```
#1.
#validators.py
def validate_name(value):
    for ch in value:
        if not (ch.isalpha() or ch.isspace()):
            raise ValidationError('Name can only contain letters and spaces')

def validate_phone_number(value):
    if not re.match(r'^\+359\d{9}$', value):
        raise ValidationError("Phone number must start with a '+359' followed by 9
digits")


class Customer(models.Model):
    name = models.CharField(
        max_length=100,
        validators=[
            validate_name,
        ],
    )
    age = models.PositiveIntegerField(
        validators=[
            MinValueValidator(18, message='Age must be greater than 18'),
        ],
    )
    email = models.EmailField(
        error_messages={'invalid': 'Enter a valid email address'}
    )
    phone_number = models.CharField(
        max_length=13,
        validators=[
            validate_phone_number,
        ],
    )
    website_url = models.URLField(
        error_messages={'invalid': 'Enter a valid URL'}
    )


#2.
class BaseMedia(models.Model):

    class Meta:
        abstract = True
        ordering = ['-created_at', 'title']

    title = models.CharField(
        max_length=100,
    )
    description = models.TextField()
    genre = models.CharField(
        max_length=50,
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
    )


class Book(BaseMedia):
    class Meta(BaseMedia.Meta):
        verbose_name = 'Model Book'
        verbose_name_plural = 'Models of type - Book'

    author = models.CharField(
        max_length=100,
        validators=[
            MinLengthValidator(5, message='Author must be at least 5 characters
long'),
        ]
    )
    isbn = models.CharField(
        max_length=20,
        unique=True,
        validators=[
            MinLengthValidator(6, message='ISBN must be at least 6 characters
long'),
        ]
    )



class Movie(BaseMedia):

    class Meta(BaseMedia.Meta):
        verbose_name = 'Model Movie'
        verbose_name_plural = 'Models of type - Music'

    director = models.CharField(
        max_length=100,
        validators=[
            MinLengthValidator(8, message='Director must be at least 8 characters
long'),
        ],
    )


class Music(BaseMedia):

    class Meta(BaseMedia.Meta):
        verbose_name = 'Model Music'
        verbose_name_plural = 'Models of type - Music'


    artist = models.CharField(
        max_length=100,
        validators=[
            MinLengthValidator(9, message='Artist must be at least 9 characters
long'),
        ],
    )

#3.
class Product(models.Model):
    name = models.CharField(
        max_length=100,
    )
    price = models.DecimalField(
        decimal_places=2,
        max_digits=10,
    )

    def calculate_tax(self):
        return self.price * Decimal(0.08)

    @staticmethod
    def calculate_shipping_cost(weight):
        return weight * Decimal(2.00)

    def format_product_name(self):
        return f'Product: {self.name}'


class DiscountedProduct(Product):
    class Meta:
        proxy = True

    def calculate_price_without_discount(self):
        return self.price * Decimal(1.20)

    def calculate_tax(self):
        return self.price * Decimal(0.05)

    @staticmethod
    def calculate_shipping_cost(weight):
        return weight * Decimal(1.50)

    def format_product_name(self):
        return f'Discounted Product: {self.name}'


#4.
#mixins.py

class RechargeEnergyMixin:
    def recharge_energy(self, amount):
        self.energy += amount

        if self.energy > 100:
            self.energy = 100

        self.save()

      #option2:
      def recharge_energy(self, amount):
            self.energy = min(self.energy + amount, 100)
            self.save()


class Hero(models.Model, RechargeEnergyMixin):
    name = models.CharField(
        max_length=100,
    )
    hero_title = models.CharField(
        max_length=100,
    )
    energy = models.PositiveIntegerField()
```

```
class SpiderHero(Hero):
    class Meta:
        proxy = True

    def swing_from_buildings(self):
        self.energy -= 80

        if self.energy > 0:
            self.save()
            return f'{self.name} as Spider Hero swings from buildings using web
shooters'

        return f'{self.name} as Spider Hero is out of web shooter fluid'


class FlashHero(Hero):
    class Meta:
        proxy = True

    def run_at_super_speed(self):
        self.energy -= 65

        if self.energy > 0:
            self.save()
            return f'{self.name} as Flash Hero runs at lightning speed, saving the
day'

        return f'{self.name} as Flash Hero needs to recharge the speed force'


#5.
class Document(models.Model):
    class Meta:
        indexes = [
            models.Index(fields=['searched_vector'])
        ]
    title = models.CharField(
        max_length=200,
    )
    content = models.TextField()
    search_vector = SearchVectorField(
        null=True,
    )
```