```python
1    """
2    module containing all models for multiple choice questions
3    """
4
5    from django.db import models
6    from learning_base.models import Question
7
8
9    class MultipleChoiceQuestion(Question):
10       """
11       A simple multiple choice question
12       :Leonhard Wiedmann:
13       """
14       __name__ = "multiple_choice"
15
16       question_image = models.TextField(
17           verbose_name="The Image for the question",
18           blank=True,
19       )
20
21       feedback_image = models.TextField(
22           verbose_name="The Image for the question",
23           blank=True,
24       )
25
26       def num_correct_answers(self):
27           """
28           :author: Claas Voelcker
29           :return: the number of correct answers
30           """
31           return len(MultipleChoiceAnswer.objects.filter(
32               question=self, is_correct=True))
33
34       def not_solvable(self):
35           """
36           Checks whether the question is actually solvable
37           :author: Claas Voelcker
38           :return: True if the number of answers is 0
39           """
40           return self.num_correct_answers() == 0
41
42       def evaluate(self, data):
43           """
44           data: Map of answers
45           :return: True if and only if the array of provided answers are exactly
46           the correct answers
47           :author: Tobias Huber
48           """
49
50           # get all correct answers, map them to their id and make a
51           # (mathematical) set out of it"
52           answers = set([x.id for x in
53                       self.multiplechoiceanswer_set.filter(is_correct=True)])
54           return answers == set(data)
55
56       def __str__(self):
57           return self.title
```

```python
    def answer_set(self):
        """
        shorthand for the answer set
        :author: Leonhard Wiedmann
        :return: the QuerySet of all answers
        """
        return self.multiplechoiceanswer_set.all()

    @staticmethod
    def get_serializer():
        """
        reverse for the serializer
        :author: Claas Voelcker
        :return: the fitting serializer
        """
        from . import serializer
        return serializer.MultipleChoiceQuestionSerializer

    def get_points(self):
        """
        returns the points value of the question
        :return: 2 if the course is hard, else 1
        """
        return 2 if self.module.course.category == 2 else 1

    @staticmethod
    def get_edit_serializer():
        """
        reverse for the serializer
        :author: Claas Voelcker
        :return: the fitting serializer
        """
        from . import serializer
        return serializer.MultipleChoiceQuestionEditSerializer


class MultipleChoiceAnswer(models.Model):
    """
    A possible answer to a multiple choice question
    :author: Claas Voelcker
    """
    question = models.ForeignKey(
        MultipleChoiceQuestion,
        on_delete=models.CASCADE
    )

    text = models.TextField(
        verbose_name="Answer text",
        help_text="The answers text"
    )

    is_correct = models.BooleanField(
        verbose_name='is the answer correct?',
        default=False
    )

    img = models.TextField(
        verbose_name="The Image for the answer",
        blank=True
    )

    def __str__(self):
        return self.text
```

```python
    @staticmethod
    def get_serializer():
        """
        reverse for the serializer
        :author: Claas Voelcker
        :return: the fitting serializer
        """
        from learning_base.multiple_choice import serializer
        return serializer.MultipleChoiceAnswerSerializer

    @staticmethod
    def get_edit_serializer():
        """
        reverse for the serializer
        :author: Claas Voelcker
        :return: the fitting serializer
        """
        from learning_base.multiple_choice import serializer
        return serializer.MultipleChoiceAnswerEditSerializer
```