



# ფინალური დავალება – ავეჯის ონლაინ მაღაზია

## Django Framework-ის გამოყენებით



### პროექტის მიზანი

შექმენით ონლაინ ავეჯის მაღაზია Django Framework-ის და Django REST Framework-ის გამოყენებით, რომელიც მოიცავს ყველა ძირითად ფუნქციონალს: პროდუქტების კატალოგს, მომხმარებლების რეგისტრაციას, კალათას და შეკვეთების მართვას.

---

## მოდელები

### 1. Category მოდელი

შექმენით კატეგორიების მოდელი შემდეგი ველებით:

სავალდებულო ველები:

- **name** – კატეგორიის სახელი
- **slug** – URL-ისთვის გამოსაყენებელი slug (უნიკალური)
- **description** – კატეგორიის მოკლე აღწერა
- **image** – სურათი (ბანერი)
- **is\_active** – აქტიურობის სტატუსი
- **created\_at** – შექმნის თარიღი

აუცილებელი კატეგორიები:

1. სკამი (Chair)
2. დივანი (Sofa)
3. მაგიდა (Table)
4. კარადა (Wardrobe)
5. საწოლი (Bed)
6. ტუმბო (Cabinet/Nightstand)
7. თარო (Shelf)
8. სავარძელი (Armchair)

## 9. აივნის/ეზოს ავეჯი (Outdoor Furniture)

---

### 2. Product მოდელი

შექმენით პროდუქტების მოდელი შემდეგი ველებით:

ძირითადი ველები:

- **name** – პროდუქტის სახელი
- **slug** – პროდუქტის slug (უნიკალური)
- **category** – ForeignKey → Category
- **description** – პროდუქტის დეტალური აღწერა
- **price** – ფასი
- **stock** – მარაგის რაოდენობა
- **is\_available** – ხელმისაწვდომობა
- **featured** – გამორჩეული პროდუქტი
- **created\_at, updated\_at** – თარიღები

ატრიბუტები (choices):

- **color** – ფერი (მაგ: თეთრი, შავი, ყავისფერი, ნაცრისფერი, ბეჟი)
- **material** – მასალა (მაგ: ხე, ლითონი, მინა, ტყავი, ტექსტილი, პლასტიკი)

სურათები:

- დამატებითი ფოტოები (2, 3, 4 – optional)
- 

### 3. CustomUser

მომხმარებლის პროფილისთვის

ველები:

- **user** – OneToOne User მოდელზე
- **first\_name** – სახელი
- **last\_name** – სახელი

- `phone` - მობილურის ნომერი
- `address` - მისამართი
- `birth_date` - დაბადების თარიღი

მეთოდები:

- `get_full_name()` - სახელი + გვარი
- 

## 4. Cart და CartItem

Cart მოდელი:

- `user` - OneToOne → CustomUser
- `updated_at` - განახლების თარიღი

CartItem მოდელი:

- `cart` - ForeignKey → Cart
- `product` - ForeignKey → Product
- `quantity` - რაოდენობა

Cart მეთოდები:

- `get_total_price()` - ჯამური ფასი
  - `get_total_items()` - კალათაში არსებული პროდუქტების სია
  - `get_total_items_count()` - კალათაში არსებული პროდუქტების რაოდენობა
- 

## 5. Order და OrderItem

Order მოდელი:

- `user` - ForeignKey → CustomUser
- `order_number` - უნიკალური ნომერი
- `status` - pending, processing, shipped, delivered, cancelled
- `total_amount` - (total price - DecimalField)
- `shipping_address`
- `phone`
- `notes` (optional)

- `created_at`
- `updated_at`

**OrderItem მოდელი:**

- `order` – ForeignKey → Order
- `product` – ForeignKey → Product
- `quantity` – რაოდენობა
- `price` – შეკვეთის მომენტში (აუცილებელია სტატიკურად შევინახოთ რა ფასად გაიყიდა პროდუქტი)

---

## Django Admin კონფიგურაცია

აუცილებელია, რომ **Django Admin** იყოს სრულად გამართული:

- **CategoryAdmin** → ძიება სახელით, ფილტრი აქტიურობის მიხედვით, `slug`-ის ავტომატური გენერაცია.
- **ProductAdmin** → ძიება სახელით, ფილტრი კატეგორიის, ფერის და მასალის მიხედვით, ფასისა და მარაგის რედაქტირება პირდაპირ სიიდან.
- **CustomUserAdmin** → ძიება ტელეფონისა და მომხმარებლის სახელით, ფილტრი ქალაქის მიხედვით.
- **CartAdmin** → ფილტრი მომხმარებლის მიხედვით, ასევე `tabular inline`-ის გამოყენებით მოდელის დეტალურზე დაამატეთ მისი მოკავშერე მოდელი **CartItem**
- **OrderAdmin** → ძიება შეკვეთის ნომრით და მომხმარებლის მიხედვით, ფილტრი სტატუსისა და თარიღის მიხედვით. ასევე `tabular inline`-ის გამოყენებით მოდელის დეტალურზე დაამატეთ მისი მოკავშერე მოდელი **OrderItem**

---

## API (Using Django Rest Framework)

პროექტს უნდა ქონდეს REST API (Django REST Framework), რომელიც უზრუნველყოფს ძირითადი მოდელების მენეჯმენტს:

**Endpoints:**

## Category

- `GET /api/categories/` – ყველა კატეგორია
- `GET /api/categories/<id>/` – კონკრეტული კატეგორია

## Product

- `GET /api/products/` – ყველა პროდუქტი
- `GET /api/products/<id>/` – კონკრეტული პროდუქტი
- ფილტრაცია შესაძლებელია კატეგორიით, ფერით, მასალით

## User (CustomUser)

- `POST /api/register/` – რეგისტრაცია
- `POST /api/login/` – ავტორიზაცია (JWT ან Token)
- `GET /api/profile/` – ავტორიზებული მომხმარებლის პროფილი
- პაროლის ცვლილების ფუნქციონალი – (Advanced)

## Cart

- `GET /api/cart/` – მომხმარებლის კალათა
- `POST /api/cart/add/` – პროდუქტის დამატება კალათაში
- `POST /api/cart/remove/` – პროდუქტის წაშლა კალათიდან

## Order

- `GET /api/orders/` – ყველა შეკვეთა (auth user)
- `GET /api/orders/<id>/` – კონკრეტული შეკვეთა
- `POST /api/orders/create/` – ახალი შეკვეთის შექმნა


---

## Celery Integration (Advanced)

1. **send\_order\_confirmation\_email** – შეკვეთის დადასტურების email მომხმარებლისთვის.
  2. **update\_order\_status** – შეკვეთის სტატუსის ავტომატური განახლება Pending → Processing-ზე გარკვეული დროის შემდეგ.
-

## წარსადგენი მასალა

1. **Django** პროექტი სრული კოდით
2. **requirements.txt** ყველა dependency-ით
3. **README.md** ინსტალაციის ინსტრუქციებით
4. მონაცემთა ბაზა (**db.sqlite3**) ტესტ მონაცემებით
5. მედია ფაილები (**media/** საქალაქადე)
6. სტატიკური საქალაქადე (**static/** საქალაქადე)

 ჩვეულებრივ Git-ში მედია ფაილები და ბაზა არ კომიტდება, მაგრამ ფინალური დავალების შემთხვევაში აუცილებელია დააკომიტო რეპოზიტორიაზე.