Knowledge Learning Project

Release 1.0

Nathi33

May 24, 2025

CONTENTS:

CHAPTER

ONE

APPLICATION CART

1.1 Tests

1.1.1 Test : views

class cart.tests.test_views.**CartViewsTests**(*methodName='runTest'*)

Bases: TestCase

Set up a user, theme, curriculum, and lesson for testing cart functionalities.

setUp()

Hook method for setting up the test fixture before exercising it.

test_add_to_cart_curriculum_success()

Test adding a curriculum to the cart successfully.

test_add_to_cart_lesson_success()

Test adding a lesson to the cart successfully.

test_cannot_add_curriculum_if_already_purchased()

Test curriculum cannot be added if already purchased.

test_cannot_add_curriculum_in_cart()

Test curriculum cannot be added again if already in the cart.

test_cannot_add_lesson_if_already_purchased()

Test lesson cannot be added if already purchased.

test_cannot_add_lesson_if_curriculum_already_purchased()

Test lesson cannot be added if the whole curriculum has already been purchased.

test_cannot_add_lesson_if_curriculum_in_cart()

Test lesson cannot be added if its curriculum is already in the cart.

test_invalid_item_type()

Test handling of invalid item type.

test_remove_from_cart()

Test removing an item from the cart.

1.2 Views

Cart management features: adding, deleting and viewing.

cart.views.**add_to_cart**(*request*, *item_id*, *item_type*)

Add a curriculum or lesson item to the user's shopping cart stored in session.

Validations: - Prevent adding items already purchased. - Prevent adding duplicate items. - Prevent mixing curriculum with its individual lessons in cart or purchases.

Parameters

- **request** (*HttpRequest*) – The POST request containing user session and data.
- **item_id** (*int or str*) – The ID of the curriculum or lesson to add.
- **item_type** (*str*) – 'curriculum' or 'lesson'.

Returns
Redirects to 'next' URL if valid, else to cart page with appropriate messages.

Return type
HttpResponseRedirect

cart.views.**get_purchased_items**(*user*)

Retrieve IDs of curriculums and lessons already purchased by the user.

Parameters
user (*User*) – The authenticated user instance.

Returns
(set of purchased curriculum IDs, set of purchased lesson IDs)

Return type
tuple

cart.views.**has_purchased_lessons_of_curriculum**(*payments_qs*, *curriculum*)

Check if the user has purchased at least one lesson from the given curriculum.

Parameters

- **payments_qs** (*QuerySet*) – Payments queryset filtered for the user with status 'paid'.
- **curriculum** (Curriculum) – Curriculum instance to check.

Returns
True if at least one lesson from the curriculum is purchased, False otherwise.

Return type
bool

cart.views.**redirect_secure**(*request*, *next_url=None*)

Redirect securely to the provided next_url if valid, else to the cart page.

Parameters

- **request** (*HttpRequest*) – The current HTTP request object.
- **next_url** (*str, optional*) – URL to redirect to. Defaults to None.

Returns
Redirect response to next_url or 'cart' view.

Return type
HttpResponseRedirect

cart.views.**remove_from_cart**(*request*, *item_id*, *item_type*)

Remove an item (curriculum or lesson) from the user's shopping cart in session.

Parameters

- **request** (*HttpRequest*) – The current request object.
- **item_id** (*int or str*) – ID of the item to remove.
- **item_type** (*str*) – 'curriculum' or 'lesson'.

Returns
Redirects to the cart page with a success message.

Return type
HttpResponseRedirect

cart.views.**view_cart**(*request*)

Render the shopping cart page with current cart items and total price.

Parameters
request (*HttpRequest*) – The current request object.

Returns
Rendered cart page with context data.

Return type
HttpResponse

CHAPTER

TWO

APPLICATION CERTIFICATES

2.1 Tests

2.1.1 Test : models

class certificates.tests.test_models.**CertificateModelTests**(*methodName='runTest'*)

Bases: TestCase

setUp()

Set up a test user, theme, curriculum, and associated lessons.

test_certificate_clean_fails_if_lessons_not_completed()

Test that certificate validation fails if not all lessons are completed.

test_certificate_clean_passes_if_all_lessons_completed()

Test that certificate validation passes if all lessons are marked as completed.

test_certificate_str_method()

Test the string representation of the Certificate model.

2.1.2 Test : views

certificates.tests.test_views.**test_view_certificate_user_not_authenticated**()

Test that an unauthenticated user is redirected when trying to access the certificate view.

2.2 Models

class certificates.models.**Certificate**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Represents a certificate issued to a user for a specific theme.

This certificate is valid only if the user has completed all lessons from the curriculums associated with the theme.

user

The user to whom the certificate is issued.

Type
ForeignKey

theme

The theme corresponding to the certificate.

Type
ForeignKey

issued_at

Automatically set date and time when the certificate is issued.

Type
DateTimeField

is_valid

Indicates whether the certificate is valid (default: True).

Type
BooleanField

clean()

Validates that the user is authenticated and has completed all the required lessons to receive this certificate.

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

clean()

Validates the certificate's integrity before saving.

Checks that: - The user is authenticated. - The user has completed all lessons in the curriculums related to the theme.

Raises a ValidationError if these conditions are not met.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

created_by_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_issued_at(*, *field=<django.db.models.fields.DateTimeField: issued_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_issued_at(*, *field=<django.db.models.fields.DateTimeField: issued_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_valid

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

issued_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

theme

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

theme_id

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

2.3 Views

certificates.views.**view_certificate**(*request*, *theme_id*)

Displays a user's certificate for a given theme if eligible.

If the user has not completed all lessons in the theme's curriculums, a page indicating they are not eligible for the certificate is shown.

Parameters

- **request** (*HttpRequest*) – The incoming HTTP request.
- **theme_id** (*int*) – The ID of the theme for which to display the certificate.

Returns

The certificate page or the "not eligible" page.

Return type

HttpResponse

CHAPTER

THREE

APPLICATION CORE

3.1 Middleware

class core.middleware.**CurrentUserMiddleware**(*get_response*)

Bases: `object`

Middleware that stores the current user in a thread-local variable.

This allows access to the current user from anywhere in the code (e.g., models, signals, utils) without explicitly passing the request object.

get_response

The next middleware or view to call.

Type
callable

core.middleware.**get_current_user**()

Retrieves the current user from thread-local storage.

Returns
The currently authenticated user, or None if not set.

Return type
User or None

3.2 Models

class core.models.**AuditableMixin**(**args*, ***kwargs*)

Bases: *TimeStampeMixin*

Abstract base model that adds user tracking fields to log which user created or last modified the object.

Inherits from:
TimeStampeMixin: Provides created_at and updated_at fields.

Fields:
created_by (ForeignKey): The user who created the object. updated_by (ForeignKey): The user who last modified the object.

save()

Overrides the default save method to automatically populate created_by and updated_by based on the current user.

class Meta

Bases: object

abstract = False

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

created_by_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

save(**args*, ***kwargs*)

Overrides the default save method to automatically assign the current user to created_by and updated_by fields.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

updated_by_id

class core.models.**TimeStampeMixin**(**args*, ***kwargs*)

Bases: Model

Abstract base model that adds timestamp fields to track creation and update times.

Fields:

created_at (DateTimeField): Automatically set to the current datetime when the object is created. updated_at (DateTimeField): Automatically updated to the current datetime each time the object is saved.

`class Meta`

Bases: `object`

`abstract = False`

`created_at`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_created_at`(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

`get_next_by_updated_at`(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

`get_previous_by_created_at`(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

`get_previous_by_updated_at`(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

`updated_at`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

3.3 Views

`core.views.home`(*request*)

Render the homepage displaying all available themes.

Retrieves all Theme objects from the database and passes them to the 'core/home.html' template for rendering.

Parameters

request (*HttpRequest*) – The incoming HTTP request.

Returns

The rendered homepage with the list of themes.

Return type

HttpResponse

CHAPTER

FOUR

APPLICATION COURSES

4.1 Templatetags

4.1.1 Templatetags : custom

courses.templatetags.custom_tags.**get_item**(*dictionary*, *key*)

Retrieve a value from a dictionary using the given key.

This filter allows access to dictionary items in Django templates.

Parameters

- **dictionary** (*dict*) – The dictionary to retrieve the value from.
- **key** – The key to look up in the dictionary.

Returns

The value corresponding to the key if it exists, otherwise None.

4.2 Tests

4.2.1 Test : models

courses.tests.test_models.**test_create_theme_curriculum_lesson**()

Test the creation of a Theme, Curriculum, and Lesson, and verify their string representations.

courses.tests.test_models.**test_lesson_completion_and_is_completed_by_user**()

Test the behavior of is_completed_by_user method for a Lesson.

courses.tests.test_models.**test_theme_is_certified_by_user**(*monkeypatch*)

Test the is_certified_by_user method of Theme using monkeypatching to simulate lesson completion scenarios.

4.2.2 Test : views

class courses.tests.test_views.**CoursesViewsTests**(*methodName='runTest'*)

Bases: TestCase

setUp()

Create a test user, theme, curriculum and lesson before each test.

test_complete_lesson_post()

Test completing a lesson via POST request after simulating a paid lesson.

test_curriculum_detail_view()

Ensure the curriculum detail view displays the lesson title.

test_theme_detail_view()

Ensure the theme detail view displays the curriculum title.

4.3 Models

class courses.models.**Curriculum**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Represents a curriculum consisting of a collection of lessons within a theme.

theme

ForeignKey to the Theme it belongs to.

Type
Theme

title

Title of the curriculum.

Type
str

price

Price in euros.

Type
int

is_paid_by_user(*user*)

Checks if the user has paid for this curriculum.

exception DoesNotExist

Bases: ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

created_by_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_paid_by_user(*user*)

Check if the user has paid for this curriculum.

Parameters
user (*User*) – The user to check payment status for.

Returns
True if the user has a paid payment record for this curriculum, False otherwise.

Return type
bool

lessons

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <django.db.models.manager.Manager object>

payment_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

theme

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

theme_id

title

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

class courses.models.**Lesson**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Represents an individual lesson within a curriculum.

curriculum

ForeignKey to the Curriculum it belongs to.

Type
Curriculum

title

Title of the lesson.

Type
str

content

Text content of the lesson.

Type
str

order

Order of the lesson within the curriculum.

Type
int

price

Price in euros for this lesson.

Type
int

is_validated

Flag indicating if the lesson is validated.

Type
bool

video_url

Optional URL for an associated video.

Type
str or None

is_completed_by_user(*user*)

Checks if the user has completed this lesson.

is_paid_by_user(*user*)

Checks if the user has paid for this lesson.

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

content

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

created_by_id

curriculum

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

curriculum_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_completed_by_user(*user*)

Check if the user has paid for this lesson.

Parameters
user (*User*) – The user to check payment status for.

Returns
True if the user has a paid payment record for this lesson, False otherwise.

Return type
bool

is_paid_by_user(*user*)

is_validated

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lessoncompletion_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <django.db.models.manager.Manager object>

order

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

payment_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

price

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

title

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

video_url

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class courses.models.**LessonCompletion**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Tracks the completion status of a lesson for a user.

user

The user who completed the lesson.

Type
User

lesson

The lesson being completed.

Type
Lesson

is_completed

Whether the lesson is completed.

Type
bool

completed_at

Date and time when the lesson was completed.

Type
datetime or None

exception DoesNotExist

Bases: `ObjectDoesNotExist`

exception MultipleObjectsReturned

Bases: `MultipleObjectsReturned`

completed_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

created_by_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_completed

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lesson

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

lesson_id

objects = <django.db.models.manager.Manager object>

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class courses.models.**Theme**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Represents a course theme grouping multiple curriculums.

name

Unique name of the theme.

Type
str

is_certified_by_user(*user*)

Checks if the user has completed all lessons in the theme.

exception DoesNotExist

Bases: ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

certificate_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

created_by_id

curriculums

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_certified_by_user(*user*)

Check if the given user has completed all lessons in all curriculums of this theme.

Parameters
user (*User*) – The user to check completion status for.

Returns
True if all lessons are completed by the user, False otherwise.

Return type
bool

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

4.4 Views

courses.views.**complete_lesson**(*request*, *lesson_id*)

Mark a lesson as completed for the logged-in user.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **lesson_id** (*int*) – Primary key of the lesson to mark complete.

Returns
Redirect to dashboard after completion.

Return type
HttpResponseRedirect

courses.views.**curriculum_detail**(*request*, *curriculum_id*)

Display detailed information about a curriculum and its lessons.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **curriculum_id** (*int*) – Primary key of the curriculum to display.

Returns
Rendered template with curriculum details and ordered lessons.

Return type
HttpResponse

courses.views.**lesson_detail**(*request*, *lesson_id*)

Display the lesson content if the user has purchased access.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **lesson_id** (*int*) – Primary key of the lesson to display.

Returns
Rendered lesson detail page or access denied page.

Return type
HttpResponse

courses.views.**purchase_curriculum**(*request*, *curriculum_id*)

View to initiate purchase of a curriculum. Requires user to be logged in.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **curriculum_id** (*int*) – Primary key of the curriculum to purchase.

Returns
Rendered purchase page for the curriculum.

Return type
HttpResponse

courses.views.**theme_detail**(*request*, *theme_id*)

Display detailed information about a specific theme including its curriculums.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **theme_id** (*int*) – Primary key of the theme to display.

Returns
Rendered template with the theme and its curriculums.

Return type
HttpResponse

courses.views.**themes_list**(*request*)

Render a page listing all available course themes.

Parameters
request (*HttpRequest*) – The HTTP request object.

Returns
Rendered template with the list of themes.

Return type
HttpResponse

CHAPTER

FIVE

APPLICATION DASHBOARD

5.1 Tests

5.1.1 Test : views

dashboard.tests.test_views.**test_dashboard_empty_data**(*client*)

Test dashboard with no purchased curriculum or standalone lessons.

dashboard.tests.test_views.**test_dashboard_requires_login**(*client*)

Test that the dashboard view requires authentication.

dashboard.tests.test_views.**test_dashboard_with_completed_lessons**(*client*)

Test dashboard progress calculation based on completed lessons.

dashboard.tests.test_views.**test_dashboard_with_purchased_curriculum**(*client*)

Test dashboard when a curriculum has been purchased by the user.

dashboard.tests.test_views.**test_dashboard_with_standalone_lessons**(*client*)

Test dashboard view when standalone lessons (not entire curriculum) are purchased.

5.2 Context_processors

dashboard.context_processors.**has_paid_purchases**(*request*)

Checks if the authenticated user has any completed (paid) payments.

Parameters

request (*HttpRequest*) – The incoming HTTP request object.

Returns

A context dictionary with the key 'has_paid_purchases' set to True
if the user has at least one payment with status 'paid', otherwise False.

Return type

dict

5.3 Views

dashboard.views.**dashboard**(*request*)

Render the user's dashboard displaying their purchased curriculums and lessons, along with their completion progress organized by themes and curriculums.

This view performs the following steps: - Retrieves all successful payments for the logged-in user. - Identifies purchased curriculums and individually purchased lessons. - Fetches lessons completed by the user. - Organizes individually purchased lessons by theme and curriculum,

including whether each lesson is purchased and/or completed.

- Calculates progress percentages for themes based on completed lessons.
- Lists curriculums purchased by the user within each theme, computing progress per curriculum and overall theme progress.
- Passes structured data to the 'dashboard/dashboard.html' template for rendering.

Context passed to the template: - 'theme_with_curriculums': List of themes with their curriculums,

each curriculum annotated with purchase status and progress percentage.

- 'standalone_lessons_by_theme': Dict mapping themes to their curriculums and lessons bought individually, with purchase and completion flags.
- 'completed_lessons_ids': IDs of lessons marked as completed by the user.
- 'theme_progress_by_theme': Progress percentage of completed lessons per theme for purchased curriculums.
- 'standalone_progress_by_theme': Progress percentage of completed individually purchased lessons per theme.

Requires user authentication.

Parameters
request (*HttpRequest*) – The HTTP request object.

Returns
Rendered dashboard page with user's learning progress and purchases.

Return type
HttpResponse

CHAPTER

SIX

APPLICATION PAYMENTS

6.1 Tests

6.1.1 Test : models

payments.tests.test_models.**test_create_payment_curriculum**()

Test creating a payment linked to a curriculum. Ensures the payment is validated, saved, and string representation is correct.

payments.tests.test_models.**test_create_payment_lesson**()

Test creating a payment linked to a lesson. Ensures the payment is validated, saved, and string representation is correct.

payments.tests.test_models.**test_payment_validation_error_both_curriculum_and_lesson**()

Test that a payment cannot be linked to both a curriculum and a lesson at the same time. Expects a ValidationError.

payments.tests.test_models.**test_payment_validation_error_neither_curriculum_nor_lesson**()

Test that a payment must be linked to either a curriculum or a lesson. Expects a ValidationError when neither is set.

payments.tests.test_models.**test_str_unknown_user_and_invalid_payment**()

Test __str__ method behavior when the user is None and the payment is invalid. Ensures fallback string is used.

6.1.2 Test : views

class payments.tests.test_views.**PaymentViewTests**(*methodName='runTest'*)

Bases: TestCase

setUp()

Set up test data: user, theme, curriculum, lesson, and authenticated client session.

test_create_checkout_session_empty_cart()

Test that posting with an empty cart returns a 400 response with an error message.

test_create_checkout_session_with_lesson()

Test that a checkout session is successfully created with a lesson in the cart. Expects a 200 response and a session ID in the JSON response.

6.2 Models

class payments.models.**Payment**(**args*, ***kwargs*)

Bases: *AuditableMixin*, Model

Model representing a payment made by a user for either a curriculum or a lesson.

user

Reference to the user who made the payment. Can be null if the user is deleted.

Type
ForeignKey

curriculum

The purchased curriculum. Null if the payment is for a lesson.

Type
ForeignKey

lesson

The purchased lesson. Null if the payment is for a curriculum.

Type
ForeignKey

amount

Amount paid in cents (e.g., 1999 for €19.99).

Type
PositiveIntegerField

timestamp

The date and time the payment was made.

Type
DateTimeField

status

The payment status. Possible values: 'paid' or 'failed'.

Type
CharField

stripe_checkout_id

Unique Stripe session ID associated with the payment.

Type
CharField

Validation:

- A payment must be linked to either a curriculum or a lesson, but not both.
- A payment must be linked to at least one (either curriculum or lesson).

exception DoesNotExist

Bases: ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

amount

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

clean()

Hook for doing any extra model-wide validation after clean() has been called on every field by self.clean_fields. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by NON_FIELD_ERRORS.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

created_by_id

curriculum

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

curriculum_id

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_timestamp(*, *field=<django.db.models.fields.DateTimeField: timestamp>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_timestamp(*, *field=<django.db.models.fields.DateTimeField: timestamp>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

get_status_display(*, *field=<django.db.models.fields.CharField: status>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lesson

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

lesson_id

objects = <django.db.models.manager.Manager object>

save(**args*, ***kwargs*)

Overrides the default save method to automatically assign the current user to created_by and updated_by fields.

status

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

stripe_checkout_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

timestamp

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

6.3 Views

payments.views.**cart_success**(*request*)

Clears the user's session cart and displays a success message after payment.

Renders the 'payments/success.html' success page.

payments.views.**create_checkout_session**(*request*)

Creates a Stripe Checkout session for the items in the cart stored in the session.

Accepted method: POST only. Retrieves the items from the user's session cart, creates a Stripe session with these items, and returns the Stripe session ID to the frontend.

Returns HTTP 400 if the method is not POST or if the cart is empty.

payments.views.**payment_success**(*request*)

Displays the payment success confirmation page.

payments.views.**stripe_webhook**(*request*)

Stripe webhook endpoint to handle Stripe events.

Verifies the webhook signature. If the event type is 'checkout.session.completed', it creates Payment entries linked to the user and the purchased items (lessons or curriculums).

CHAPTER

SEVEN

APPLICATION USERS

7.1 Tests

7.1.1 Test : models

users.tests.test_models.**test_create_superuser_success**()

Test successful creation of a superuser with required fields. Checks that is_staff and is_superuser are True.

users.tests.test_models.**test_create_superuser_without_is_staff_raises**()

Test that creating a superuser with is_staff=False raises a ValueError indicating is_staff must be True for superusers.

users.tests.test_models.**test_create_superuser_without_is_superuser_raises**()

Test that creating a superuser with is_superuser=False raises a ValueError indicating is_superuser must be True for superusers.

users.tests.test_models.**test_create_user_success**()

Test successful creation of a regular user with valid email, password, first name, and last name. Checks attributes and string representation.

users.tests.test_models.**test_create_user_without_email_raises**()

Test that creating a user without an email raises a ValueError with the message indicating email must be provided.

7.1.2 Test : views

users.tests.test_views.**test_account_activation**(*client*)

Test the account activation flow. Verifies that accessing the activation link activates the user, redirects to home, and logs the user in.

users.tests.test_views.**test_login_success**(*client*)

Test successful login with correct credentials. Verifies redirect to home and that the session contains the user's id.

users.tests.test_views.**test_registration_user**(*client*)

Test the user registration process. Checks that submitting the registration form creates an inactive user and redirects to the login page.

users.tests.test_views.**test_registration_with_existing_email**(*client*)

Test that registering with an already used email address returns the registration form with an email error.

7.2 Admin

class users.admin.**CustomUserAdmin**(*model*, *admin_site*)

Bases: UserAdmin

add_fieldsets = ((None, {'classes': ('wide',), 'fields': ('email', 'last_name', 'first_name', 'password1', 'password2', 'is_staff', 'is_active')}),)

fieldsets = ((None, {'fields': ('email', 'password')}), ('Informations personnelles', {'fields': ('first_name', 'last_name')}), ('Permissions', {'fields': ('is_active', 'is_staff', 'is_superuser')}), ('Groupes', {'fields': ('groups', 'user_permissions')}))

list_display = ('email', 'first_name', 'last_name', 'is_staff')

property media

model

alias of *CustomUser*

ordering = ('email',)

search_fields = ('email', 'first_name', 'last_name')

7.3 Backends

class users.backends.**EmailBackend**

Bases: ModelBackend

Custom authentication backend that authenticates users using their email address instead of username.

authenticate(*request*, *username=None*, *password=None*, ***kwargs*)

Authenticate a user based on email and password.

Parameters

- **request** (*HttpRequest*) – The current request instance.
- **username** (*str*) – The email address of the user (passed as 'username' by Django auth system).
- **password** (*str*) – The user's password.
- ****kwargs** – Additional keyword arguments.

Returns

User instance if authentication is successful, otherwise None.

7.4 Forms

class users.forms.**CustomAuthenticationForm**(*request=None*, **args*, ***kwargs*)

Bases: AuthenticationForm

Authentication form that uses email instead of username to login.

Fields:

username (EmailField): Email field replacing the default username field.

base_fields = {'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.EmailField object>}

confirm_login_allowed(*user*)

Check if the given user account is active before allowing login.

Raises
ValidationError – If the user account is inactive.

declared_fields = {'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.EmailField object>}

property media

Return all media required to render the widgets on this form.

class users.forms.**CustomUserCreationForm**(**args*, ***kwargs*)

Bases: UserCreationForm

Form for creating a new user with required fields: email, last name, and first name.

Fields:
email (EmailField): Required email address. last_name (CharField): Required last name. first_name (CharField): Required first name.

class Meta

Bases: object

fields = ['last_name', 'first_name', 'email', 'password1', 'password2']

model

alias of *CustomUser*

base_fields = {'email': <django.forms.fields.EmailField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>}

declared_fields = {'email': <django.forms.fields.EmailField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>}

property media

Return all media required to render the widgets on this form.

save(*commit=True*)

Save the new user instance with the provided email, last name, and first name.

Parameters
commit (*bool*) – Whether to save the user to the database immediately.

Returns
The saved user instance.

Return type
user (*CustomUser*)

7.5 Models

class users.models.**CustomUser**(**args*, ***kwargs*)

Bases: AbstractUser, *AuditableMixin*

Custom user model that uses email instead of username for authentication.

Includes additional fields such as *is_client*, *is_admin*, and *email_verified*. Inherits auditing features from AuditableMixin.

exception DoesNotExist

Bases: ObjectDoesNotExist

exception MultipleObjectsReturned

Bases: MultipleObjectsReturned

REQUIRED_FIELDS = ['first_name', 'last_name']

USERNAME_FIELD = 'email'

certificate_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

certificate_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

certificate_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

created_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

created_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

created_by_id

curriculum_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

curriculum_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

customuser_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

customuser_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

date_joined

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

email

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

email_verified

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

first_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=True*, ***kwargs*)

get_next_by_date_joined(*, *field=<django.db.models.fields.DateTimeField: date_joined>*, *is_next=True*, ***kwargs*)

get_next_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=True*, ***kwargs*)

get_previous_by_created_at(*, *field=<django.db.models.fields.DateTimeField: created_at>*, *is_next=False*, ***kwargs*)

get_previous_by_date_joined(*, *field=<django.db.models.fields.DateTimeField: date_joined>*, *is_next=False*, ***kwargs*)

get_previous_by_updated_at(*, *field=<django.db.models.fields.DateTimeField: updated_at>*, *is_next=False*, ***kwargs*)

groups

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_admin

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_client

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_staff

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_superuser

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_login

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lesson_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

lesson_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

lessoncompletion_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

lessoncompletion_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

lessoncompletion_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

logentry_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = <users.models.CustomUserManager object>

password

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

payment_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

payment_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

payment_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

theme_created_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

theme_updated_by

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

updated_at

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated_by

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

updated_by_id

user_permissions

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

username = None

class users.models.**CustomUserManager**(**args*, ***kwargs*)

Bases: `BaseUserManager`

Custom manager for the CustomUser model based on email authentication.

Provides methods to create regular users and superusers without a username field, using only the email address for identification.

create_superuser(*email*, *password=None*, ***extra_fields*)

Create and return a superuser with administrative privileges.

Parameters

- **email** (*str*) – The superuser's email address.
- **password** (*str, optional*) – The superuser's password.
- ****extra_fields** – Additional fields to be set on the superuser.

Returns

The created superuser instance.

Return type

CustomUser

Raises

ValueError – If *is_staff* or *is_superuser* are not set to True.

create_user(*email*, *password=None*, ***extra_fields*)

Create and return a user with the given email and password.

Parameters

- **email** (*str*) – The user's email address.

- **password** (*str, optional*) – The user's password.
- ****extra_fields** – Additional fields to be set on the user.

Returns
The created user instance.

Return type
CustomUser

Raises
ValueError – If no email is provided.

use_in_migrations = True

If set to True the manager will be serialized into migrations and will thus be available in e.g. RunPython operations.

7.6 Views

class users.views.**CustomLoginView**(***kwargs*)

Bases: LoginView

Custom login view.

Uses a custom authentication form and displays a welcome message upon successful login.

authentication_form

alias of *CustomAuthenticationForm*

form_valid(*form*)

Display a success message and proceed with login if the form is valid.

Parameters
form (*AuthenticationForm*) – The validated authentication form.

Returns
A redirect to the success URL.

Return type
HttpResponseRedirect

template_name = 'users/login.html'

class users.views.**CustomLogoutView**(***kwargs*)

Bases: LogoutView

Custom logout view.

Displays a confirmation message after user logout.

dispatch(*request*, **args*, ***kwargs*)

Handle the logout request.

Parameters
request (*HttpRequest*) – The HTTP request object.

Returns
The response after logout.

Return type
HttpResponse

users.views.**activate**(*request*, *uidb64*, *token*)

Activate a user account via an email verification link.

Decodes the UID, verifies the token, activates the user, logs them in, and redirects to the homepage or a next URL if provided.

Parameters

- **request** (*HttpRequest*) – The HTTP request object.
- **uidb64** (*str*) – Base64-encoded user ID.
- **token** (*str*) – Verification token.

Returns
A redirect or an invalid activation message page.

Return type
HttpResponse

users.views.**register_view**(*request*)

Handle user registration.

Displays a registration form, creates an inactive user upon successful submission, sends an activation email, and redirects to the login page.

Parameters
request (*HttpRequest*) – The HTTP request object.

Returns
A rendered registration form or a redirect to login.

Return type
HttpResponse

CHAPTER

EIGHT

APPLICATION KNOWLEDGE_LEARNING_PROJECT

8.1 Knowledge_learning_project.asgi

ASGI config for knowledge_learning_project project.

It exposes the ASGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/5.2/howto/deployment/asgi/

8.2 Knowledge_learning_project.settings

Django settings for knowledge_learning_project project.

Generated by 'django-admin startproject' using Django 5.2.

For more information on this file, see https://docs.djangoproject.com/en/5.2/topics/settings/

For the full list of settings and their values, see https://docs.djangoproject.com/en/5.2/ref/settings/

8.3 Knowledge_learning_project.urls

URL configuration for knowledge_learning_project project.

The *urlpatterns* list routes URLs to views. For more information please see:
https://docs.djangoproject.com/en/5.2/topics/http/urls/

Examples: Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

8.4 Knowledge_learning_project.wsgi

WSGI config for knowledge_learning_project project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/5.2/howto/deployment/wsgi/

PYTHON MODULE INDEX

c

d

k

p

u

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V