Django Custom User

| In Django, the `authenticate` function is used to validate the credentials of a user, but it expects the default `username` and `password` fields for authentication. If you have extended or customized the user model (such as using `pid` instead of `username`), you'll need to make sure that Django's authentication system is aware of these custom fields.  Here's a step-by-step guide to address the issues in your code:  ### 1. \*\*Custom User Model\*\*  If you've created a custom user model with a `pid` field instead of the default `username`, you need to ensure that Django's authentication system is configured to use this custom model.  ### Custom User Model Example  ```python  from django.contrib.auth.models import AbstractBaseUser, BaseUserManager  from django.db import models  class CustomUserManager(BaseUserManager):  def create\_user(self, pid, password=None, \*\*extra\_fields):  if not pid:  raise ValueError('The PID field must be set')  user = self.model(pid=pid, \*\*extra\_fields)  user.set\_password(password)  user.save(using=self.\_db)  return user  def create\_superuser(self, pid, password=None, \*\*extra\_fields):  extra\_fields.setdefault('is\_staff', True)  extra\_fields.setdefault('is\_superuser', True)  return self.create\_user(pid, password, \*\*extra\_fields)  class Users(AbstractBaseUser):  pid = models.CharField(max\_length=30, unique=True)  # Additional fields  is\_active = models.BooleanField(default=True)  is\_staff = models.BooleanField(default=False)  objects = CustomUserManager()  USERNAME\_FIELD = 'pid'  REQUIRED\_FIELDS = [] # Add any additional fields required for superuser creation  ```  ### 2. \*\*Configure Authentication Backends\*\*  Make sure your custom user model is used by Django's authentication system. In your `settings.py` file, specify the custom user model:  ```python  AUTH\_USER\_MODEL = 'yourapp.Users'  ```  And, if you need to customize the authentication backend to work with the `pid` field, you can create a custom authentication backend:  ### Custom Authentication Backend  Create a file `authentication.py` in your app:  ```python  from django.contrib.auth.backends import BaseBackend  from .models import Users  class CustomBackend(BaseBackend):  def authenticate(self, request, pid=None, password=None, \*\*kwargs):  try:  user = Users.objects.get(pid=pid)  if user.check\_password(password):  return user  except Users.DoesNotExist:  return None  def get\_user(self, user\_id):  try:  return Users.objects.get(pk=user\_id)  except Users.DoesNotExist:  return None  ```  Add this backend to your `settings.py`:  ```python  AUTHENTICATION\_BACKENDS = [  'yourapp.authentication.CustomBackend',  'django.contrib.auth.backends.ModelBackend', # Keep this if you need the default backend as well  ]  ```  ### 3. \*\*Update Your View\*\*  Your view should now use the custom `authenticate` method correctly:  ```python  from django.shortcuts import render, redirect  from django.contrib import messages  from django.contrib.auth import authenticate, login as auth\_login  from django.http import HttpResponse  def index(request):  return HttpResponse("Hello To Prodvi")  def login\_page(request):  if request.method == "POST":  pid = request.POST.get("pid")  password = request.POST.get("password")  user = authenticate(request, pid=pid, password=password)  if user is None:  messages.error(request, 'Invalid Credentials')  return redirect('users:login') # Use the named URL pattern    auth\_login(request, user)  return redirect('users:index') # Use the named URL pattern  return render(request, 'users/login.html')  ```  ### Summary  1. \*\*Custom User Model\*\*: Ensure your user model uses a custom field (e.g., `pid`).  2. \*\*Authentication Backend\*\*: Implement a custom backend to handle authentication with the `pid` field.  3. \*\*Configure Settings\*\*: Update `settings.py` to use your custom user model and backend.  4. \*\*Update Views\*\*: Ensure your `authenticate` function and login process are correctly integrated.  By following these steps, you should be able to get Django’s authentication system working with your custom user model. If you run into further issues, make sure to check Django’s logs for any errors or exceptions that might give more insight into what’s going wrong. |
| --- |

Object Fields

Note: there are basic. Subset of these and others are also there

| Certainly! Here’s a detailed list of Django model fields along with their typical arguments and how you might use them in practice:  ### Basic Field Types with Arguments  1. \*\*CharField\*\*  - \*\*Description\*\*: For small to medium-sized strings.  - \*\*Arguments\*\*:  - `max\_length`: Required. The maximum length of the field.  - `blank`: Boolean. Whether the field is allowed to be empty in forms.  - `null`: Boolean. Whether the field can store `NULL` values in the database.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  name = models.CharField(max\_length=100, blank=False, null=False, default='Unnamed')  ```  2. \*\*TextField\*\*  - \*\*Description\*\*: For large text fields.  - \*\*Arguments\*\*:  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  description = models.TextField(blank=True, null=True)  ```  3. \*\*IntegerField\*\*  - \*\*Description\*\*: For integer values.  - \*\*Arguments\*\*:  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - `validators`: List of validators to apply to the field.  - \*\*Example\*\*:  ```python  age = models.IntegerField(default=0, validators=[MinValueValidator(0)])  ```  4. \*\*FloatField\*\*  - \*\*Description\*\*: For floating-point numbers.  - \*\*Arguments\*\*:  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - `validators`: List of validators to apply to the field.  - \*\*Example\*\*:  ```python  price = models.FloatField(default=0.0)  ```  5. \*\*DecimalField\*\*  - \*\*Description\*\*: For fixed-point decimal numbers.  - \*\*Arguments\*\*:  - `max\_digits`: The total number of digits.  - `decimal\_places`: The number of decimal places.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  salary = models.DecimalField(max\_digits=10, decimal\_places=2, default=0.00)  ```  6. \*\*BooleanField\*\*  - \*\*Description\*\*: For True/False values.  - \*\*Arguments\*\*:  - `default`: The default value (True or False).  - `blank`: Boolean. Whether the field is allowed to be empty.  - \*\*Example\*\*:  ```python  is\_active = models.BooleanField(default=True)  ```  7. \*\*DateField\*\*  - \*\*Description\*\*: For date values.  - \*\*Arguments\*\*:  - `auto\_now`: Boolean. If `True`, the field is updated every time the object is saved.  - `auto\_now\_add`: Boolean. If `True`, the field is set to the current date when the object is created.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  birth\_date = models.DateField(auto\_now\_add=True)  ```  8. \*\*DateTimeField\*\*  - \*\*Description\*\*: For date and time values.  - \*\*Arguments\*\*:  - `auto\_now`: Boolean. If `True`, the field is updated every time the object is saved.  - `auto\_now\_add`: Boolean. If `True`, the field is set to the current date and time when the object is created.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  created\_at = models.DateTimeField(auto\_now\_add=True)  ```  9. \*\*TimeField\*\*  - \*\*Description\*\*: For time values.  - \*\*Arguments\*\*:  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  start\_time = models.TimeField(default='00:00:00')  ```  ### Advanced Field Types with Arguments  10. \*\*EmailField\*\*  - \*\*Description\*\*: For email addresses.  - \*\*Arguments\*\*:  - `max\_length`: Maximum length of the field.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  email = models.EmailField(max\_length=254, blank=False, null=False)  ```  11. \*\*URLField\*\*  - \*\*Description\*\*: For URLs.  - \*\*Arguments\*\*:  - `max\_length`: Maximum length of the URL.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  website = models.URLField(max\_length=200, blank=True)  ```  12. \*\*SlugField\*\*  - \*\*Description\*\*: For URL-friendly strings.  - \*\*Arguments\*\*:  - `max\_length`: Maximum length of the field.  - `unique`: Boolean. Whether the field should be unique across the model.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  slug = models.SlugField(max\_length=50, unique=True)  ```  13. \*\*FileField\*\*  - \*\*Description\*\*: For file uploads.  - \*\*Arguments\*\*:  - `upload\_to`: Directory path where the file will be uploaded.  - `max\_length`: Maximum length of the file path.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  resume = models.FileField(upload\_to='resumes/', blank=True, null=True)  ```  14. \*\*ImageField\*\*  - \*\*Description\*\*: For image uploads.  - \*\*Arguments\*\*:  - `upload\_to`: Directory path where the image will be uploaded.  - `height\_field`: Optional. Field to store the height of the image.  - `width\_field`: Optional. Field to store the width of the image.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - `default`: The default value for the field.  - \*\*Example\*\*:  ```python  profile\_picture = models.ImageField(upload\_to='profile\_pics/', blank=True, null=True)  ```  15. \*\*ForeignKey\*\*  - \*\*Description\*\*: For many-to-one relationships.  - \*\*Arguments\*\*:  - `to`: The model to which this field is related.  - `on\_delete`: The behavior to adopt when the referenced object is deleted.  - `related\_name`: Name of the reverse relation from the related object back to this one.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - \*\*Example\*\*:  ```python  author = models.ForeignKey('auth.User', on\_delete=models.CASCADE, related\_name='posts')  ```  16. \*\*ManyToManyField\*\*  - \*\*Description\*\*: For many-to-many relationships.  - \*\*Arguments\*\*:  - `to`: The model to which this field is related.  - `through`: Optional. The model to use for the intermediate table.  - `related\_name`: Name of the reverse relation from the related object back to this one.  - `blank`: Boolean. Whether the field is allowed to be empty.  - `null`: Boolean. Whether the field can store `NULL` values.  - \*\*Example\*\*:  ```python  tags = models.ManyToManyField('Tag', blank=True, related\_name='posts')  ```  17. \*\*OneToOneField\*\*  - \*\*Description\*\*: For one-to-one relationships.  - \*\*Arguments\*\*:  - `to`: The model to which this field is related.  - `on\_delete`: The behavior to adopt when the referenced object is deleted.  - `related\_name`: Name of the reverse relation from the related object back to this one.  - `blank`: Boolean. Whether the |
| --- |