When creating a new user with the custom user model, you can modify the additional fields, such as `is\_premium\_member`, by providing the corresponding values during the creation process. You can achieve this in your view or serializer where you handle the user creation.

Here's an example of how you might modify the `create\_user` method in your custom manager (`CustomUserManager`) to accept the `is\_premium\_member` field:

```python

# models.py

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

from django.db import models

class CustomUserManager(BaseUserManager):

def create\_user(self, email, password=None, is\_premium\_member=False, \*\*extra\_fields):

if not email:

raise ValueError('The Email field must be set')

email = self.normalize\_email(email)

user = self.model(email=email, is\_premium\_member=is\_premium\_member, \*\*extra\_fields)

user.set\_password(password)

user.save(using=self.\_db)

return user

def create\_superuser(self, email, password=None, \*\*extra\_fields):

extra\_fields.setdefault('is\_staff', True)

extra\_fields.setdefault('is\_superuser', True)

return self.create\_user(email, password, \*\*extra\_fields)

class CustomUser(AbstractBaseUser,

PermissionsMixin):

email = models.EmailField(unique=True)

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

first\_name = models.CharField(max\_length=30, blank=True)

last\_name = models.CharField(max\_length=30, blank=True)

date\_joined = models.DateTimeField(auto\_now\_add=True)

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

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

is\_premium\_member = models.BooleanField(default=False)

objects = CustomUserManager()

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = ['username']

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

return self.email

```

Then, when you create a new user instance in your view or serializer, you can pass the `is\_premium\_member` value:

```python

# views.py or serializers.py

from .models import CustomUser

# ...

class YourUserCreationViewOrSerializer:

# ...

def create\_user(self, validated\_data):

is\_premium\_member = validated\_data.pop('is\_premium\_member', False)

# Create the user with the remaining data

user = CustomUser.objects.create\_user(\*\*validated\_d

ata, is\_premium\_member=is\_premium\_member)

# ... rest of your code ...

```

Now, when you create a new user, you can pass the `is\_premium\_member` value to indicate whether the user should be a premium member or not. Make sure to adjust your views or serializers accordingly.