```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}')"
```