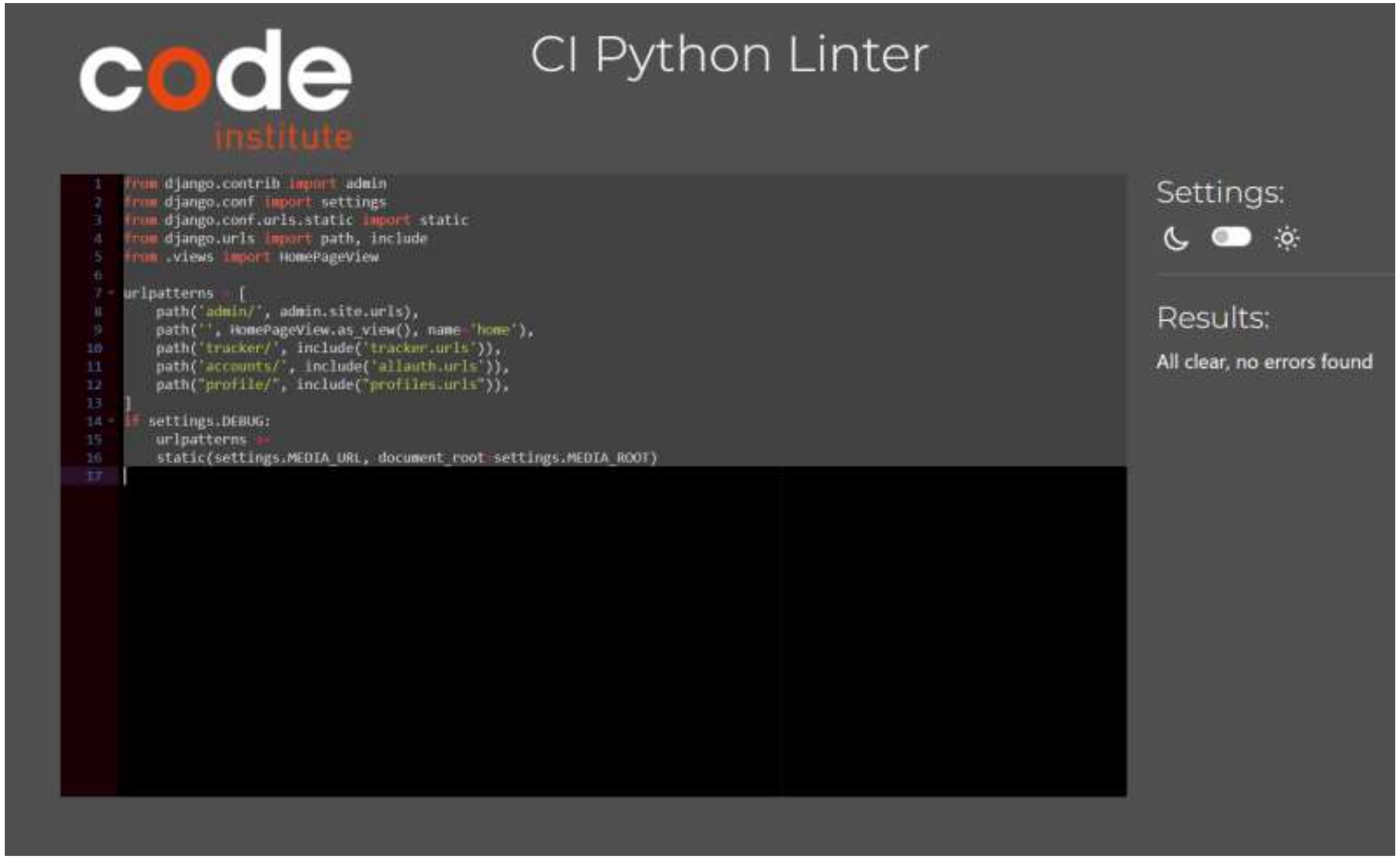
Inner Balance project

Urls.py


CI Python Linter

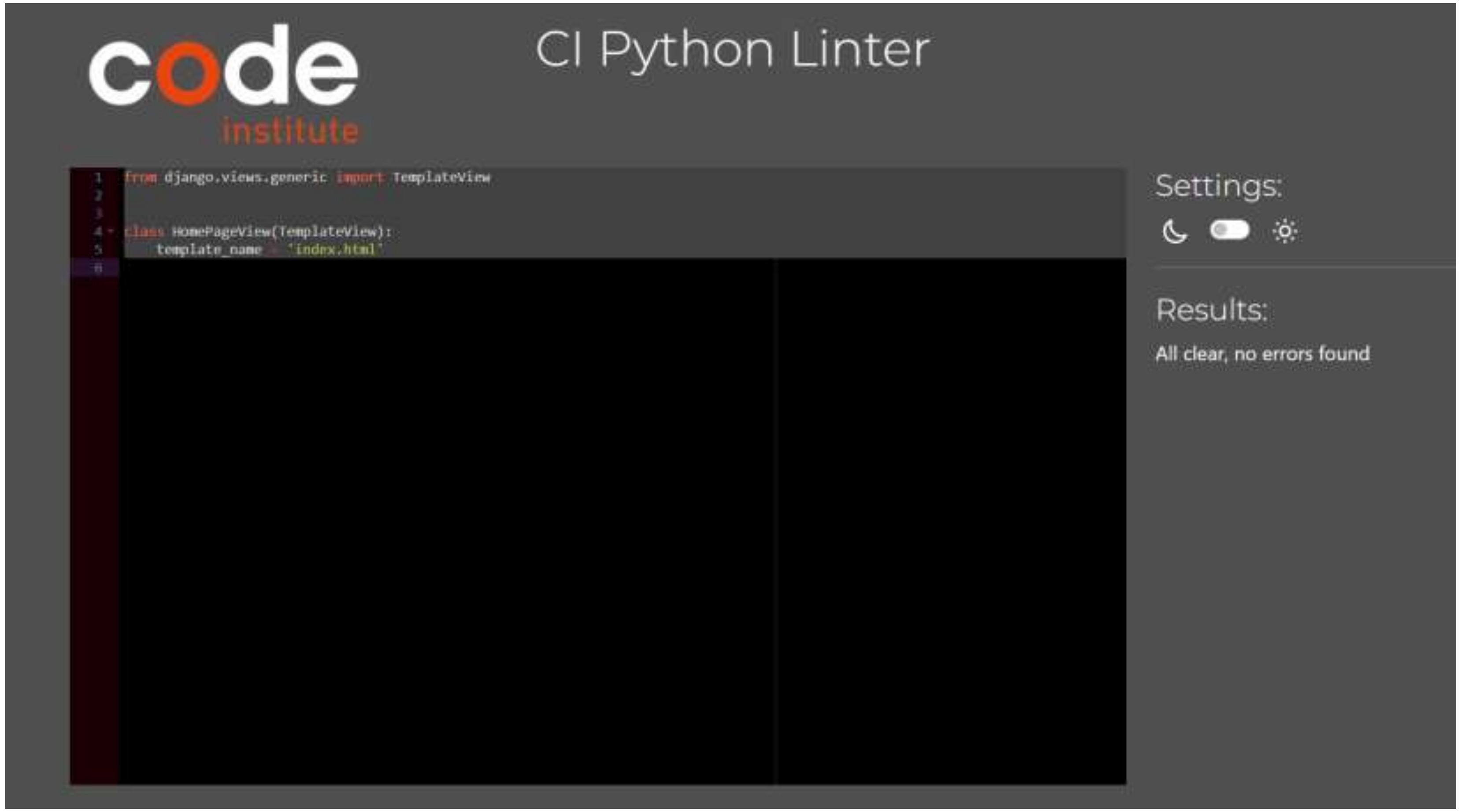
```python
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path, include
from .views import HomePageView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', HomePageView.as_view(), name='home'),
    path('tracker/', include('tracker.urls')),
    path('accounts/', include('allauth.urls')),
    path("profile/", include("profiles.urls")),
]
if settings.DEBUG:
    urlpatterns +=
    static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Settings:

Results:

All clear, no errors found

Views.py

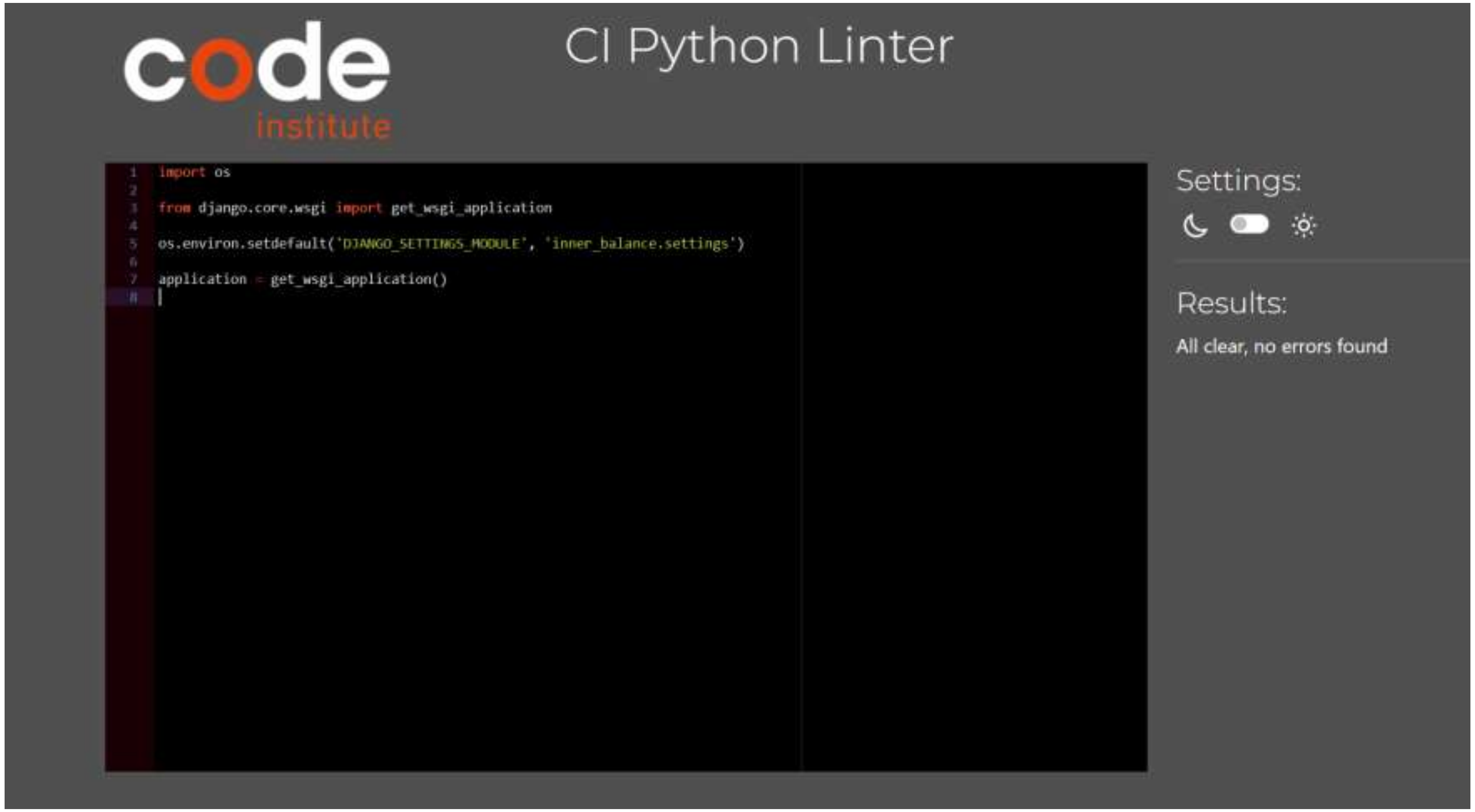


```
from django.views.generic import TemplateView


class HomePageView(TemplateView):
    template_name = 'index.html'
```

Results:

All clear, no errors found

Wsgi.py



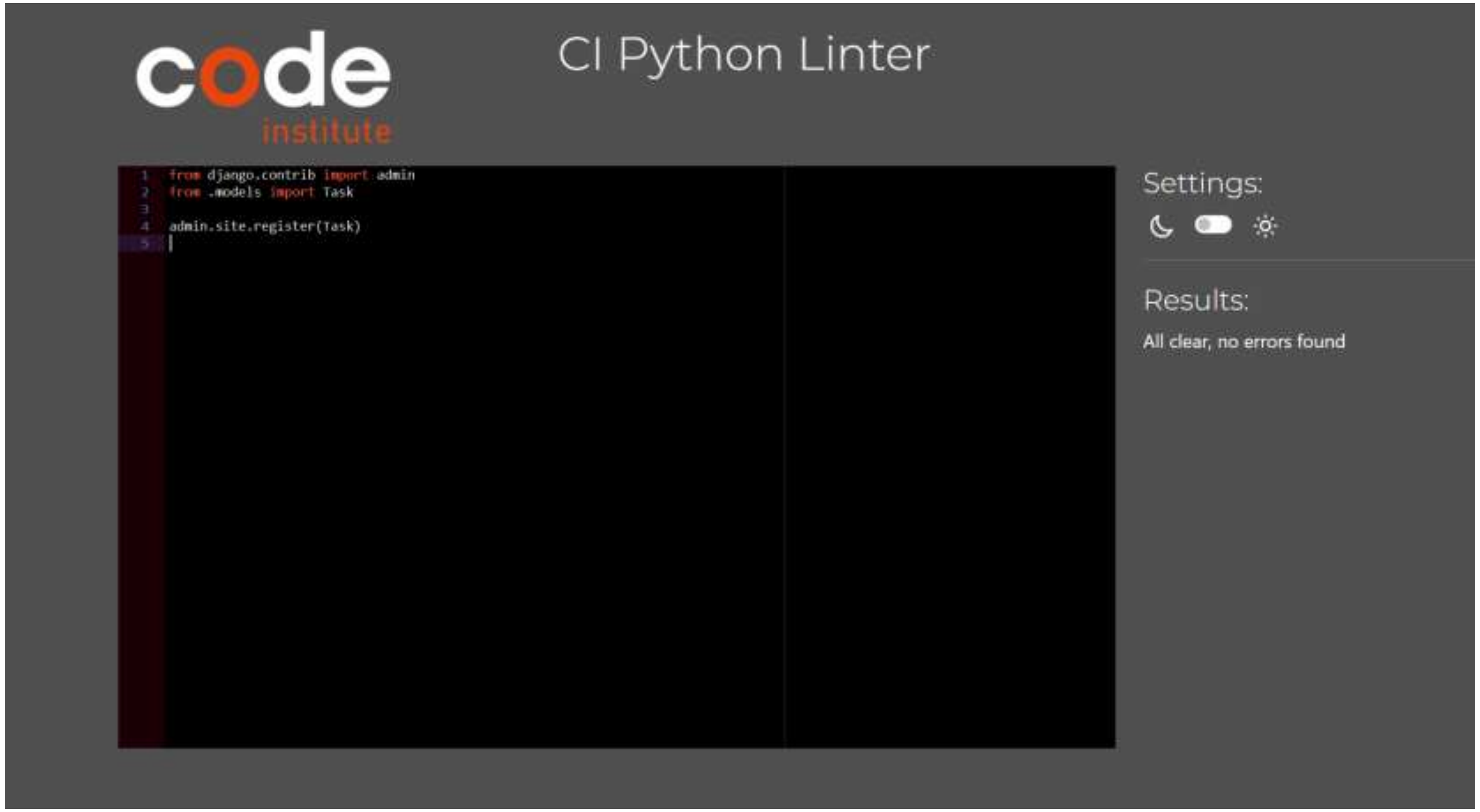
```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'inner_balance.settings')

application = get_wsgi_application()
```

Results:

All clear, no errors found

Tracker App

Admin.py

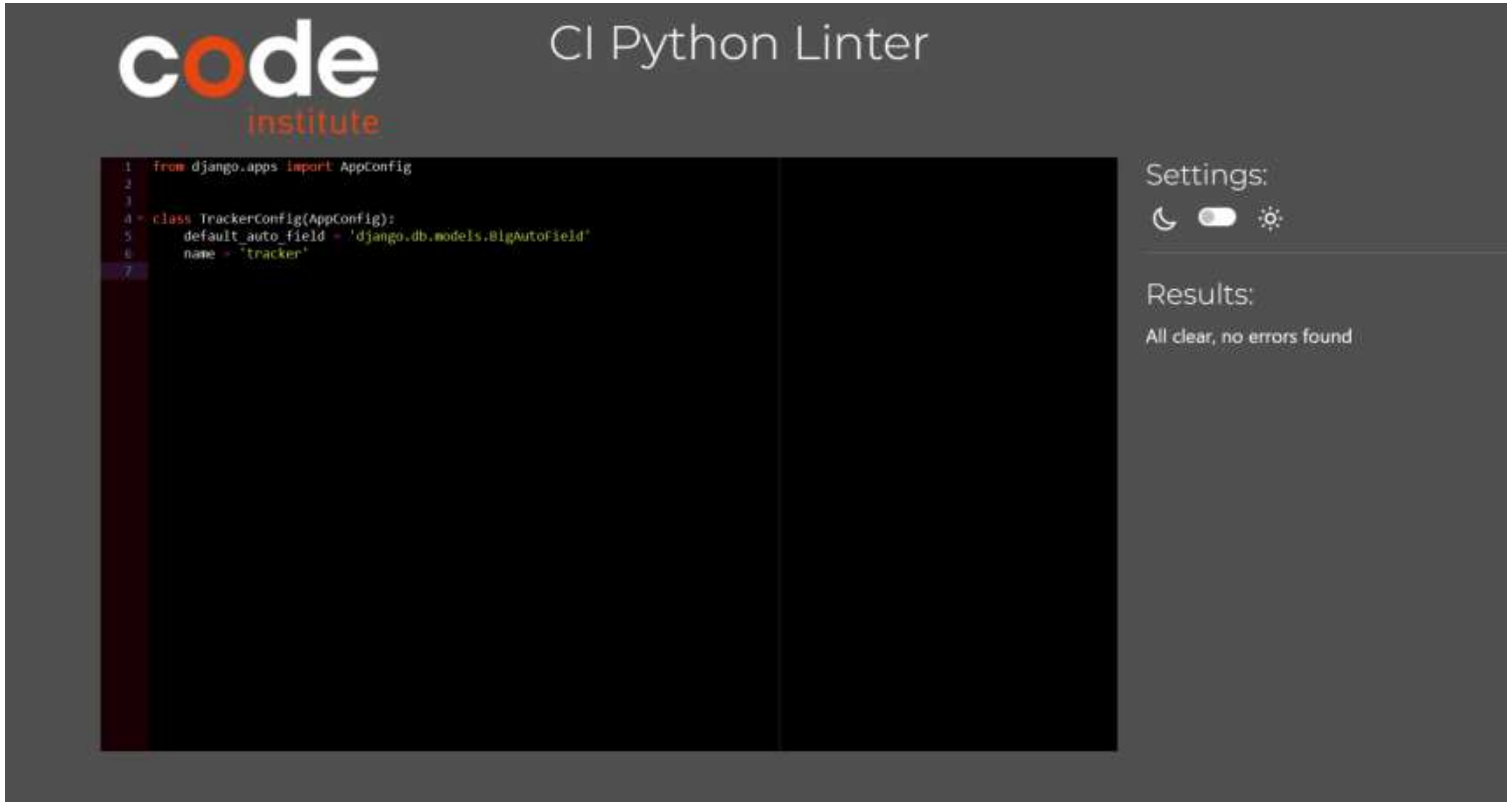


```python
from django.contrib import admin
from .models import Task

admin.site.register(Task)
```

Results:

All clear, no errors found

Apps.py

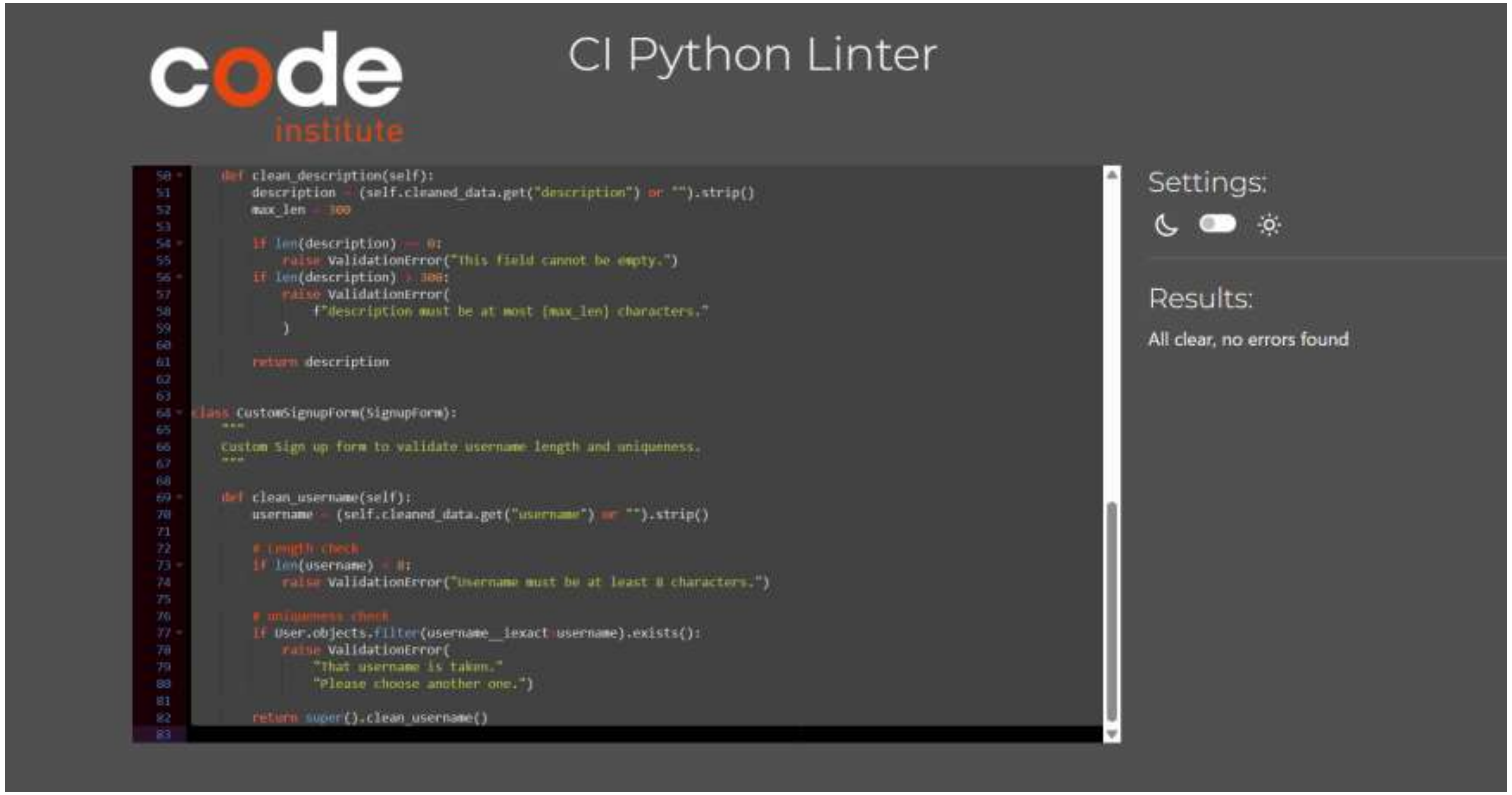


```
from django.apps import AppConfig


class TrackerConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'tracker'
```

Results:

All clear, no errors found

Forms.py



CI Python Linter

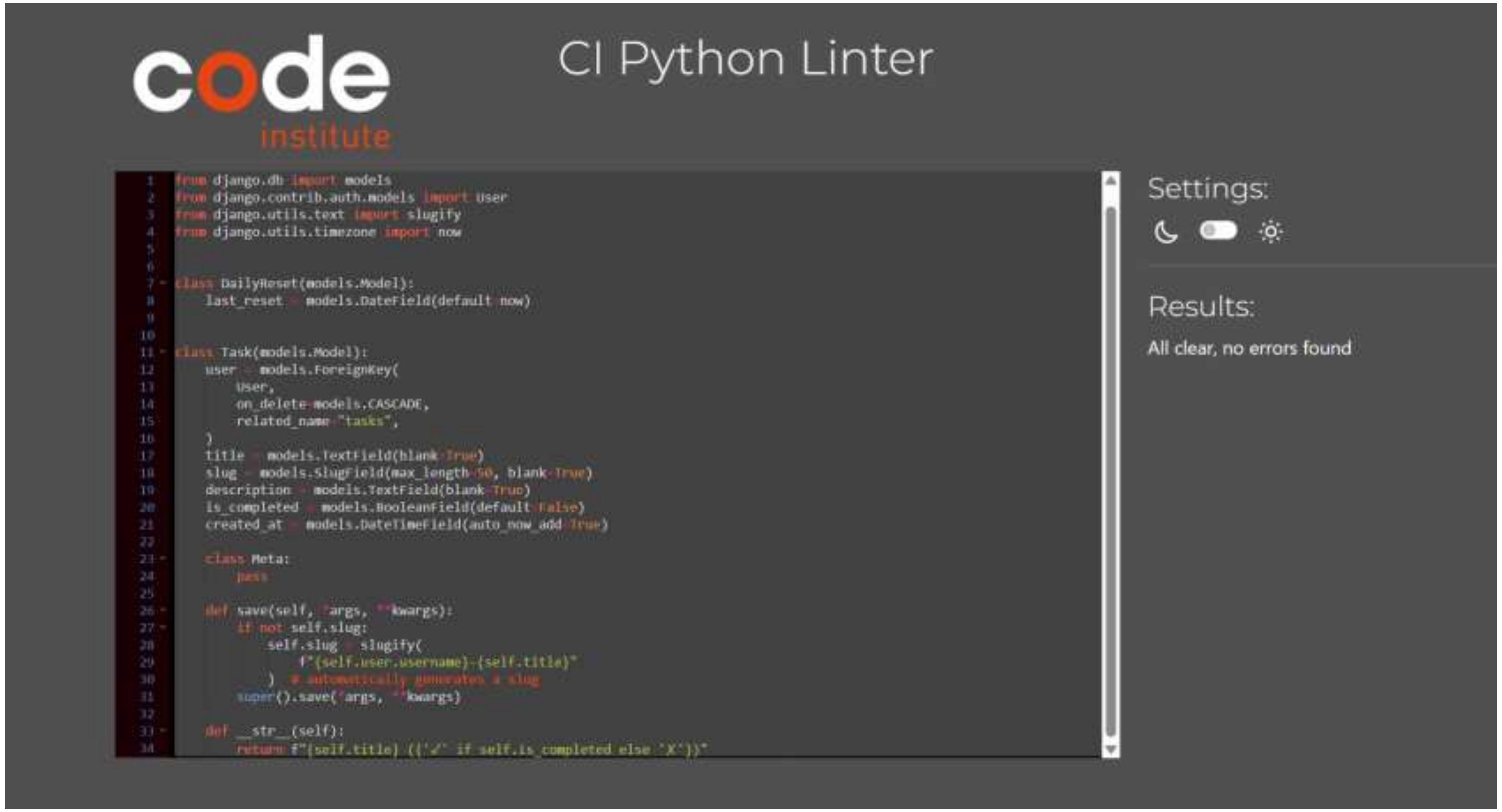
```python
    def clean_description(self):
        description = (self.cleaned_data.get("description") or "").strip()
        max_len = 300

        if len(description) == 0:
            raise ValidationError("This field cannot be empty.")
        if len(description) > 300:
            raise ValidationError(
                f"description must be at most {max_len} characters."
            )

        return description


class CustomSignupForm(SignupForm):
    """
    Custom Sign up form to validate username length and uniqueness.
    """

    def clean_username(self):
        username = (self.cleaned_data.get("username") or "").strip()

        # Length check
        if len(username) < 8:
            raise ValidationError("Username must be at least 8 characters.")

        # uniqueness check
        if User.objects.filter(username__iexact=username).exists():
            raise ValidationError(
                "That username is taken."
                "Please choose another one.")

        return super().clean_username()
```

Settings:

Results:

All clear, no errors found

Models.py



CI Python Linter

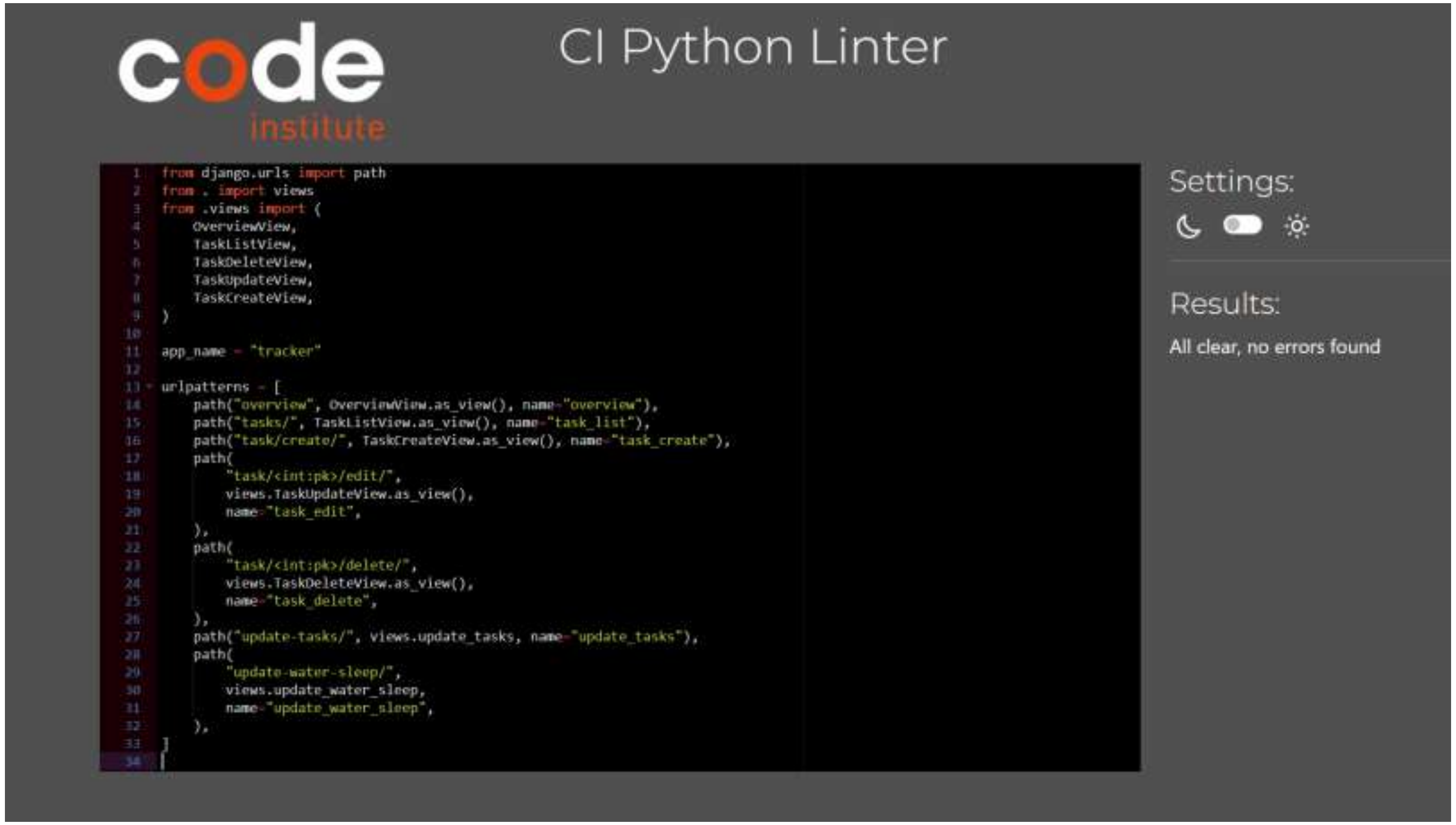
```python
from django.db import models
from django.contrib.auth.models import User
from django.utils.text import slugify
from django.utils.timezone import now


class DailyReset(models.Model):
    last_reset = models.DateField(default=now)


class Task(models.Model):
    user = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name="tasks",
    )
    title = models.TextField(blank=True)
    slug = models.SlugField(max_length=50, blank=True)
    description = models.TextField(blank=True)
    is_completed = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        pass

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(
                f"{self.user.username}-{self.title}"
            )  # automatically generates a slug
        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.title} ({'✓' if self.is_completed else 'X'})"
```

Settings:

Results:

All clear, no errors found

Urls.py



CI Python Linter

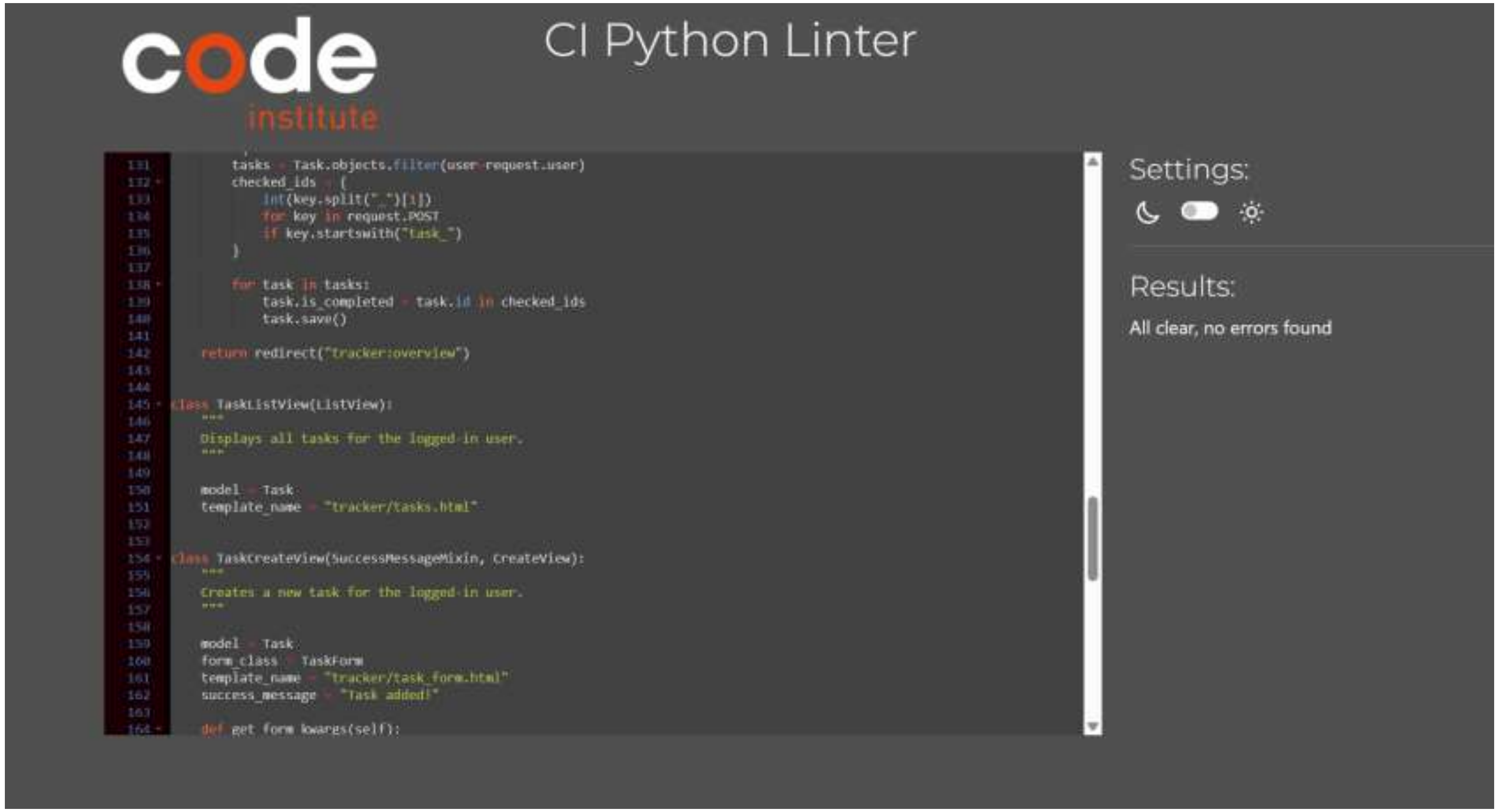
```python
from django.urls import path
from . import views
from .views import (
    OverviewView,
    TaskListView,
    TaskDeleteView,
    TaskUpdateView,
    TaskCreateView,
)

app_name = "tracker"

urlpatterns = [
    path("overview", OverviewView.as_view(), name="overview"),
    path("tasks/", TaskListView.as_view(), name="task_list"),
    path("task/create/", TaskCreateView.as_view(), name="task_create"),
    path(
        "task/<int:pk>/edit/",
        views.TaskUpdateView.as_view(),
        name="task_edit",
    ),
    path(
        "task/<int:pk>/delete/",
        views.TaskDeleteView.as_view(),
        name="task_delete",
    ),
    path("update-tasks/", views.update_tasks, name="update_tasks"),
    path(
        "update-water-sleep/",
        views.update_water_sleep,
        name="update_water_sleep",
    ),
]
```

Settings:

Results:

All clear, no errors found

Views.py



CI Python Linter

```python
        tasks = Task.objects.filter(user=request.user)
        checked_ids = {
            int(key.split("_")[1])
            for key in request.POST
            if key.startswith("task_")
        }

        for task in tasks:
            task.is_completed = task.id in checked_ids
            task.save()

    return redirect("tracker:overview")


class TaskListView(ListView):
    """
    Displays all tasks for the logged-in user.
    """

    model = Task
    template_name = "tracker/tasks.html"


class TaskCreateView(SuccessMessageMixin, CreateView):
    """
    Creates a new task for the logged-in user.
    """

    model = Task
    form_class = TaskForm
    template_name = "tracker/task_form.html"
    success_message = "Task added!"

    def get_form_kwargs(self):
```

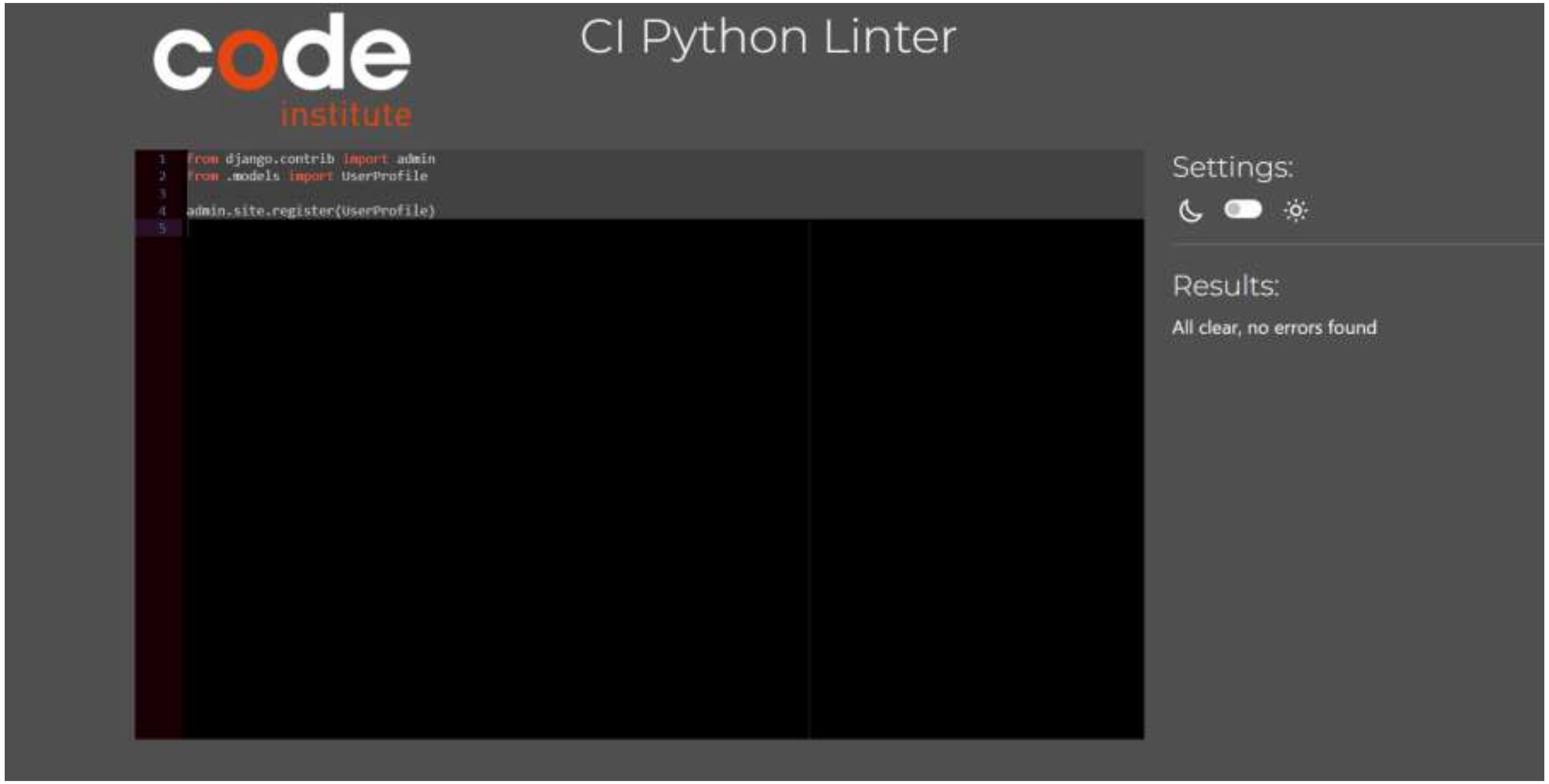
Settings:

Results:

All clear, no errors found

Profiles App

Admin.py

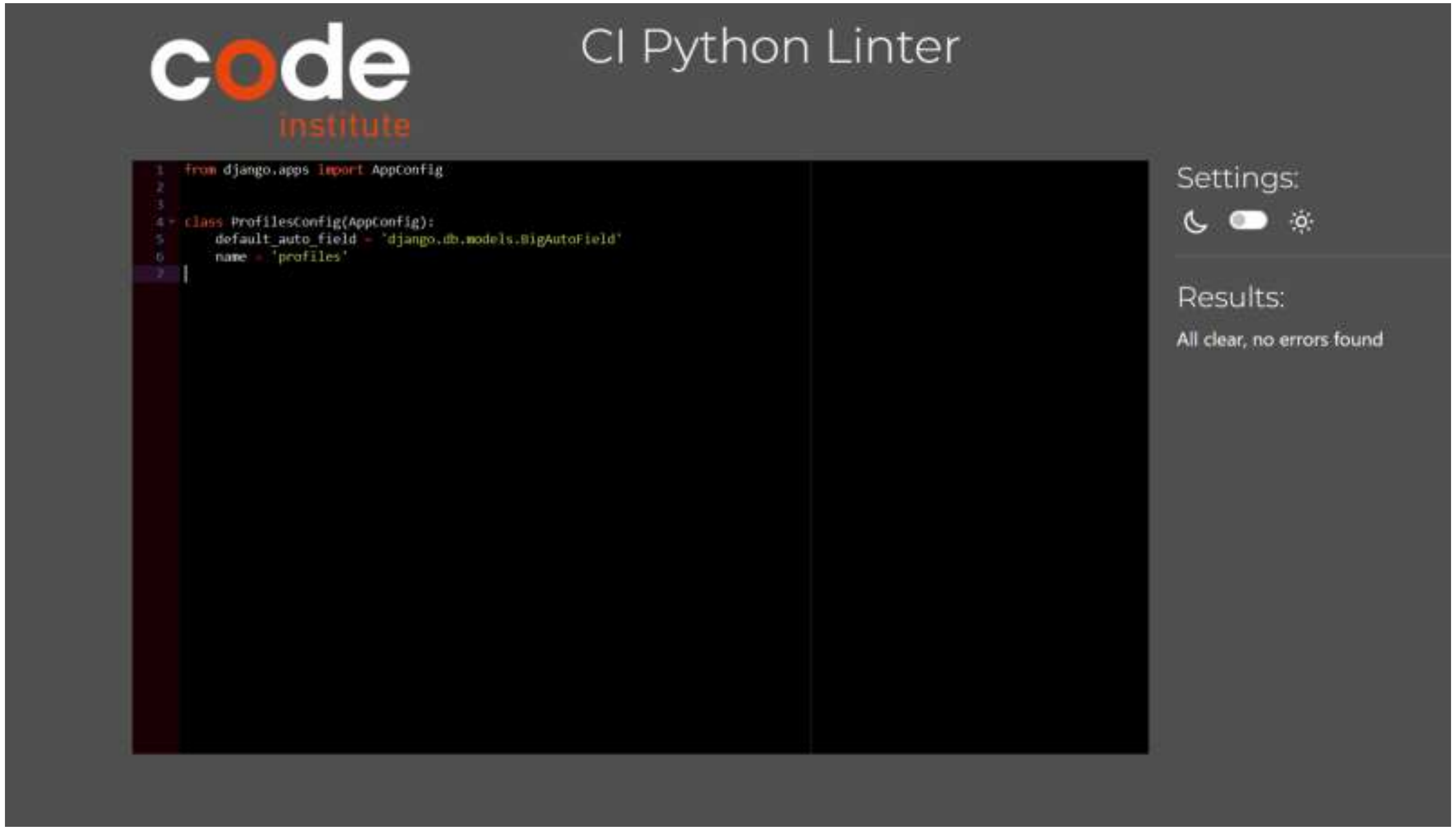


```
from django.contrib import admin
from .models import UserProfile

admin.site.register(UserProfile)
```

Results:

All clear, no errors found

Apps.py

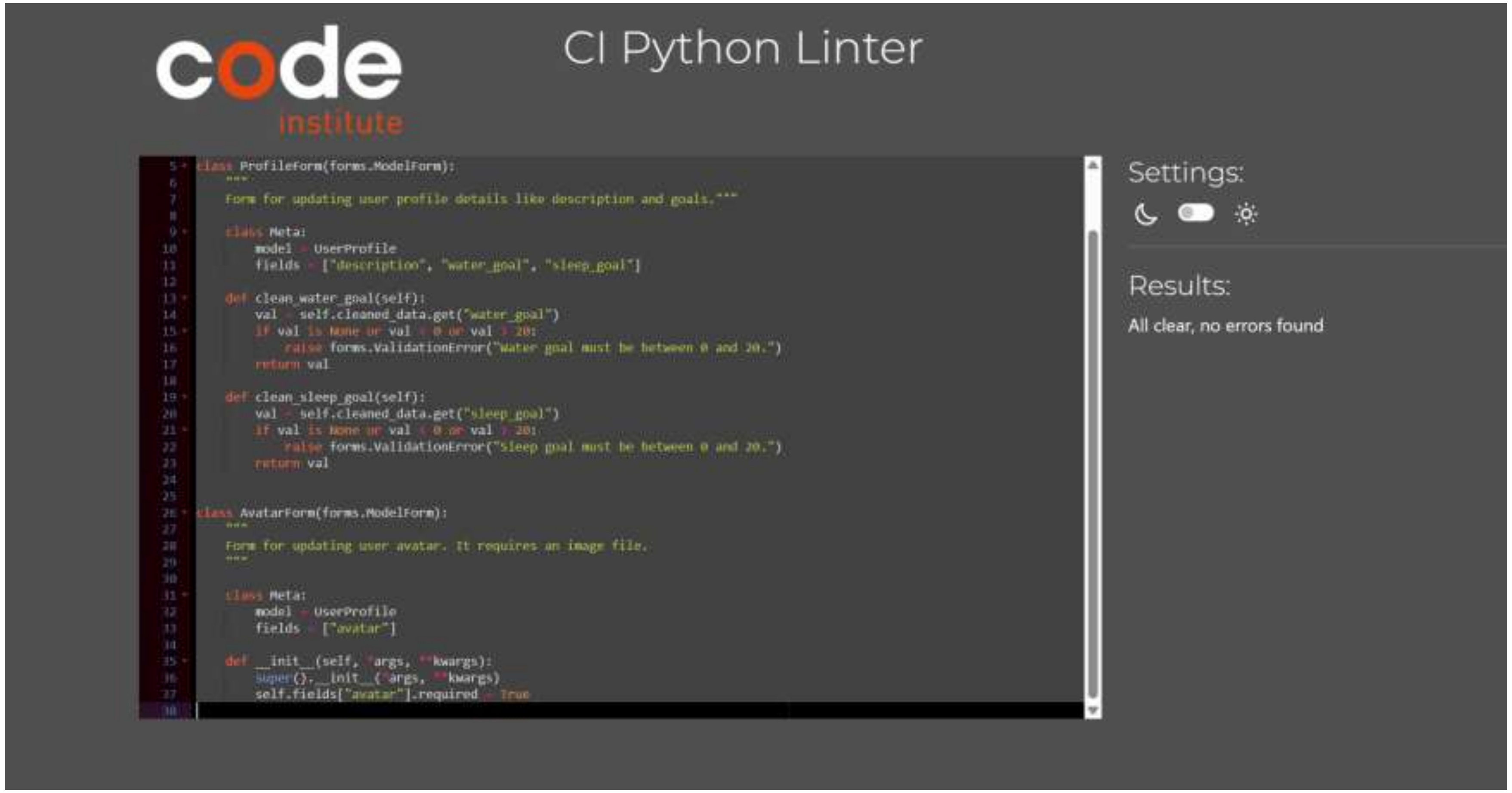


```
from django.apps import AppConfig


class ProfilesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'profiles'
```

Results:

All clear, no errors found

Forms.py



CI Python Linter

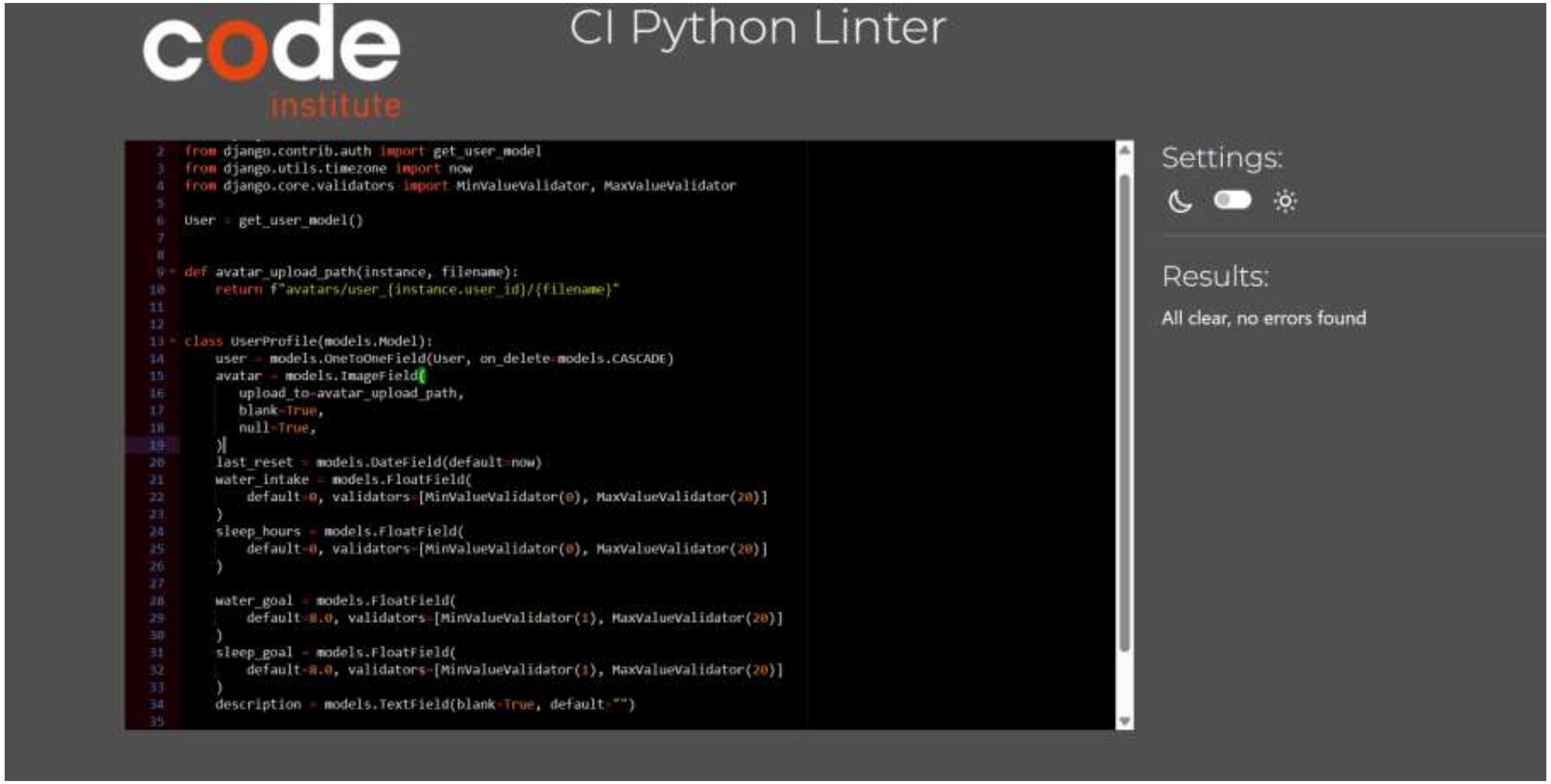
```python
class ProfileForm(forms.ModelForm):
    """
    Form for updating user profile details like description and goals."""

    class Meta:
        model = UserProfile
        fields = ["description", "water_goal", "sleep_goal"]

    def clean_water_goal(self):
        val = self.cleaned_data.get("water_goal")
        if val is None or val < 0 or val > 20:
            raise forms.ValidationError("Water goal must be between 0 and 20.")
        return val

    def clean_sleep_goal(self):
        val = self.cleaned_data.get("sleep_goal")
        if val is None or val < 0 or val > 20:
            raise forms.ValidationError("Sleep goal must be between 0 and 20.")
        return val


class AvatarForm(forms.ModelForm):
    """
    Form for updating user avatar. It requires an image file.
    """

    class Meta:
        model = UserProfile
        fields = ["avatar"]

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields["avatar"].required = True
```

Settings:

Results:

All clear, no errors found

Models.py



CI Python Linter

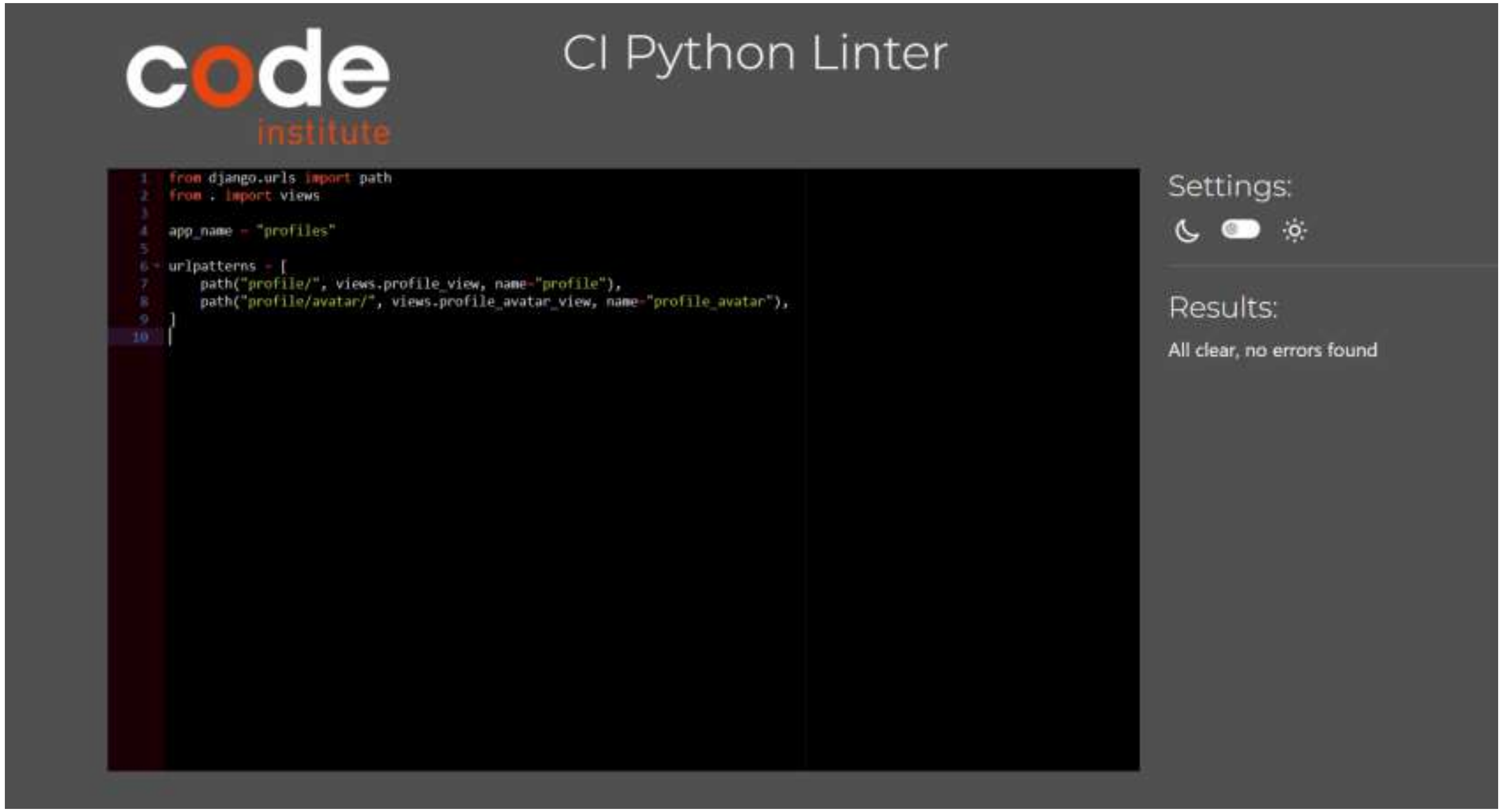
```python
from django.contrib.auth import get_user_model
from django.utils.timezone import now
from django.core.validators import MinValueValidator, MaxValueValidator

User = get_user_model()


def avatar_upload_path(instance, filename):
    return f"avatars/user_{instance.user_id}/{filename}"


class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    avatar = models.ImageField(
        upload_to=avatar_upload_path,
        blank=True,
        null=True,
    )
    last_reset = models.DateField(default=now)
    water_intake = models.FloatField(
        default=0, validators=[MinValueValidator(0), MaxValueValidator(20)]
    )
    sleep_hours = models.FloatField(
        default=0, validators=[MinValueValidator(0), MaxValueValidator(20)]
    )

    water_goal = models.FloatField(
        default=8.0, validators=[MinValueValidator(1), MaxValueValidator(20)]
    )
    sleep_goal = models.FloatField(
        default=8.0, validators=[MinValueValidator(1), MaxValueValidator(20)]
    )
    description = models.TextField(blank=True, default="")
```

Settings:

Results:

All clear, no errors found

Urls.py

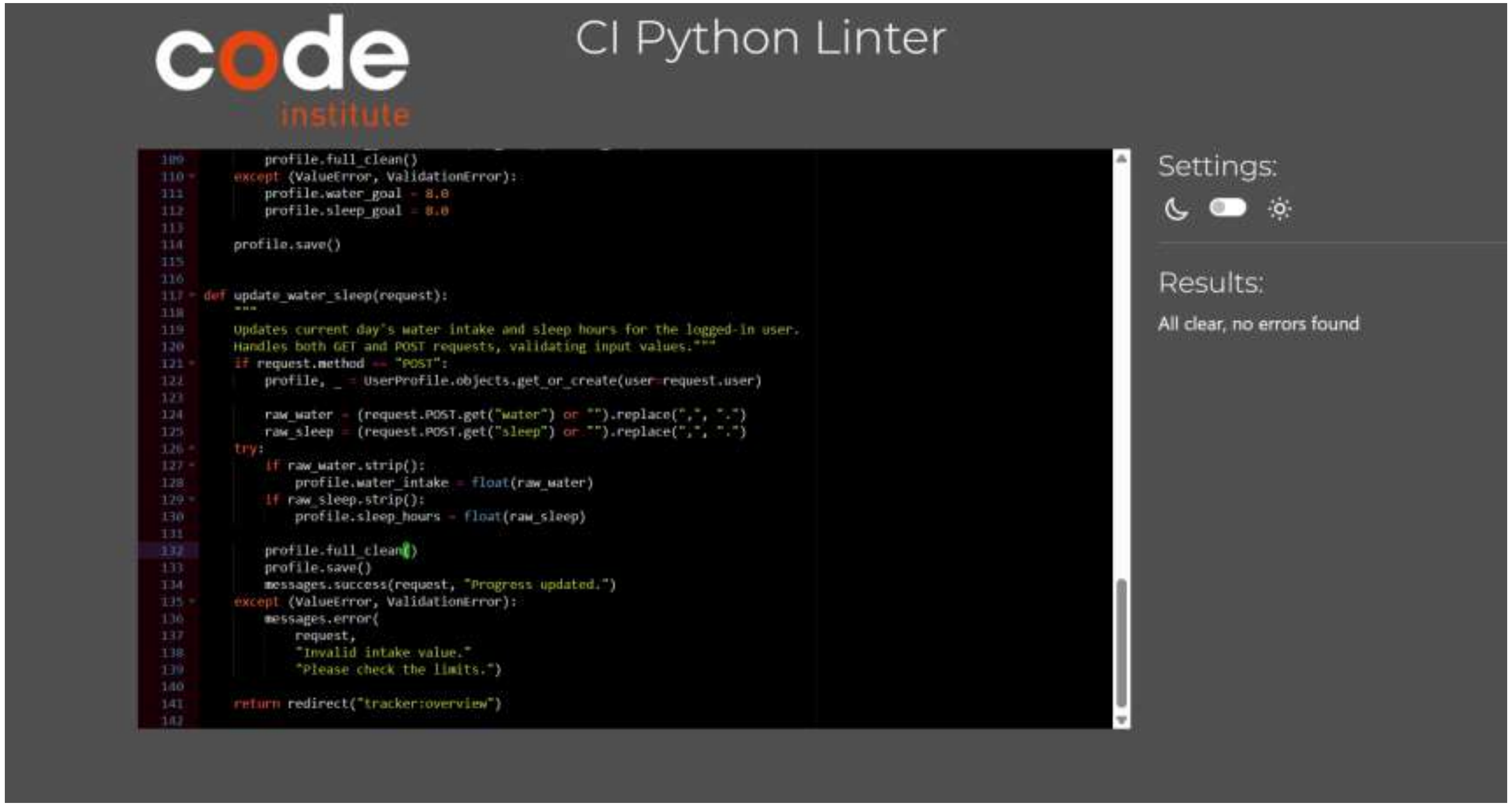


```python
from django.urls import path
from . import views

app_name = "profiles"

urlpatterns = [
    path("profile/", views.profile_view, name="profile"),
    path("profile/avatar/", views.profile_avatar_view, name="profile_avatar"),
]
```

Results:

All clear, no errors found

Views.py



CI Python Linter

```
        profile.full_clean()
    except (ValueError, ValidationError):
        profile.water_goal = 8.0
        profile.sleep_goal = 8.0

    profile.save()


def update_water_sleep(request):
    """
    Updates current day's water intake and sleep hours for the logged-in user.
    Handles both GET and POST requests, validating input values."""
    if request.method == "POST":
        profile, _ = UserProfile.objects.get_or_create(user=request.user)

        raw_water = (request.POST.get("water") or "").replace(",", ".")
        raw_sleep = (request.POST.get("sleep") or "").replace(",", ".")
    try:
        if raw_water.strip():
            profile.water_intake = float(raw_water)
        if raw_sleep.strip():
            profile.sleep_hours = float(raw_sleep)

        profile.full_clean()
        profile.save()
        messages.success(request, "Progress updated.")
    except (ValueError, ValidationError):
        messages.error(
            request,
            "Invalid intake value."
            "Please check the limits.")

    return redirect("tracker:overview")
```

Settings:

Results:

All clear, no errors found