From app.py (Snippets only for User Authentication)

from flask import Flask, render\_template, redirect, url\_for, flash, request, session

from flask\_login import LoginManager, login\_user, logout\_user, login\_required, current\_user

from werkzeug.security import generate\_password\_hash

from datetime import datetime

from models import db, User, Question, StudentSubmission, Quiz, Subject, QuizSubmission

from forms import LoginForm, RegistrationForm, GradeSubmissionForm

from question\_forms import get\_question\_form, MultipleChoiceQuestionForm, IdentificationQuestionForm, TrueFalseQuestionForm

from flask\_wtf.csrf import CSRFProtect

from flask\_wtf import FlaskForm

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'your-secret-key-here'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///users.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

# Initialize extensions

db.init\_app(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

# Initialize CSRF protection

csrf = CSRFProtect()

csrf.init\_app(app)

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('dashboard'))

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(username=form.username.data).first()

if user and user.check\_password(form.password.data):

login\_user(user)

next\_page = request.args.get('next')

return redirect(next\_page) if next\_page else redirect(url\_for('dashboard'))

flash('Invalid username or password')

return render\_template('auth/login.html', form=form)

@app.route('/register', methods=['GET', 'POST'])

def register():

if current\_user.is\_authenticated:

return redirect(url\_for('dashboard'))

form = RegistrationForm()

if form.validate\_on\_submit():

if User.query.filter\_by(email=form.email.data).first():

flash('Email address already registered. Please use a different email.')

return render\_template('auth/register.html', form=form)

user = User(username=form.username.data, email=form.email.data, role=form.role.data)

user.set\_password(form.password.data)

db.session.add(user)

if form.role.data == 'teacher' and form.subject\_code.data:

subject = Subject(name=form.username.data + "'s Class",

subject\_code=form.subject\_code.data,

teacher\_id=user.id)

db.session.add(subject)

try:

db.session.commit()

flash('Registration successful! Please login.')

return redirect(url\_for('login'))

except Exception as e:

db.session.rollback()

if 'UNIQUE constraint' in str(e):

if 'username' in str(e):

flash('Username already taken. Please choose a different username.')

else:

flash('Email address already registered. Please use a different email.')

else:

flash('An error occurred during registration. Please try again.')

return render\_template('auth/register.html', form=form)

return render\_template('auth/register.html', form=form)

From forms.py (Snippets for User Authentication ONLY)

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, SubmitField, EmailField, TextAreaField, SelectField, FloatField, IntegerField, BooleanField, FieldList, FormField

from wtforms.validators import DataRequired, Email, Length, EqualTo, Optional, NumberRange

class LoginForm(FlaskForm):

username = StringField('Username', validators=[DataRequired()])

password = PasswordField('Password', validators=[DataRequired()])

submit = SubmitField('Login')

class RegistrationForm(FlaskForm):

username = StringField('Username', validators=[DataRequired(), Length(min=4, max=20)])

email = EmailField('Email', validators=[DataRequired(), Email()])

role = SelectField('Role', choices=[('student', 'Student'), ('teacher', 'Teacher')], validators=[DataRequired()])

subject\_code = StringField('Subject Code', validators=[Optional(), Length(min=3, max=20)])

password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])

confirm\_password = PasswordField('Confirm Password', validators=[DataRequired(), EqualTo('password')])

submit = SubmitField('Register')

models.py (Snippets only)

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import UserMixin

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from datetime import datetime

import json

db = SQLAlchemy()

class Subject(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

subject\_code = db.Column(db.String(20), unique=True, nullable=False)

teacher\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

created\_at = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

teacher = db.relationship('User', backref='subjects\_taught', foreign\_keys=[teacher\_id])

enrolled\_students = db.relationship('User', secondary='student\_subjects', backref='enrolled\_subjects')

quizzes = db.relationship('Quiz', backref='subject', lazy=True)

student\_subjects = db.Table('student\_subjects',

db.Column('student\_id', db.Integer, db.ForeignKey('user.id'), primary\_key=True),

db.Column('subject\_id', db.Integer, db.ForeignKey('subject.id'), primary\_key=True)

)

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

password\_hash = db.Column(db.String(128))

role = db.Column(db.String(20), nullable=False, default='student')

questions = db.relationship('Question', backref='author', lazy=True)

submissions = db.relationship('StudentSubmission', backref='student', lazy=True)

def set\_password(self, password):

self.password\_hash = generate\_password\_hash(password)

def check\_password(self, password):

return check\_password\_hash(self.password\_hash, password)

def \_\_repr\_\_(self):

return f'<User {self.username}>'