```python
from django.contrib.auth.base_user import AbstractBaseUser
from django.db import models


class User(AbstractBaseUser):
    """
    User model.
    """

    USERNAME_FIELD = "email"

    REQUIRED_FIELDS = ["first_name", "last_name"]

    email = models.EmailField(
        verbose_name="E-mail",
        unique=True
    )

    first_name = models.CharField(
        verbose_name="First name",
        max_length=30
    )

    last_name = models.CharField(
        verbose_name="Last name",
        max_length=40
    )

    city = models.CharField(
```

```python
        verbose_name="City",

        max_length=40

    )


    stripe_id = models.CharField(

        verbose_name="Stripe ID",

        unique=True,

        max_length=50,

        blank=True,

        null=True

    )


    objects = UserManager()


    @property

    def get_full_name(self):

        return f"{self.first_name} {self.last_name}"


    class Meta:

        verbose_name = "User"

        verbose_name_plural = "Users"



class Profile(models.Model):

    """

    User's profile.

    """


    phone_number = models.CharField(
```

```python
        verbose_name="Phone number",

        max_length=15

    )


    date_of_birth = models.DateField(

        verbose_name="Date of birth"

    )


    postal_code = models.CharField(

        verbose_name="Postal code",

        max_length=10,

        blank=True

    )


    address = models.CharField(

        verbose_name="Address",

        max_length=255,

        blank=True

    )


    class Meta:

        abstract = True


class UserProfile(Profile):

    """

    User's profile model.

    """
```

```python
    user = models.OneToOneField(
        to=User, on_delete=models.CASCADE, related_name="profile",
    )

    group = models.CharField(
        verbose_name="Group type",
        choices=GroupTypeChoices.choices(),
        max_length=20,
        default=GroupTypeChoices.EMPLOYEE.name,
    )

    def __str__(self):
        return self.user.email

    class Meta:


# user 1 - employer
user1, _ = User.objects.get_or_create(
    email="foo@bar.com",
    first_name="Employer",
    last_name="Testowy",
    city="Białystok",
)

user1.set_unusable_password()

group_name = "employer"

_profile1, _ = UserProfile.objects.get_or_create(
```

```python
        user=user1,

        date_of_birth=datetime.now() - timedelta(days=6600),

        group=GroupTypeChoices(group_name).name,

        address="Myśliwska 14",

        postal_code="15-569",

        phone_number="+48100200300",

    )


    # user2 - employee

    user2, _ = User.objects.get_or_create()

        email="bar@foo.com",

        first_name="Employee",

        last_name="Testowy",

        city="Białystok",

    )


    user2.set_unusable_password()


    group_name = "employee"


    _profile2, _ = UserProfile.objects.get_or_create()

        user=user2,

        date_of_birth=datetime.now() - timedelta(days=7600),

        group=GroupTypeChoices(group_name).name,

        address="Myśliwska 14",

        postal_code="15-569",

        phone_number="+48200300400",

    )
```

```python
response_customer = stripe.Customer.create(
    email=user.email,
    description=f"EMPLOYER - {user.get_full_name}",
    name=user.get_full_name,
    phone=user.profile.phone_number,
)


user1.stripe_id = response_customer.stripe_id
user1.save()


mcc_code, url = "1520", "https://www.softserveinc.com/"


response_ca = stripe.Account.create(
    type="custom",
    country="PL",
    email=user2.email,
    default_currency="pln",
    business_type="individual",
    settings={"payouts": {"schedule": {"interval": "manual", }}},
    requested_capabilities=["card_payments", "transfers", ],
    business_profile={"mcc": mcc_code, "url": url},
    individual={
        "first_name": user2.first_name,
        "last_name": user2.last_name,
        "email": user2.email,
        "dob": {
            "day": user2.profile.date_of_birth.day,
            "month": user2.profile.date_of_birth.month,
            "year": user2.profile.date_of_birth.year,
```

```python
        },
        "phone": user2.profile.phone_number,
        "address": {
            "city": user2.city,
            "postal_code": user2.profile.postal_code,
            "country": "PL",
            "line1": user2.profile.address,
        },
    },
)


user2.stripe_id = response_ca.stripe_id
user2.save()


tos_acceptance = {"date": int(time.time()), "ip": user_ip},


stripe.Account.modify(user2.stripe_id, tos_acceptance=tos_acceptance)


passport_front = stripe.File.create(
    purpose="identity_document",
    file=_file, # ContentFile object
    stripe_account=user2.stripe_id,
)


individual = {
    "verification": {
        "document": {"front": passport_front.get("id"),},
        "additional_document": {"front": passport_front.get("id"),},
    }
```

```python
}

stripe.Account.modify(user2.stripe_id, individual=individual)

new_card_source = stripe.Customer.create_source(user1.stripe_id, source=token)

stripe.SetupIntent.create(
    payment_method_types=["card"],
    customer=user1.stripe_id,
    description="some description",
    payment_method=new_card_source.id,
)

payment_method = stripe.Customer.retrieve(user1.stripe_id).default_source

payment_intent = stripe.PaymentIntent.create(
    amount=amount,
    currency="pln",
    payment_method_types=["card"],
    capture_method="manual",
    customer=user1.stripe_id, # customer
    payment_method=payment_method,
    application_fee_amount=application_fee_amount,
    transfer_data={"destination": user2.stripe_id}, # connect account
    description=description,
    metadata=metadata,
)
```

```python
payment_intent_confirm = stripe.PaymentIntent.confirm(
    payment_intent.stripe_id, payment_method=payment_method
)


stripe.PaymentIntent.capture(
    payment_intent.id, amount_to_capture=amount
)
stripe.Balance.retrieve(stripe_account=user2.stripe_id)


stripe.Charge.create(
    amount=amount,
    currency="pln",
    source=user2.stripe_id,
    description=description
)


stripe.PaymentIntent.cancel(payment_intent.id)



        unique_together = ("user", "group")
```