



# Assessments



Also check out [rXX:8000/admin/doc/models](http://rXX:8000/admin/doc/models)

Assessments

assessments_assessmentsession	
123 id	NOT NULL
🕒 created	NOT NULL
🕒 modified	NOT NULL
ABC notes	NOT NULL
🕒 start_at	
🕒 end_at	
ABC status	NOT NULL
🕒 status_changed	NOT NULL
ABC week_group	NOT NULL
ABC assessment_id	NOT NULL
ABC cohort_id	NOT NULL
ABC dri_id	

assessments_submission	
123 id	NOT NULL
🕒 created	NOT NULL
🕒 modified	NOT NULL
ABC status	NOT NULL
🕒 status_changed	NOT NULL
ABC notes	NOT NULL
🕒 start_at	
🕒 end_at	
123 grade	
ABC feedback	NOT NULL
123 time_hrs	
ABC comments	NOT NULL
123 assessmentsession_id	NOT NULL
123 enrollment_id	NOT NULL
ABC grader_id	

Submission File Path:  
f'submissions/{cohort\_id}/{instance.  
id}-{filename}'

assessments_assessment	
ABC id	NOT NULL
🕒 created	NOT NULL
🕒 modified	NOT NULL
ABC notes	NOT NULL
ABC status	NOT NULL
🕒 status_changed	NOT NULL
ABC title	NOT NULL
ABC description	NOT NULL
ABC dri_id	

assessments_submissionfile	
ABC id	NOT NULL
🕒 created	NOT NULL
🕒 modified	NOT NULL
ABC zipfile	NOT NULL
123 submission_id	NOT NULL

# Mixins

```
7 class StaffOnlyNotesModel(models.Model):
8     """Mixin for models with a staff-only note field."""
9
10    notes = models.TextField(
11        blank=True,
12        verbose_name="staff notes",
13        help_text="These are never shown to students.",
14    )
15
16    class Meta:
17        abstract = True
```

```
19  class TimeStampedModel(models.Model):
20      """
21      An abstract base class model that provides self-updating
22      ``created`` and ``modified`` fields.
23
24      """
25      created = AutoCreatedField(_('created'))
26      modified = AutoLastModifiedField(_('modified'))
27
28      def save(self, *args, **kwargs):
29          """
30          Overriding the save method in order to make sure that
31          modified field is updated even if it is not given as
32          a parameter to the update field argument.
33          """
34          update_fields = kwargs.get('update_fields', None)
35          if update_fields:
36              kwargs['update_fields'] = set(update_fields).union({'modified'})
37
38          super().save(*args, **kwargs)
39
40      class Meta:
41          abstract = True
42
```

```
20 class TimeBoundModel(models.Model):
21     """Mixin for models with a start and end date."""
22
23     start_at = models.DateTimeField(
24         blank=True,
25         null=True,
26         db_index=True,
27     )
28
29     end_at = models.DateTimeField(
30         blank=True,
31         null=True,
32         db_index=True,
33     )
34
35     class Meta:
36         abstract = True
37
38     def clean(self):
39         """If start_at & end_at given, start_at cannot be after end_at."""
40
41         super().clean()
42         if self.end_at and self.start_at:
43             if self.end_at <= self.start_at:
44                 raise ValidationError({
45                     "start_at": "End date must be after start date.",
46                     "end_at": "End date must be after start date.",
47                 })
```

```
189 SUB_STATUS = [  
190     # they start as "open", but students can only download assets/submit  
191     # during after the open_at time window for them. "open" here just means  
192     # "we've agreed that this student will be given this assessment"  
193     ("open", "Open"),  
194     ("submitted", "Submitted"),  
195     ("claimed", "Claimed"),  
196     ("tentative", "Tentative"),  
197     ("graded", "Graded"),  
198     ("sent", "Feedback Sent"),  
199     ("ack", "Feedback Read"),  
200     # only choose private if we, for some reason, want the student not to be  
201     # able to ever see their assessment (this entirely hides it from them)  
202     ("private", "Private"),  
203 ]
```

```

class WorkflowModel(models.Model):
    """Mixin for models that have a workflow state.

    This is used for everything that goes through our normal workflow:
    private, retired, published.

    This adds `status` and `status_changed` fields, as well as query managers
    for private, published, retired, and "public" (not private).

    If your model class overrides `_workflow_required` like this::

        class YourModel(WorkflowModel, ...):
            _workflow_required = {"published": ["dri", "description"]}

    Your instance cannot be published if the dri or description fields are
    not set.
    """

    _workflow_required = {}

    status = StatusField(db_index=True)

    status_changed = MonitorField(
        monitor='status',
        editable=False,
    )

```



```
def clean(self):
    """Require that all state requirements are met."""

    super().clean()

    # noinspection PyUnresolvedReferences
    required = self._workflow_required.get(self.status)
    if not required:
        return True

    missing = {}
    for field_name in required:
        val = getattr(self, field_name)
        if val is None or val == "":
            missing[field_name] = f"Required for state '{self.status}'."

    if missing:
        raise ValidationError(missing)
```

```
100 class PubWorkflowModel(WorkflowModel):
101     """Our standard workflow model for private/retired/published."""
102
103     class WorkflowStates(models.TextChoices):
104         PRIVATE = 'private', 'Private'
105         PUBLISHED = 'published', 'Published'
106         RETIRED = 'retired', 'Retired'
107
108     STATUS = WorkflowStates.choices
109
110     class Meta:
111         abstract = True
112
113     status = StatusField(
114         db_index=True,
115         help_text=(
116             'Hidden when private (accessible, but unlisted, when retired).'),
117     )
118
119     objects = QueryManager()
120     private = QueryManager(status='private')
121     published = QueryManager(status='published')
122     retired = QueryManager(status='retired')
123     public = QueryManager(status__in=['published', 'retired'])
124
```



# Overview of Models



# Assessment Class

Column Name	Type	Relationship
Id	SlugField	Primary Key
Title	CharField	N/A
description	TextField	N/A
Dri_id (Directly Responsible Individual)	TextField	Foreign Key( <b>staff.Staffmember</b> )
notes	TextField	N/A
created	Timestamp from django <i>TimeStampedModel</i>	N/A
modified	Timestamp from django <i>TimeStampedModel</i>	N/A
status	StatusField	WorkFlowModel Mixin
status_changed	DateTimeField	WorkFlowModel Mixin

# Assessment Class

“Assessment across all of Rithm (ie. Parent Assessment)”

- Workflow\_required = {"published": ["dri", "description"]}
  - Instance cannot be published if dri or description fields are not set.

3 mixins: TimeStampedModel, PubWorkflowModel, StaffOnlyNotesModel

```
[sis=# SELECT * from assessments_assessment;
```

created	modified	notes	status	status_changed	id	title	description	dri_id
2023-05-09 19:56:05.583354-04	2023-05-09 19:56:05.600502-04	Testing	private	2023-05-09 19:56:05.583375-04	mgregerson-test	Test Assessment		mgregerson4321

(1 row)



# Let's Walk Through Creating a New Assessment





# Assessment Session Class





Column Name	Type	Relationship
Id	SlugField	Primary Key
assessment_id	CharField	Foreign Key ( <b>Assessment</b> )
cohort_id	TextField	Foreign Key( <b>courses.Cohort</b> )
dri_id	TextField	Foreign Key( <b>staff.Staffmember</b> )
week_group	CharField	N/A
require_github_url	BooleanField	N/A
require_deployment_url	BooleanField	N/A
require_zipfile	BooleanField	N/A
is_pass_fail	BooleanField	N/A
is_prework	BooleanField	N/A
created	TimeStamp	Mixin: TimestampedModel
modified	Timestamp	Mixin: TimestampedModel
status	StatusField	Mixin: PubWorkflowModel
status_changed	StatusField	Mixin: PubWorkflowModel

# Assessment Session Class

## “Assessment instance for a single cohort”

Workflow\_required = “Dri, start\_at, end\_at”

- Instance cannot be published if dri\_id, start\_at, and end\_at are not defined.

Meta:

- ‘Assessment’ and ‘cohort’ must be **unique together** (Same assessment can not be assigned to a cohort twice).



# Submission Class



Column Name	Type	Relationship
Id	Num	Primary Key
enrollment_id	Num	Foreign Key (Students.Enrollment)
assessmentsession_id	Num	Foreign Key (AssessmentSession)
feedback	TextField	N/A
grader	(Uses __str__ method in Staff Member)	Foreign Key (StaffMember)
feedback	(Uses __str__ method in Staff Member)	Foreign Key (StaffMember)
time_hours	FloatField	N/A
comments	TextField	N/A
last_submit_at	DateTimeField	N/A
Created	TimeStamp	TimeStampedModel
Modified	Timestamp	TimeStampedModel
Notes	TextField	StaffOnlyNotesModel
Grade	0-100 (PositiveSmallIntegerField)	N/A
github_url	URLField	N/A
deployment_url	URLField	N/A

# Submission Class

- No workflow required.
- 4 mixins:
  - TimestampedModel, TimeBoundModel, WorkFlowModel, StaffOnlyNotesModel
- Meta:
  - 'Enrollment\_id' and 'assessmentsession\_id' must be **unique together** (Same student can not have multiple submissions for the same assessment UNLESS they are uploading to same assessment)
- Many, many methods in this class. Some to note...
  - Submit(): Submits an assessment. Returns true/false.
  - submissionfile\_path():

```
def submissionfile_path(instance, filename):  
    cohort_id = instance.submission.assessmentsession.cohort_id  
    return f'submissions/{cohort_id}/{instance.id}-{filename}'
```

```
SUB_STATUS = [  
    # they start as "open", but students can only download assets/submit  
    # during after the open_at time window for them. "open" here just means  
    # "we've agreed that this student will be given this assessment"  
    ("open", "Open"),  
    ("submitted", "Submitted"),  
    ("claimed", "Claimed"),  
    ("tentative", "Tentative"),  
    ("graded", "Graded"),  
    ("sent", "Feedback Sent"),  
    ("ack", "Feedback Read"),  
    # only choose private if we, for some reason, want the student not to be  
    # able to ever see their assessment (this entirely hides it from them)  
    ("private", "Private"),  
]
```



Let's Take a Look at a Submission on the  
Admin Site





# Submission File Class





Column Name	Type	Relationship
Id	UUID Field	Primary Key
submission_id	Num (ID)	Foreign Key (Submission)
zipfile	FileField	Foreign Key (AssessmentSession)
created	Timestamp	N/A
modified	Timestamp	N/A

created	modified	id	zipfile	submission_id
2023-05-10 13:30:49.710784-04	2023-05-10 13:30:49.710784-04	ac455c98-e84a-48cc-8f35-72092fd63ed1	submissions/mgregerson/ac455c98-e84a-48cc-8f35-72092fd63ed1-flask-2.zip	1
(1 row)				

# Submission File Class

- No workflow required.
- 1 mixin:
  - TimestampedModel
- Zipfile field is important to note here:

```
zipfile = models.FileField(  
    upload_to=submissionfile_path,  
    validators=[  
        validators.FileExtensionValidator(allowed_extensions=['zip'])  
    ],  
)
```

# BONUS:

- Downloading submission files
- Sending emails (aka meet the “Humble Staff Robot”)