Agenda of the week:

Django framework has its own users and admin panels. But to develop our custom models we have some steps to follow thoroughly:

Steps to follow the custom model:

First develop the custom model in models.py

```
class MyAccountManager(BaseUserManager):
    def create_user(self, email, username, password):
        if not email:
            raise ValueError("Users must have an email address")
        if not username:
            raise ValueError("Users must have an username address")

        user  = self.model(
                email=self.normalize_email(email),
                username=username,
            )

        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, username, password):
        user =  self.create_user(
                email=self.normalize_email(email),
                password=password,
                username=username,
        )
        user.is_admin = True
        user.is_staff = True
        user.is_superuser = False
        user.save(using=self._db)
        return user

    class Meta:
        managed = False
        db_table = 'auth_user'


class AuthUser(AbstractBaseUser):
    id = models.AutoField(primary_key=True)
    password = models.CharField(max_length=128)
    last_login = models.DateTimeField(blank=True, null=True)
    # is_superuser = models.IntegerField()
    is_superuser = models.BooleanField(default=False)
    username = models.CharField(unique=True, max_length=150)
    first_name = models.CharField(max_length=150)
```

```
    last_name = models.CharField(max_length=150)
    email = models.CharField(max_length=254)
    is_staff = models.BooleanField(default=True)
    is_active = models.BooleanField(default=True)
    date_joined = models.DateTimeField(default=now, blank=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username,']

    objects = MyAccountManager()

    def __str__(self):
        return self.username


    class Meta:
        managed = False
        db_table = 'auth_user'
        ordering = ['id']

class AuthGroup(models.Model):
    name = models.CharField(unique=True, max_length=150)

    class Meta:
        managed = False
        db_table = 'auth_group'

class AuthUserGroups(models.Model):
    id = models.BigAutoField(primary_key=True)
    user = models.ForeignKey(AuthUser, models.DO_NOTHING)
    group = models.ForeignKey(AuthGroup, models.DO_NOTHING)

    class Meta:
        managed = False
        db_table = 'auth_user_groups'
        unique_together = (('user', 'group'),)
```

in views.py add custom view:

```
def registration_view(request):
    context = {}
    if request.POST:
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            email = form.cleaned_data.get('email')
            raw_password = form.cleaned_data.get('password1')
            account = authenticate(email=email, password=raw_password)
```

```
            user = AuthUser.objects.latest('id')
            user1= User.objects.get(username = user).pk
            g=AuthUserGroups(user_id=user1, group_id='1')
            g.save()
            return redirect('/showform/')
        else:
            context['registration_form'] = form
    else:
        form = RegistrationForm()
        context['registration_form'] = form
    return render(request, 'register.html',context)
```

As per the model it need, custom fields for the login system:

```
class RegistrationForm(UserCreationForm):
    email = forms.EmailField(max_length=60, help_text='Add Valid Email
Address')

    class Meta:
        model  = AuthUser
        fields = ("email", "username", "password1", "password2")
```