So wat i want to create is abit complicated  
let me show you the models.py for the vendors..  
**from** django.db **import** models

**from** django.contrib.auth.models **import** AbstractBaseUser, BaseUserManager, PermissionsMixin

**from** django.utils **import** timezone

**from** django.contrib.auth **import** get\_user\_model

**class** VendorManager(BaseUserManager):

**def** create\_user(self, username, email, first\_name, last\_name, storename, store\_phone, password**=None**):

**if** **not** email:

**raise** ValueError('Users must have an email address')

        user **=** self.model(

            username**=**username,

            email**=**self.normalize\_email(email),

            first\_name**=**first\_name,

            last\_name**=**last\_name,

            storename**=**storename,

            store\_phone**=**store\_phone,

        )

        user.set\_password(password)

        user.save(using**=**self.\_db)

**return** user

**def** create\_superuser(self, username, email, first\_name, last\_name, storename, store\_phone, password**=None**):

        user **=** self.create\_user(

            username,

            email,

            first\_name,

            last\_name,

            storename,

            store\_phone,

            password**=**password

        )

        user.is\_admin **=** **True**

        user.is\_superuser **=** **True**

        user.is\_staff **=** **True**

        user.save(using**=**self.\_db)

**return** user

**class** Vendor(AbstractBaseUser, PermissionsMixin):

    username **=** models.CharField(max\_length**=**30, unique**=True**)

    email **=** models.EmailField(unique**=True**)

    first\_name **=** models.CharField(max\_length**=**30)

    last\_name **=** models.CharField(max\_length**=**30)

    storename **=** models.CharField(max\_length**=**50)

    store\_phone **=** models.CharField(max\_length**=**15)

    verification\_code **=** models.CharField(max\_length**=**6, null**=True**, blank**=True**)

    verification\_code\_created\_at **=** models.DateTimeField(null**=True**, blank**=True**)

    is\_active **=** models.BooleanField(default**=True**)

    is\_admin **=** models.BooleanField(default**=False**)

    is\_superuser **=** models.BooleanField(default**=False**)

    is\_staff **=** models.BooleanField(default**=False**)

    is\_vendor **=** models.BooleanField(default**=True**)

    objects **=** VendorManager()

    USERNAME\_FIELD **=** 'username'

    REQUIRED\_FIELDS **=** ['email', 'first\_name', 'last\_name', 'storename', 'store\_phone']

**def** \_\_str\_\_(self):

**return** self.username

**def** has\_perm(self, perm, obj**=None**):

**return** **True**

**def** has\_module\_perms(self, app\_label):

**return** **True**

so this is my working models for vendors..  
no i want us to create a POS system for them but in an interesting way   
i want us to create sub accounts(these accounts should have a one to many relation ship with the vendor table in that one vendor can create more than one of these account and they are ubder that vendor entirely) they are to helo with POS.. selling tickets with cash and doing verification .. they are gonna have the privilledges only a vendir assigns to them but the system only gives them authority to generate tickets and verify them for the vendor who created them only and its up to a vendor to assign them to one event or to all events or so on .. so wen a vendor assigns them to event(s) on their dashboard they should be able to see that event and its information as per the event model in events/models.py   
**from** django.db **import** models

**from** django.conf **import** settings

**from** django.contrib.contenttypes.models **import** ContentType

**from** django.contrib.contenttypes.fields **import** GenericForeignKey

**from** vendors.models **import** Vendor  *# This remains as is*

**from** django.contrib.auth **import** get\_user\_model

**from** django.utils **import** timezone

**class** Event(models.Model):

    STATUS\_CHOICES **=** [

        ('pending', 'Pending'),

        ('approved', 'Approved'),

        ('rejected', 'Rejected'),

    ]

    vendor **=** models.ForeignKey(Vendor, on\_delete**=**models.CASCADE, related\_name**=**'events')

    poster **=** models.ImageField(upload\_to**=**'event\_posters/')

    title **=** models.CharField(max\_length**=**200)

    description **=** models.TextField()

    category **=** models.CharField(max\_length**=**100)

    start\_date **=** models.DateTimeField()

    end\_date **=** models.DateTimeField(null**=True**, blank**=True**)

    venue\_name **=** models.CharField(max\_length**=**100)

    regular\_price **=** models.DecimalField(max\_digits**=**10, decimal\_places**=**2)

    sale\_price **=** models.DecimalField(max\_digits**=**10, decimal\_places**=**2, null**=True**, blank**=True**)

    tickets\_available **=** models.PositiveIntegerField(null**=True**, blank**=True**)

    tickets\_sold **=** models.PositiveIntegerField(default**=**0)

    status **=** models.CharField(max\_length**=**10, choices**=**STATUS\_CHOICES, default**=**'pending')

    adminaction **=** models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete**=**models.SET\_NULL, null**=True**, blank**=True**, editable**=False**)  *# Stores the admin who approved/rejected*

**def** is\_sold\_out(self):

**return** self.tickets\_available **is** **not** **None** **and** self.tickets\_sold **>=** self.tickets\_available

**def** \_\_str\_\_(self):

**return** self.title

**class** TicketCategory(models.Model):

    event **=** models.ForeignKey(Event, on\_delete**=**models.CASCADE, related\_name**=**'ticket\_categories')  *# Mandatory, linked to Event*

    category\_title **=** models.CharField(max\_length**=**100)  *# Required field*

    category\_price **=** models.DecimalField(max\_digits**=**10, decimal\_places**=**2)  *# Required field*

    category\_tickets\_available **=** models.PositiveIntegerField()  *# Required field*

    category\_tickets\_sold **=** models.PositiveIntegerField(default**=**0)  *# New field to track tickets sold in this category*

**def** is\_category\_sold\_out(self):

**return** self.category\_tickets\_sold **>=** self.category\_tickets\_available

**def** \_\_str\_\_(self):

**return** **f**'{self.category\_title} - {self.event.title}'

and the only action they can make on the event is to generate rate ticket, or when the start date is the same as the current date the verification should be active (from midnight) so they will have ot be able to verify the tickets made by only these vendors they are under and those assigned to   
 but first of all lets first work on these accounts creation before assigning them their roles...  
 so in the vendor dashboard i wanna put a button called POS system specifically in this part

<div *class*="d-flex justify-content-between mb-4">

*<!-- Navigation Buttons on the Left -->*

            <div *class*="btn-group">

                <a *href*="{% url 'create\_event' %}" *class*="btn btn-outline-event btn-icon"><i *class*="fa fa-plus"></i> Create Event</a>

                <a *href*="{% url 'vendor\_events' %}" *class*="btn btn-outline-event btn-icon">View Events</a>

                <a *href*="{% url 'verifiable\_events' %}" *class*="btn btn-outline-event btn-icon">Event Verification</a>

                <a *href*="{% url 'view\_inventory' %}" *class*="btn btn-outline-event btn-icon">View Inventory</a>

                {% if user.is\_superuser %}

                <a *href*="{% url 'pending\_events' %}" *class*="btn btn-outline-event btn-icon">

                    Pending Events <span *class*="badge badge-danger">{{ pending\_count }}</span>

                </a>

                <a *href*="{% url 'events\_confirmed' %}" *class*="btn btn-outline-event btn-icon">Events Confirmed</a>

                {% endif %}

            </div>

*<!-- Profile Dropdown on the Right -->*

            <div *class*="dropdown">

                <button *class*="btn btn-outline-profile dropdown-toggle" *type*="button" *id*="profileDropdown" *aria-haspopup*="true" *aria-expanded*="false">

                    Profile

                </button>

                <div *class*="dropdown-menu" *aria-labelledby*="profileDropdown">

                    <a *class*="dropdown-item" *href*="{% url 'update\_vendor' %}">Update Profile</a>

                    <a *class*="dropdown-item" *href*="{% url 'change\_password' %}">Change Password</a>

                    <a *class*="dropdown-item" *href*="{% url 'logout' %}">Logout</a>

                </div>

            </div>

        </div>

And when this vendr taps on this button i want a page that will have information like(Welcome to our POS system, here we can enable u create accounts that can help you sell tickets with cash and also verifiy the tickets at the entrance .. tap the button and create as many accounts as u like when you press create account, an email containing the password and the their email should be sent to thtat email and they can log in and work as intended.. the accounts should always be active so long as the events haven’t passed and when an event assigned to them passes they will be inactive but you can activate them later on and reassign them to events )  
so wen u tap create account, i want a fields of First name, last name, email.. and then below we should have a list of all the upcoming events for this account with the status == approved listed with chenck boxes and they can tick atleast one event to assign to this account..  
 so when they have filled all that and pressed create account..  
the system should save that information, the system will send an email to that account telling them that they have been selected as the apos agents for these event “list them down that were selected ” by “vendor first name and the storename ” and use this email to login and this is the password (the system should generate a 6 digit code that they will use to authenticate this will all be saved under their model and also under which vendor since they have a one to many relationship..) these accounts should be allowed to change the password to what they feel like but not the email.., (i want a new Django app that will handle POS so the model, the admin.py and the view code is going there ) so on authentication the system had its own custom back end that goes like this   
**from** django.contrib.auth.backends **import** ModelBackend

**from** vendors.models **import** Vendor

**from** customers.models **import** Customer

**class** VendorOrCustomerModelBackend(ModelBackend):

**def** authenticate(self, request, username**=None**, password**=None**, **\*\***kwargs):

        is\_vendor\_login **=** kwargs.get('is\_vendor\_login', **False**)

**try**:

**if** is\_vendor\_login:

*# Authenticate as a vendor*

                vendor **=** Vendor.objects.get(username**=**username)

**if** vendor.check\_password(password) **and** vendor.is\_vendor:

**return** vendor

**else**:

*# Authenticate as a customer*

                customer **=** Customer.objects.get(username**=**username)

**if** customer.check\_password(password) **and** customer.is\_customer:

**return** customer

**except** (Vendor.DoesNotExist, Customer.DoesNotExist):

**return** **None**

**def** get\_user(self, user\_id):

**try**:

**return** Vendor.objects.get(pk**=**user\_id)

**except** Vendor.DoesNotExist:

**try**:

**return** Customer.objects.get(pk**=**user\_id)

**except** Customer.DoesNotExist:

**return** **None**