

---

# **digiMED Documentation**

*Release*

## **Mind Optimizers**

**Nov 26, 2020**



## CONTENTS:

<b>1</b>	<b>source</b>	<b>1</b>
<b>2</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</b>



## 1.1 Account package

### 1.1.1 Subpackages

### 1.1.2 Submodules

### 1.1.3 Account.admin module

### 1.1.4 Account.apps module

```
class Account.apps.AccountConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'Account'
```

### 1.1.5 Account.models module

```
class Account.models.UserProfile(id, user, profile_type, age)
    Bases: django.db.models.base.Model

exception DoesNotExist
    Bases: django.core.exceptions.ObjectDoesNotExist

exception MultipleObjectsReturned
    Bases: django.core.exceptions.MultipleObjectsReturned
```

#### **age**

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

#### **doctor\_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.

**id**

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>

**patient\_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.

**profile\_type**

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

**user**

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Restaurant.place is a ForwardOneToOneDescriptor instance.

**user\_id**

### 1.1.6 Account.tests module

### 1.1.7 Account.urls module

### 1.1.8 Account.views module

Account.views.login(request)

Logs in the users to their respective accounts

**Parameters** request – Default HttpRequest object

**Returns** Rendered HTML Webpage

Account.views.logout(request)

Redirects the user to the register page when they logout of the respective account

**Parameters** request – Default HttpRequest object

**Returns** Rendered HTML Webpage

Account.views.register(request)

Takes the information of the users (doctor/patients) and fills the same in the database

**Parameters** request – Default HttpRequest object

**Returns** Rendered HTML Webpage

### 1.1.9 Module contents

## 1.2 db\_models module

## 1.3 digiMED package

### 1.3.1 Submodules

### 1.3.2 digiMED.asgi module

ASGI config for digiMED 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/3.1/howto/deployment/asgi/>

### 1.3.3 digiMED.settings module

Django settings for digiMED project.

Generated by ‘`django-admin startproject`’ using Django 3.1.2.

For more information on this file, see <https://docs.djangoproject.com/en/3.1/topics/settings/>

For the full list of settings and their values, see <https://docs.djangoproject.com/en/3.1/ref/settings/>

### 1.3.4 digiMED.urls module

digiMED URL Configuration

**The `urlpatterns` list routes URLs to views. For more information please see:** <https://docs.djangoproject.com/en/3.1/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'))`

### 1.3.5 digiMED.wsgi module

WSGI config for digiMED 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/3.1/howto/deployment/wsgi/>

### 1.3.6 Module contents

## 1.4 doctor package

### 1.4.1 Subpackages

### 1.4.2 Submodules

### 1.4.3 doctor.admin module

### 1.4.4 doctor.apps module

```
class doctor.apps.DoctorConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig

    name = 'doctor'
```

### 1.4.5 doctor.models module

```
class doctor.models.doctorRequestsTable(id, pid, docid)
    Bases: django.db.models.base.Model
```

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**docid**

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.

**docid\_id**

**id**

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>**

**pid**

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.

`pid_id`

### 1.4.6 doctor.tests module

### 1.4.7 doctor.urls module

### 1.4.8 doctor.views module

`doctor.views.doctorRequest` (*request*)

It will store the patient id whose health records the current user/doctor wants to access

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.doctorSignup` (*request*)

It registers the details of the doctor and redirects the user to home page

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.doctorshowhistory` (*request, patientid*)

It will show the health record of the patient

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.index` (*request*)

It will fetch the patient id whose health records are to be shown

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.login` (*request*)

Backup login page for doctor

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.logout` (*request*)

Logs out the user from his/her account

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.saveprescription` (*request, patient*)

It will create a prescription record linking the doctor with his/her patient for whom the medicines are prescribed

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`doctor.views.saveprescriptionmedicine` (*request*)

It will save the medicines prescribed by the doctor to his/her patient in the database

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

## 1.4.9 Module contents

# 1.5 lab package

## 1.5.1 Subpackages

## 1.5.2 Submodules

## 1.5.3 lab.admin module

## 1.5.4 lab.apps module

```
class lab.apps.LabConfig(app_name, app_module)
    Bases: django.apps.config.AppConfig
    name = 'lab'
```

## 1.5.5 lab.models module

```
class lab.models.LabTest(id, pid, date, test)
    Bases: django.db.models.base.Model
```

### **exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

### **exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

### **date**

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

### **id**

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>

### **pid**

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.

### **pid\_id**

### **test**

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

## 1.5.6 lab.tests module

## 1.5.7 lab.urls module

## 1.5.8 lab.views module

`lab.views.index` (*request*)

It fetches the pending requests from the database and shows on the main page of lab technician and also inserts into the database any of the lab test details filled by the lab technician

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`lab.views.log` (*request*)

Backup login page for lab technician

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

`lab.views.test` (*request*)

It will show the webpage containing the form to be filled for the respective lab test

**Parameters** `request` – Default HttpRequest object

**Returns** Rendered HTML Webpage

## 1.5.9 Module contents

## 1.6 manage module

Django's command-line utility for administrative tasks.

`manage.main` ()

Run administrative tasks.

## 1.7 patient package

### 1.7.1 Subpackages

### 1.7.2 Submodules

### 1.7.3 patient.admin module

### 1.7.4 patient.apps module

**class** `patient.apps.PatientConfig` (*app\_name*, *app\_module*)

Bases: `django.apps.config.AppConfig`

`name` = 'patient'

## 1.7.5 patient.models module

**class** `patient.models.Basicmetabolismpanel` (*id, pid, date, glucose, creatinine, sodium, potassium, chloride, carbon\_dioxide*)

Bases: `django.db.models.base.Model`

**exception** `DoesNotExist`

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception** `MultipleObjectsReturned`

Bases: `django.core.exceptions.MultipleObjectsReturned`

**carbon\_dioxide**

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

**chloride**

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

**creatinine**

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

**date**

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\_date** (\*, *field=<django.db.models.fields.DateField: date>, is\_next=True, \*\*kwargs*)

**get\_previous\_by\_date** (\*, *field=<django.db.models.fields.DateField: date>, is\_next=False, \*\*kwargs*)

**glucose**

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

**id**

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>`

**pid**

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.

**pid\_id**

**potassium**

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

**sodium**

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

**class** `patient.models.Commonbloodcount` (*id, pid, date, wbc, rbc, hemoglobin, hematocrit, mcv, mch, mchc, rdw, platelets, neurophils, lymphs, monocytes*)

Bases: `django.db.models.base.Model`

**exception** `DoesNotExist`

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception** `MultipleObjectsReturned`

Bases: `django.core.exceptions.MultipleObjectsReturned`

**date**

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\_date** (\*, *field=<django.db.models.fields.DateField: date>, is\_next=True, \*\*kwargs*)

**get\_previous\_by\_date** (\*, *field=<django.db.models.fields.DateField: date>, is\_next=False, \*\*kwargs*)

**hematocrit**

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

**hemoglobin**

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

**id**

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

**lymphs**

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

**mch**

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

**mchc**

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

**mcv**

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

**monocytes**

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

**neurophils**

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>`

**pid**

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.

#### pid\_id

#### platelets

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

#### rbc

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

#### rdw

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

#### wbc

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

**class** patient.models.Doctor(*did, name, surname, gender, dob, country, profile\_id*)

Bases: django.db.models.base.Model

#### exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

#### exception MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

#### country

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

#### did

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

#### dob

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

#### doctorrequeststable\_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.

#### gender

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

**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>

**patientgranttable\_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.

**prescription\_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.

**profile\_id**

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.

**profile\_id\_id**

**surname**

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

**class** patient.models.**Medication** (*medid, name, generic\_name*)

Bases: django.db.models.base.Model

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**generic\_name**

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

**medid**

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

**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>

**prescriptmed\_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.

**class** patient.models.Mri(id, pid, date, mri)

Bases: django.db.models.base.Model

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**date**

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\_date** (\*, field=<django.db.models.fields.DateField: date>, is\_next=True, \*\*kwargs)

**get\_previous\_by\_date** (\*, field=<django.db.models.fields.DateField: date>, 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.

**mri**

Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width\_field/height\_field, if appropriate.

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

**pid**

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.

**pid\_id**

**class** `patient.models.Patient` (*pid, name, surname, gender, date, country, street\_address, city, state, postal, profile\_id*)

Bases: `django.db.models.base.Model`

**exception** `DoesNotExist`

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception** `MultipleObjectsReturned`

Bases: `django.core.exceptions.MultipleObjectsReturned`

**basicmetabolismpanel\_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.

**city**

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

**commonbloodcount\_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.

**country**

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

**date**

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

**doctorrequeststable\_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.

**gender**

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\_date** (\*, field=<django.db.models.fields.DateField: date>, is\_next=True, \*\*kwargs)

**get\_previous\_by\_date** (\*, field=<django.db.models.fields.DateField: date>, is\_next=False, \*\*kwargs)

**labtest\_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.

**mri\_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.

**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>**

**patientgranttable\_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.

**pid**

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

**postal**

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

**prescription\_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.

**profile\_id**

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.

**profile\_id\_id**

**state**

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

**street\_address**

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

**surname**

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

**xray\_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.

**class** patient.models.Prescription (prescript\_id, pid, did, date)

Bases: django.db.models.base.Model

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**date**

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

**did**

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.

**did\_id**

**get\_next\_by\_date** (\*, field=<django.db.models.fields.DateField: date>, is\_next=True, \*\*kwargs)

**get\_previous\_by\_date** (\*, field=<django.db.models.fields.DateField: date>, is\_next=False, \*\*kwargs)

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

**pid**

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.

**pid\_id****prescript\_id**

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

**prescriptmed\_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.

**class** patient.models.Prescriptmed(id, prescript\_id, medid, description, days)

Bases: django.db.models.base.Model

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**days**

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

**description**

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

**id**

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

**medid**

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.

**medid\_id**

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

**prescript\_id**

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.

**prescript\_id\_id**

**class** patient.models.Xray(*id, pid, date, xray*)

Bases: django.db.models.base.Model

**exception DoesNotExist**

Bases: django.core.exceptions.ObjectDoesNotExist

**exception MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

**date**

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\_date** (\*, *field=<django.db.models.fields.DateField: date>*, *is\_next=True*, *\*\*kwargs*)

**get\_previous\_by\_date** (\*, *field=<django.db.models.fields.DateField: date>*, *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.

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

**pid**

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.

**pid\_id**

**xray**

Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width\_field/height\_field, if appropriate.

**class** patient.models.patientGrantTable (*id, pid, docid*)

Bases: django.db.models.base.Model

**exception** DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

**exception** MultipleObjectsReturned

Bases: django.core.exceptions.MultipleObjectsReturned

**docid**

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.

**docid\_id**

**id**

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>

**pid**

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.

**pid\_id**

## 1.7.6 patient.tests module

## 1.7.7 patient.urls module

## 1.7.8 patient.views module

`patient.views.grantedRequests` (*request*)

Function call to render login page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.index` (*request*)

Function call to render home page of Patient

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.login` (*request*)

Function call to render login page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.logout` (*request*)

Function call to render logout/login page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.patientGrant` (*request*)

Function call to render login page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.patientSignup` (*request*)

Function call to render login page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.requestLab` (*request*)

Function call to render Lab Test Request page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

`patient.views.showHistory` (*request*)

Function call to render Patints History page

**Parameters** `request` – default HttpRequest object

**Returns** Rendered HTML webpage

## 1.7.9 Module contents



## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### a

Account, 3  
Account.admin, 1  
Account.apps, 1  
Account.models, 1  
Account.tests, 2  
Account.urls, 2  
Account.views, 2

### d

digimed, 4  
digimed.asgi, 3  
digimed.settings, 3  
digimed.urls, 3  
digimed.wsgi, 3  
doctor, 6  
doctor.admin, 4  
doctor.apps, 4  
doctor.models, 4  
doctor.tests, 5  
doctor.urls, 5  
doctor.views, 5

### l

lab, 7  
lab.admin, 6  
lab.apps, 6  
lab.models, 6  
lab.tests, 7  
lab.urls, 7  
lab.views, 7

### m

manage, 7

### p

patient, 19  
patient.admin, 7  
patient.apps, 7  
patient.models, 8  
patient.tests, 19  
patient.urls, 19  
patient.views, 19



## A

Account (module), 3  
 Account.admin (module), 1  
 Account.apps (module), 1  
 Account.models (module), 1  
 Account.tests (module), 2  
 Account.urls (module), 2  
 Account.views (module), 2  
 AccountConfig (class in Account.apps), 1  
 age (Account.models.UserProfile attribute), 1

## B

Basicmetabolismpanel (class in patient.models), 8  
 Basicmetabolismpanel.DoesNotExist, 8  
 Basicmetabolismpanel.MultipleObjectsReturned, 8  
 basicmetabolismpanel\_set (patient.models.Patient attribute), 13

## C

carbon\_dioxide (patient.models.Basicmetabolismpanel attribute), 8  
 chloride (patient.models.Basicmetabolismpanel attribute), 8  
 city (patient.models.Patient attribute), 13  
 Commonbloodcount (class in patient.models), 8  
 Commonbloodcount.DoesNotExist, 9  
 Commonbloodcount.MultipleObjectsReturned, 9  
 commonbloodcount\_set (patient.models.Patient attribute), 13  
 country (patient.models.Doctor attribute), 10  
 country (patient.models.Patient attribute), 13  
 creatinine (patient.models.Basicmetabolismpanel attribute), 8

## D

date (lab.models.LabTest attribute), 6  
 date (patient.models.Basicmetabolismpanel attribute), 8  
 date (patient.models.Commonbloodcount attribute), 9  
 date (patient.models.Mri attribute), 12  
 date (patient.models.Patient attribute), 13  
 date (patient.models.Prescription attribute), 15  
 date (patient.models.Xray attribute), 17

days (patient.models.Prescriptmed attribute), 16  
 description (patient.models.Prescriptmed attribute), 16  
 did (patient.models.Doctor attribute), 10  
 did (patient.models.Prescription attribute), 16  
 did\_id (patient.models.Prescription attribute), 16  
 digiMED (module), 4  
 digiMED.asgi (module), 3  
 digiMED.settings (module), 3  
 digiMED.urls (module), 3  
 digiMED.wsgi (module), 3  
 dob (patient.models.Doctor attribute), 10  
 docid (doctor.models.doctorRequestsTable attribute), 4  
 docid (patient.models.patientGrantTable attribute), 18  
 docid\_id (doctor.models.doctorRequestsTable attribute), 4  
 docid\_id (patient.models.patientGrantTable attribute), 18  
 Doctor (class in patient.models), 10  
 doctor (module), 6  
 doctor.admin (module), 4  
 doctor.apps (module), 4  
 Doctor.DoesNotExist, 10  
 doctor.models (module), 4  
 Doctor.MultipleObjectsReturned, 10  
 doctor.tests (module), 5  
 doctor.urls (module), 5  
 doctor.views (module), 5  
 doctor\_set (Account.models.UserProfile attribute), 1  
 DoctorConfig (class in doctor.apps), 4  
 doctorRequest() (in module doctor.views), 5  
 doctorRequestsTable (class in doctor.models), 4  
 doctorRequestsTable.DoesNotExist, 4  
 doctorRequestsTable.MultipleObjectsReturned, 4  
 doctorrequeststable\_set (patient.models.Doctor attribute), 10  
 doctorrequeststable\_set (patient.models.Patient attribute), 13  
 doctorshowhistory() (in module doctor.views), 5  
 doctorSignup() (in module doctor.views), 5

## G

gender (patient.models.Doctor attribute), 10  
 gender (patient.models.Patient attribute), 13

generic\_name (patient.models.Medication attribute), 11  
 get\_next\_by\_date() (patient.models.Basicmetabolismpanel method), 8  
 get\_next\_by\_date() (patient.models.Commonbloodcount method), 9  
 get\_next\_by\_date() (patient.models.Mri method), 12  
 get\_next\_by\_date() (patient.models.Patient method), 14  
 get\_next\_by\_date() (patient.models.Prescription method), 16  
 get\_next\_by\_date() (patient.models.Xray method), 17  
 get\_previous\_by\_date() (patient.models.Basicmetabolismpanel method), 8  
 get\_previous\_by\_date() (patient.models.Commonbloodcount method), 9  
 get\_previous\_by\_date() (patient.models.Mri method), 12  
 get\_previous\_by\_date() (patient.models.Patient method), 14  
 get\_previous\_by\_date() (patient.models.Prescription method), 16  
 get\_previous\_by\_date() (patient.models.Xray method), 17  
 glucose (patient.models.Basicmetabolismpanel attribute), 8  
 grantedRequests() (in module patient.views), 19

## H

hematocrit (patient.models.Commonbloodcount attribute), 9  
 hemoglobin (patient.models.Commonbloodcount attribute), 9

## I

id (Account.models.UserProfile attribute), 1  
 id (doctor.models.doctorRequestsTable attribute), 4  
 id (lab.models.LabTest attribute), 6  
 id (patient.models.Basicmetabolismpanel attribute), 8  
 id (patient.models.Commonbloodcount attribute), 9  
 id (patient.models.Mri attribute), 12  
 id (patient.models.patientGrantTable attribute), 18  
 id (patient.models.Prescriptmed attribute), 17  
 id (patient.models.Xray attribute), 17  
 index() (in module doctor.views), 5  
 index() (in module lab.views), 7  
 index() (in module patient.views), 19

## L

lab (module), 7  
 lab.admin (module), 6  
 lab.apps (module), 6  
 lab.models (module), 6  
 lab.tests (module), 7

lab.urls (module), 7  
 lab.views (module), 7  
 LabConfig (class in lab.apps), 6  
 LabTest (class in lab.models), 6  
 LabTest.DoesNotExist, 6  
 LabTest.MultipleObjectsReturned, 6  
 labtest\_set (patient.models.Patient attribute), 14  
 log() (in module lab.views), 7  
 login() (in module Account.views), 2  
 login() (in module doctor.views), 5  
 login() (in module patient.views), 19  
 logout() (in module Account.views), 2  
 logout() (in module doctor.views), 5  
 logout() (in module patient.views), 19  
 lymphs (patient.models.Commonbloodcount attribute), 9

## M

main() (in module manage), 7  
 manage (module), 7  
 mch (patient.models.Commonbloodcount attribute), 9  
 mchc (patient.models.Commonbloodcount attribute), 9  
 mcv (patient.models.Commonbloodcount attribute), 9  
 Medication (class in patient.models), 11  
 Medication.DoesNotExist, 11  
 Medication.MultipleObjectsReturned, 11  
 medid (patient.models.Medication attribute), 11  
 medid (patient.models.Prescriptmed attribute), 17  
 medid\_id (patient.models.Prescriptmed attribute), 17  
 monocytes (patient.models.Commonbloodcount attribute), 9  
 Mri (class in patient.models), 12  
 mri (patient.models.Mri attribute), 12  
 Mri.DoesNotExist, 12  
 Mri.MultipleObjectsReturned, 12  
 mri\_set (patient.models.Patient attribute), 14

## N

name (Account.apps.AccountConfig attribute), 1  
 name (doctor.apps.DoctorConfig attribute), 4  
 name (lab.apps.LabConfig attribute), 6  
 name (patient.apps.PatientConfig attribute), 7  
 name (patient.models.Doctor attribute), 10  
 name (patient.models.Medication attribute), 12  
 name (patient.models.Patient attribute), 14  
 neutrophils (patient.models.Commonbloodcount attribute), 9

## O

objects (Account.models.UserProfile attribute), 2  
 objects (doctor.models.doctorRequestsTable attribute), 4  
 objects (lab.models.LabTest attribute), 6  
 objects (patient.models.Basicmetabolismpanel attribute), 8  
 objects (patient.models.Commonbloodcount attribute), 9

objects (patient.models.Doctor attribute), 11  
 objects (patient.models.Medication attribute), 12  
 objects (patient.models.Mri attribute), 12  
 objects (patient.models.Patient attribute), 14  
 objects (patient.models.patientGrantTable attribute), 18  
 objects (patient.models.Prescription attribute), 16  
 objects (patient.models.Prescriptmed attribute), 17  
 objects (patient.models.Xray attribute), 17

## P

Patient (class in patient.models), 12  
 patient (module), 19  
 patient.admin (module), 7  
 patient.apps (module), 7  
 Patient.DoesNotExist, 13  
 patient.models (module), 8  
 Patient.MultipleObjectsReturned, 13  
 patient.tests (module), 19  
 patient.urls (module), 19  
 patient.views (module), 19  
 patient\_set (Account.models.UserProfile attribute), 2  
 PatientConfig (class in patient.apps), 7  
 patientGrant() (in module patient.views), 19  
 patientGrantTable (class in patient.models), 18  
 patientGrantTable.DoesNotExist, 18  
 patientGrantTable.MultipleObjectsReturned, 18  
 patientgranttable\_set (patient.models.Doctor attribute), 11  
 patientgranttable\_set (patient.models.Patient attribute), 14  
 patientSignup() (in module patient.views), 19  
 pid (doctor.models.doctorRequestsTable attribute), 4  
 pid (lab.models.LabTest attribute), 6  
 pid (patient.models.Basicmetabolismpanel attribute), 8  
 pid (patient.models.Commonbloodcount attribute), 9  
 pid (patient.models.Mri attribute), 12  
 pid (patient.models.Patient attribute), 14  
 pid (patient.models.patientGrantTable attribute), 18  
 pid (patient.models.Prescription attribute), 16  
 pid (patient.models.Xray attribute), 17  
 pid\_id (doctor.models.doctorRequestsTable attribute), 4  
 pid\_id (lab.models.LabTest attribute), 6  
 pid\_id (patient.models.Basicmetabolismpanel attribute), 8  
 pid\_id (patient.models.Commonbloodcount attribute), 10  
 pid\_id (patient.models.Mri attribute), 12  
 pid\_id (patient.models.patientGrantTable attribute), 18  
 pid\_id (patient.models.Prescription attribute), 16  
 pid\_id (patient.models.Xray attribute), 18  
 platelets (patient.models.Commonbloodcount attribute), 10  
 postal (patient.models.Patient attribute), 14  
 potassium (patient.models.Basicmetabolismpanel attribute), 8

prescript\_id (patient.models.Prescription attribute), 16  
 prescript\_id (patient.models.Prescriptmed attribute), 17  
 prescript\_id\_id (patient.models.Prescriptmed attribute), 17  
 Prescription (class in patient.models), 15  
 Prescription.DoesNotExist, 15  
 Prescription.MultipleObjectsReturned, 15  
 prescription\_set (patient.models.Doctor attribute), 11  
 prescription\_set (patient.models.Patient attribute), 14  
 Prescriptmed (class in patient.models), 16  
 Prescriptmed.DoesNotExist, 16  
 Prescriptmed.MultipleObjectsReturned, 16  
 prescriptmed\_set (patient.models.Medication attribute), 12  
 prescriptmed\_set (patient.models.Prescription attribute), 16  
 profile\_id (patient.models.Doctor attribute), 11  
 profile\_id (patient.models.Patient attribute), 15  
 profile\_id\_id (patient.models.Doctor attribute), 11  
 profile\_id\_id (patient.models.Patient attribute), 15  
 profile\_type (Account.models.UserProfile attribute), 2

## R

rbc (patient.models.Commonbloodcount attribute), 10  
 rdw (patient.models.Commonbloodcount attribute), 10  
 register() (in module Account.views), 2  
 requestLab() (in module patient.views), 19

## S

saveprescription() (in module doctor.views), 5  
 saveprescriptionmedicine() (in module doctor.views), 5  
 showHistory() (in module patient.views), 19  
 sodium (patient.models.Basicmetabolismpanel attribute), 8  
 state (patient.models.Patient attribute), 15  
 street\_address (patient.models.Patient attribute), 15  
 surname (patient.models.Doctor attribute), 11  
 surname (patient.models.Patient attribute), 15

## T

test (lab.models.LabTest attribute), 6  
 test() (in module lab.views), 7

## U

user (Account.models.UserProfile attribute), 2  
 user\_id (Account.models.UserProfile attribute), 2  
 UserProfile (class in Account.models), 1  
 UserProfile.DoesNotExist, 1  
 UserProfile.MultipleObjectsReturned, 1

## W

wbc (patient.models.Commonbloodcount attribute), 10

## X

Xray (class in patient.models), 17

xray (patient.models.Xray attribute), 18

Xray.DoesNotExist, 17

Xray.MultipleObjectsReturned, 17

xray\_set (patient.models.Patient attribute), 15