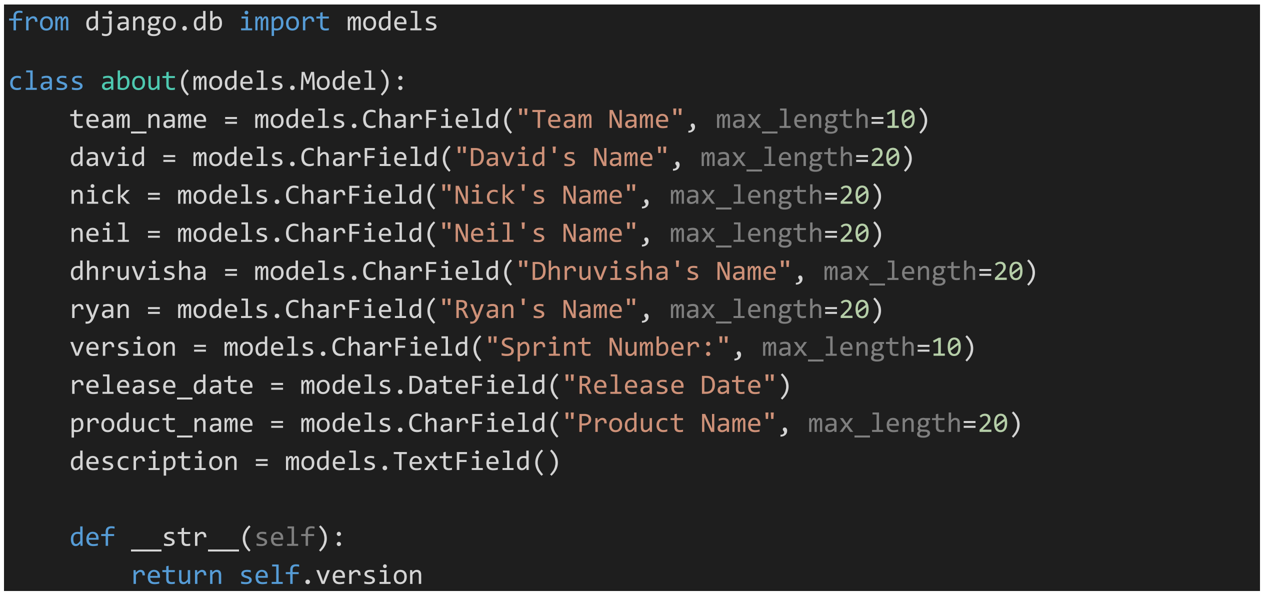
Recreating the Database

Preface: Django allows its users to define the database tables within files called models.py. The django.db module contains a submodule called models. Models are used to define the structure of your database tables, including fields and their data types.

1. **Creating the Django Project**
   a. Set up the virtual environment (python -m venv <name>).
   b. Activate the virtual environment (source <name>/Scripts/activate).
   c. Install Django and other necessary packages using pip install.
   d. Start the project (django-admin startproject <name>0.
2. **Creating a New App**
   a. Type python manage.py startapp <name>
   b. Open settings.py
   c. In INSTALLED_APPS, add the app name.
   d. Add urls.py to the app.
   e. Add "from . import views" to the file.
3. **Creating the models.**
   a. **About**



```
from django.db import models

class about(models.Model):
    team_name = models.CharField("Team Name", max_length=10)
    david = models.CharField("David's Name", max_length=20)
    nick = models.CharField("Nick's Name", max_length=20)
    neil = models.CharField("Neil's Name", max_length=20)
    dhruvisha = models.CharField("Dhruvisha's Name", max_length=20)
    ryan = models.CharField("Ryan's Name", max_length=20)
    version = models.CharField("Sprint Number:", max_length=10)
    release_date = models.DateField("Release Date")
    product_name = models.CharField("Product Name", max_length=20)
    description = models.TextField()

    def __str__(self):
        return self.version
```

b. **Application**

```
from django.db import models

from members.models import DriverProfile
from django.core.exceptions import ValidationError
from django.core.validators import RegexValidator

# Define the custom validator functions
def validate_phone_number(value):
    if not value.isdigit():
        raise ValidationError("Phone number must contain only numeric
characters.")

# Define a RegexValidator for the phone number format
phone_number_validator = RegexValidator(
    regex=r'^\d{10}$',
    message="Phone number must be exactly 10 digits long.",
)
class Application(models.Model):
    driver = models.ForeignKey(DriverProfile, on_delete=models.SET_NULL,
null=True)
    sponsor_name = models.CharField(max_length=50, blank=False)
    first_name = models.CharField(max_length=30, blank=False)
    last_name = models.CharField(max_length=30, blank=False)
    middle_initial = models.CharField(max_length=1, blank=True)
    email = models.EmailField(max_length=75, blank=False)
    phone = models.CharField(max_length=10, validators=[validate_phone_number,
phone_number_validator])

    street_address = models.CharField(max_length= 85, blank=False)
    city = models.CharField(max_length=40, blank=False)
    state = models.CharField(max_length=2, blank=False)
    zipcode = models.CharField(max_length=5, blank=False)

    license_num = models.CharField(max_length=30, blank=False)
    plate_num = models.CharField(max_length=10, blank=False)
    year = models.PositiveIntegerField(null=False)
    make = models.CharField(max_length=25, blank=False)
    model = models.CharField(max_length=25, blank=False)
    vin = models.CharField(max_length=17, blank=False)
    provider_name = models.CharField(max_length=100, blank=False)
    policy_number = models.CharField(max_length=50, blank=False)
    date_created = models.DateTimeField(auto_now=True)
    application_reason = models.TextField(max_length=250, blank=True)
    is_open = models.BooleanField('application status', default=True)
    is_approved = models.BooleanField('approval status', default=False)
    is_waitlisted = models.BooleanField('waitlist status', default=False)


    def __str__(self):
        return f"{self.driver.user.username}'s application to
{self.sponsor_name}"
```

c. **Reports**

```
from django.db import models


# Create your models here.

class log(models.Model):
    Datestamp = models.DateTimeField(auto_now=True)
    username =
models.CharField(max_length=150,null=False,blank=False,default="test")

class login_log(models.Model):
    Datestamp = models.DateTimeField(auto_now=True)
    username =
models.CharField(max_length=150,null=False,blank=False,default="test")
    login_success = models.BooleanField(null=False)

class password_change_log(log):
    password_change_type = models.BooleanField(null=False)
```

d. **Members**

```
from django.db import models

from django.contrib.auth.models import User

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    first_name = models.CharField(max_length=30, blank=True)
    last_name = models.CharField(max_length=30, blank=True)
    middle_initial = models.CharField(max_length=1, blank=True)
    email = models.EmailField(max_length=75, blank=True)
    is_driver = models.BooleanField('driver status', default=False)
    is_sponsor = models.BooleanField('sponsor status', default=False)

    def __str__(self):
        return self.user.username
```

```python
class SponsorUserProfile(UserProfile):
    sponsor_name = models.CharField(max_length=25)

    def __str__(self):
        return self.user.username

class SponsorList(models.Model):
    sponsor_name = models.CharField(max_length=25, unique=True)
    point_conversion = models.FloatField(default=0.01)
    # point_neg = models.BooleanField('negative points', default=False) # Can
negative point values be accepted?

    def __str__(self):
        return self.sponsor_name

class DriverProfile(UserProfile):
    points = models.IntegerField(default=0)
    sponsors = models.ManyToManyField(SponsorList,
related_name='sponsored_users')
    street_address = models.CharField(max_length= 85, blank=True)
    city = models.CharField(max_length=40, blank=True)
    state = models.CharField(max_length=2, blank=True)
    zipcode = models.CharField(max_length=5, blank=True)
    phone_number = models.CharField(max_length=15, blank=True)
    date_of_birth = models.DateField(blank=True, null=True)
    drivers_license = models.CharField(max_length=15, blank=True)

    # Vehicle information
    year = models.PositiveIntegerField(null=True)
    make = models.CharField(max_length=25, blank=True)
    model = models.CharField(max_length=25, blank=True)
    vin = models.CharField(max_length=17, blank=True)
    provider_name = models.CharField(max_length=100, blank=True)
    policy_number = models.CharField(max_length=50, blank=True)
    # Emergency contact
    emergency_contact_name = models.CharField(max_length=100, blank=True)
    emergency_contact_phone = models.CharField(max_length=20, blank=True)

    def __str__(self):
        return self.user.username

class PointReason(models.Model):
    point_amt = models.IntegerField(default=0)
    point_reason = models.TextField(max_length=250, blank=True)
    driver = models.ForeignKey(DriverProfile, on_delete=models.SET_NULL,
null=True)
    sponsor = models.ForeignKey(SponsorUserProfile, on_delete=models.SET_NULL,
null=True)
    is_add = models.BooleanField('point change type', default=True)

class PasswordResetToken(models.Model):
    user_profile = models.ForeignKey(UserProfile, on_delete=models.CASCADE)
    token = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
```