id)
}}">Update Details</a>
        <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="">Add To Class</a> -->
        <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="">Remove From Class</a> -->
        <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-
target="#deleteModal">Delete</button>
      </div>
    </div>
  </article>
  <!-- Modal -->
  <div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="deleteModalLabel">Delete Class?</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&amp;times;</span>
          </button>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
          <form action="{{ url_for('delete_homework', homework_id=homework.id) }}" method="POST">
            <input class="btn btn-danger" type="submit" value="Delete">
          </form>
        </div>
      </div>
    </div>
  </div>
```

```
  <script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
  <script type="text/javascript" src="{{ url_for('static', filename='js/dashboard_charts.js') }}"></script>
  <script type="text/javascript">

    // Setting graph to load onload.
    $(document).ready(function() {
      changeChart("performanceGraph", "bar",
                {{ class_perf.labels | safe }},
                {{ class_perf.datasets | safe }});
    });
    </script>
{% endblock content %}
```

# single_post.html

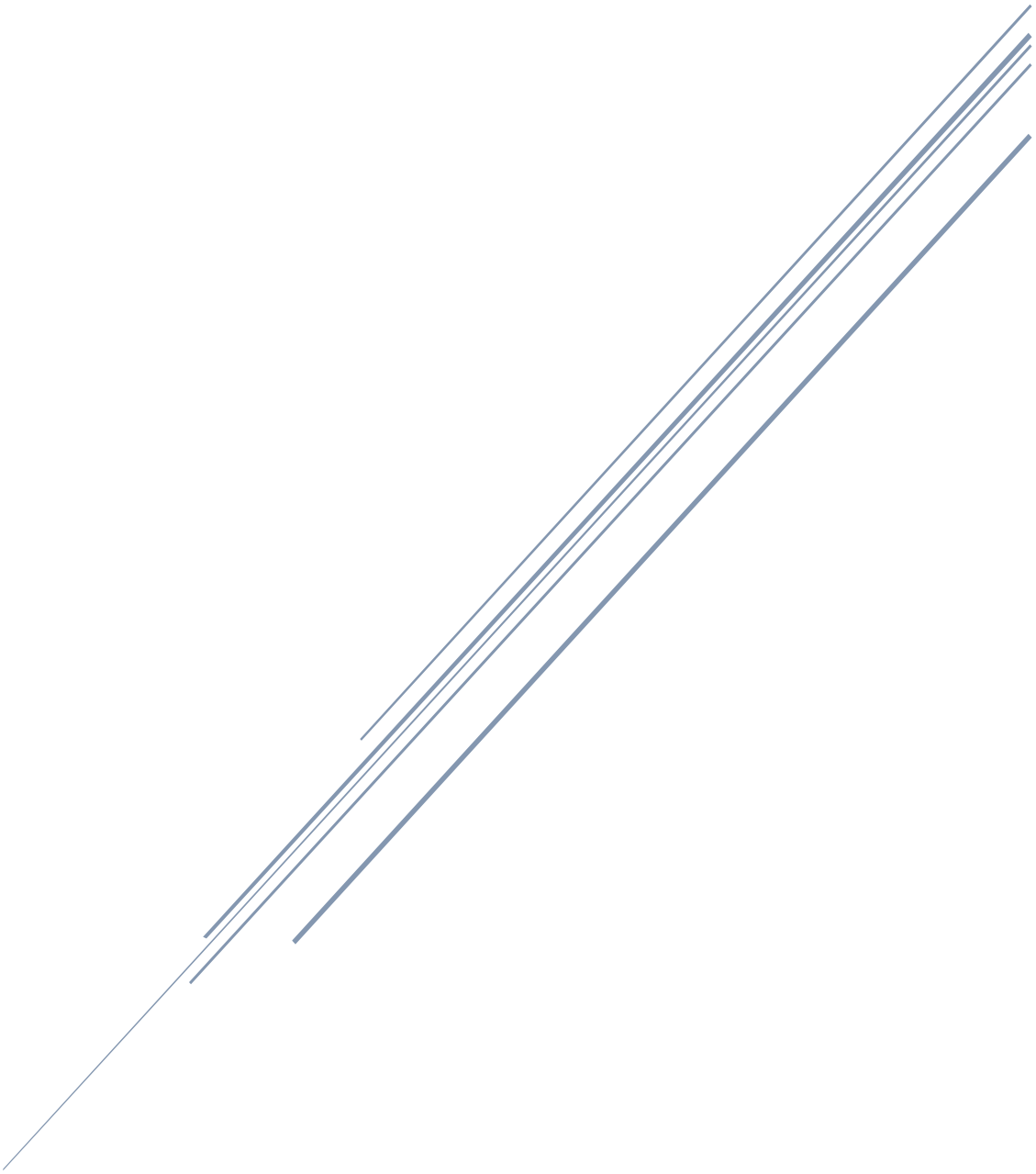


```
{% extends "layout_content.html" %}
{% block content %}
  <article class="content-section">

    <div class="media mb-4">

    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + post.author.image_file) }}">
    <div class="media-body">
      <div class="article-metadata">
        <a class="mr-2" href="{{ url_for('user_posts', username=post.author.username) }}">{{ post.author.username }}</a>
        <small class="text-muted">{{ post.date_posted.strftime('%d %b %Y - %H:%M') }}</small>
        {% if post.author == current_user %}
          <div class="mt-1 mb-1">
            <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_post', post_id=post.id) }}">Update</a>
            <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-
target="#deleteModal">Delete</button>
          </div>
        {% endif %}
      </div>
      <h2 class="article-title">{{ post.title }}</h2>
      <p class="article-content">{{ post.content }}</p>
    </div>

  </div>

  <form method="POST" action="{{ url_for('add_comment', post_id=post.id) }}">
    {{ form.hidden_tag() }}
    <legend class="border-bottom">{{ legend }}</legend>
    {{ form.comment.label(class="form-control-label", style="font-size:14px;") }}
    <div class="media form-group">

      {% if form.comment.errors %}
        {{ form.comment(class="form-control form-control-lg is-invalid", style="height: 37px;font-size:13px;") }}
        <div class="invalid-feedback">
          {% for error in form.comment.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
      {% else %}
        {{ form.comment(class="form-control form-control-lg", style="height: 37px;font-size:13px;") }}
      {% endif %}

      {{ form.submit(class="btn-sm btn-outline-info ml-2", style="font-size:13px;") }}
    </div>

  </form>

  {% for comment in post.comments %}
  <div class="media mb-4">
    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/' + comment.commenter.image_file) }}">
    <div class="media-body" style="float:left;">
      <div class="article-metadata">
        <a class="mr-2" href="{{ url_for('user_posts', username=comment.commenter.username) }}">{{
comment.commenter.username }}</a>
        <small class="text-muted">{{ comment.date_posted.strftime('%d %b %Y - %H:%M') }}</small>

      </div>
      <p id="CE{{ comment.id }}" class="article-content" contenteditable="true">{{ comment.content }}</p>
    </div>
    {% if comment.commenter == current_user %}
      <div class="ml-2 mt-4" style="float:right;">
        <a class="btn btn-secondary btn-sm" style="color:white;" onClick="updateComment({{ comment.id }})">Update</a>
        <a class="btn btn-danger btn-sm" href="{{ url_for('delete_comment', post_id=post.id, comment_id=comment.id)
}}">Delete</a>
      </div>
```

```
    {% endif %}
  </div>
  {% endfor %}




  </article>
  <!-- Modal -->
  <div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="deleteModalLabel">Delete Post?</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
          <form action="{{ url_for('delete_post', post_id=post.id) }}" method="POST">
            <input class="btn btn-danger" type="submit" value="Delete">
          </form>
        </div>
      </div>
    </div>
  </div>

<script type="text/javascript">
function updateComment(id) {
  var editableDiv =  document.getElementById("CE" + id);
  var comment = editableDiv.innerText;

  const Http = new XMLHttpRequest();
  const url= "/comment/" + id + "/update";

  Http.open("POST", url);
  Http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  Http.send("comment="+comment);

  Http.onreadystatechange = (e) => {
    if (Http.status == 201 && Http.responseText && !document.getElementById("SE" + id)) {

      if (Http.responseText[0] == "P"){
        errorMsg = document.createElement("div");
        errorMsg.className = "alert alert-danger";
        errorMsg.id = "SE" + id;
        var textnode = document.createTextNode(Http.responseText);
        errorMsg.appendChild(textnode);
        editableDiv.parentNode.prepend(errorMsg);
      } else {
        successMsg = document.createElement("div");
        successMsg.className = "alert alert-success";
        successMsg.id = "SE" + id;
        var textnode = document.createTextNode(Http.responseText);
        successMsg.appendChild(textnode);
        editableDiv.parentNode.prepend(successMsg);
      }
    }
  }
  return 0;
}

</script>
```

```
{% endblock content %}
```

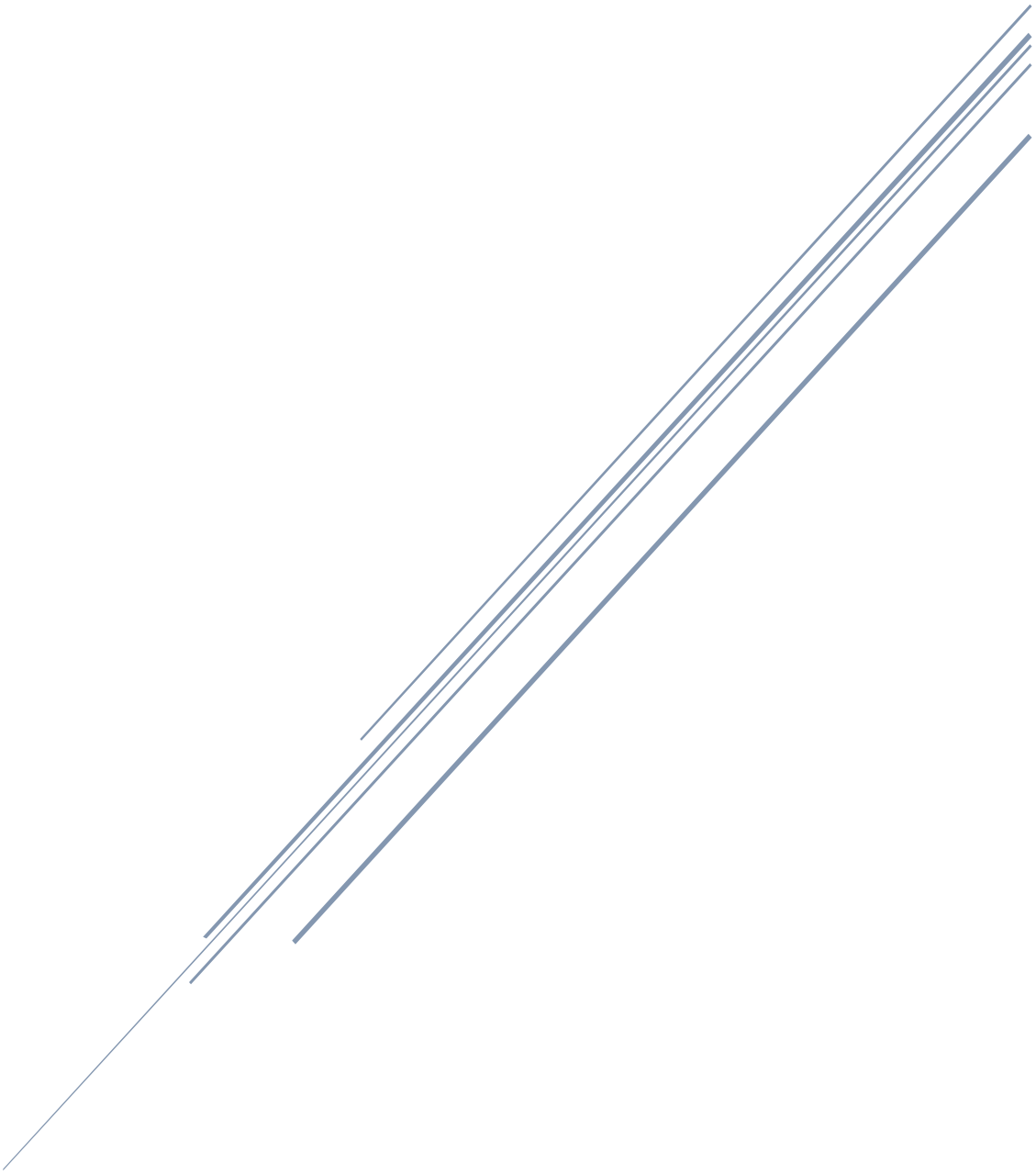
# single_student.html



```
{% extends "layout_content.html" %}
{% block content %}
 <article class="media content-section">
  <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/def.png') }}">
  <div class="media-body">
   <h2><a class="article-title" href="{{ url_for('student', student_id=student.id) }}">{{ student.name }}</a></h2>

   <h5>Student's Performance:</h5>
   <div id="graph-container">
    <canvas id="performanceGraph"></canvas>
   </div>


   <h5>Student's Information:</h5>

   <hr class="mb-0">
   <p class="mb-0"><strong>Name:</strong> {{ student.name }}</p>
   <p class="mb-0"><strong>Email:</strong> {{ student.email }}</p>
   <p class="mb-0"><strong>Address:</strong><pre>{{ student.address }}</pre>
   <p class="mb-0"><strong>Parent's Phone Number:</strong> {{ student.parent_phone }}</p>
   <p class="mb-0"><strong>Predicted Grade:</strong> {{ student.predicted_grade }}</p>
   <hr>

   <h5>Student's Classes:</h5>
   {% for class in student.classes %}
    <div>
     <small>{{ class.class_name }}</small>
     <small class="text-muted">(Beginning: {{ class.class_starting_date.strftime('%d %b %Y') }})</small>
    </div>
   {% endfor %}
   <div class="mt-3 mb-0">
    <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('student_report', student_id=student.id) }}">Get Report</a>
    <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="{{ url_for('update_student', student_id=student.id) }}">Update
Details</a>
    <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="{{ url_for('add_to_class', student_id=student.id) }}">Add To
Class</a>
    <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="{{ url_for('remove_from_class', student_id=student.id)
}}">Remove From Class</a>
    <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-
target="#deleteModal">Delete</button>
   </div>
  </div>
 </article>
 <!-- Modal -->
 <div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
   <div class="modal-content">
    <div class="modal-header">
     <h5 class="modal-title" id="deleteModalLabel">Delete Student?</h5>
     <button type="button" class="close" data-dismiss="modal" aria-label="Close">
      <span aria-hidden="true">&times;</span>
     </button>
    </div>
    <div class="modal-footer">
     <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
     <form action="{{ url_for('delete_student', student_id=student.id) }}" method="POST">
      <input class="btn btn-danger" type="submit" value="Delete">
     </form>
    </div>
   </div>
  </div>
 </div>

 <!-- Linking all the scripts! -->
```

```
  <script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
  <script type="text/javascript" src="{{ url_for('static', filename='js/dashboard_charts.js') }}"></script>
  <script type="text/javascript">

    // Setting graph to load onload.
    $(document).ready(function() {
      changeChart("performanceGraph", "line",
                {{ class_perf.labels | safe }},
                {{ class_perf.datasets | safe }});
    });
  </script>
{% endblock content %}
```

# single_test.html



```
{% extends "layout_content.html" %}
{% block content %}
  <article class="parent media content-section">

    <div class="ribbon"><a style="color:white;" href="{{ url_for('courses', course_id=test.topic.course.id) }}">{{
test.topic.course.name }}</a></div>

    <img class="rounded-circle article-img" src="{{ url_for('static', filename='profile_pics/def.png') }}">
    <div class="media-body">
      <h2><a class="article-title" href="{{ url_for('test', test_id=test.id) }}">{{ test.name }}</a></h2>


      <h4 class="m-0 p-0 mb-2">Classes Assigned To</h4>
  <p class="m-0 p-0 mb-3">
      {% for class in test.topic.course.classes %}
      - {{ class.class_name }}<br>
      {% endfor %}

      </p>

      <h4 class="mt-2">Recorded Performance</h4>

      {% if class_perf.datasets %}

      <div id="graph-container">
        <canvas id="performanceGraph"></canvas>
      </div>


      {% else %}
      <p>No data to graph.</p>
      {% endif %}

      <div class="mt-3 mb-0">
        <a class="btn btn-secondary btn-sm mt-1 mb-1" href="{{ url_for('update_test', test_id=test.id) }}">Update Details</a>
        <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="">Add To Class</a> -->
        <!-- <a class="btn btn-secondary btn-sm mt-1 mb-1 ml-1" href="">Remove From Class</a> -->
        <button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-
target="#deleteModal">Delete</button>
      </div>
    </div>
  </article>
  <!-- Modal -->
  <div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="deleteModalLabel">Delete Class?</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
          <form action="{{ url_for('delete_test', test_id=test.id) }}" method="POST">
            <input class="btn btn-danger" type="submit" value="Delete">
          </form>
        </div>
      </div>
    </div>
  </div>



  <script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
```

```
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>
  <script type="text/javascript" src="{{ url_for('static', filename='js/dashboard_charts.js') }}"></script>
  <script type="text/javascript">

    // Setting graph to load onload.
    $(document).ready(function() {
      changeChart("performanceGraph", "bar",
                {{ class_perf.labels | safe }},
                {{ class_perf.datasets | safe }});
    });
    </script>
{% endblock content %}
```