Exercise 3

Group number:
22

- Trym Grande
- Hans Kristian Granli

Introduction

In this exercise we have continued our forked version of the secfit project. Our GitLab repository can be found here.

For task 1, we selected some of the guidelines from the lecture, while also included some from the RSPEC-1720 list. In addition, we included the set of rules included in the PEP 8 style guide, used in the automatic refactoring. The results found in task 2 are listed with reference to the checklist in our guidelines, as well as corresponding code snipptets.

Task 1 - Code Review Guidelines

Section class

Feature	Checklist item
Inheritance	• Q1: Are all inheritance necessary?

Section method

Feature	Checklist item
Data referencing	• Q2: Are all parameters used? • Q3: Are all variable assignments used?

Feature	Checklist item
Functionality	• Q4: Is the same functionality implemented multiple times (duplicate code)? • Q5: Functions and methods should not be empty, unless they have a reasonable comment explaining why. • Q6: Are jump statements redundant? • Q7: Does the method exceed 50 lines?
Style and readability	• Q8: Does the code exceed 3 levels of nesting? • Q9: Does the method have a docstring if it is a non-trivial method to explain what it does?

Section General Code

Feature	Checklist item
Style and readability	• Q10: Does the code have matching/correct indentation? (4 spaces/tab) • Q11: Are all variables initialized with meaningful name? • Q12: Are paramaters with default values modified? rule reference Q13: Are all imported methods used? (wildcard imports) • Q14: Do two branches in conditional (if-statement) have the same implementation? rule reference • Q15: Are variables, functions, and class names using appropriate conventions? • Q16: Is there a newline at the end of the file if it is a python file? Q17: Are comments only consisting of explanatory text? (not old/unused code) • Q18: Is the python code formatted after the PEP8 standard?

Task 2 - Refactoring Results

After refactoring all our code-tests including selenium are passing, which means that the correctness has not been altered by the refactoring.

Table

Bad code smell category	How is the code smell identified (code review or automatic analysis tools?)	Relate test scripts	file name	occurances
Q17	Manually	none	users/views.py	1
Q13	Manually	none	users/views.py	3
Q10	Manually	none	users/urls.py	1
Q13	Manually	none	users/urls.py	2
Q9	Manually	none	users/views.py	5
Q9	Manually	none	gallery.js	3
Q7	Manually	none	gallery.js	1
Q3	Manually	none	gallery.js	1
Q9	Manually	none	exercise.js	7
Q13	Manually	none	workouts/views.py	3
Q17	Manually	none	workouts/views.py	3
Q17	Manually	none	comments/views.py	1
Q13	Manually	none	comments/views.py	1
Q18	Automatically	code_formatting.sh	models.py, workouts/views.py, workouts/serializers.py, admin.py, users/views.py, urls.py	26

Automatic Code Formatting

For automatic refactoring, we decided to use an automated tool called "autopep8", found here. This automatically checks and refactors Python code to conform to the PEP 8 style guide.

```
#!/bin/sh
# code_formatting.sh

pip install autopep8

autopep8 --in-place backend/secfit/workouts/models.py
autopep8 --in-place backend/secfit/workouts/views.py
autopep8 --in-place backend/secfit/workouts/serializers.py
autopep8 --in-place backend/secfit/users/admin.py
autopep8 --in-place backend/secfit/users/views.py
autopep8 --in-place backend/secfit/users/urls.py
autopep8 --in-place backend/secfit/comments/views.py
```

Code Snippets

gallery.js

Q9

before

after

```
function handleGoBackToWorkoutClick()
    /**
     * Updates and adds a location for the back-button to return to workout page
     */

async function handleDeleteImgClick (id, http_keyword, fail_alert_text, host_variable, a
    /**
     * Sends a delete request to the server and reloads the page
     */

async function validateImgFileType(id, host_variable, acceptedFileTypes)
    /**
     * Validates that a file is of a supported image type
     */

async function retrieveWorkoutImages(id)
    /**
     * Fetches images related to a workout from the server
     */

async function processWorkoutImages(workoutJson)
    /**
     * Processes a response given to a workout-images request
     */
```

Q7

before

```
async function retrieveWorkoutImages(id) {
    let workoutData = null;
    let response = await sendRequest("GET", `${HOST}/api/workouts/${id}/`);
    if (!response.ok) {
        let data = await response.json();
        let alert = createAlert("Could not retrieve workout data!", data);
        document.body.prepend(alert);
    } else {
        workoutData = await response.json();

        document.getElementById("workout-title").innerHTML = "Workout name: " + workoutD
        document.getElementById("workout-owner").innerHTML = "Owner: " + workoutData["ow

        let hasNoImages = workoutData.files.length == 0;
        let noImageText = document.querySelector("#no-images-text");

        if(hasNoImages){
            noImageText.classList.remove("hide");
            return;
        }

        noImageText.classList.add("hide");


        let filesDiv = document.getElementById("img-collection");
        let filesDeleteDiv = document.getElementById("img-collection-delete");

        const currentImageFileElement = document.querySelector("#current");
        let isFirstImg = true;

        let fileCounter = 0;

        for (let file of workoutData.files) {
            let a = document.createElement("a");
            a.href = file.file;
            let pathArray = file.file.split("/");
            a.text = pathArray[pathArray.length - 1];
            a.className = "me-2";



            let isImage = ["jpg", "png", "gif", "jpeg", "JPG", "PNG", "GIF", "JPEG"].inc

            if(isImage){
                let deleteImgButton = document.createElement("input");
                deleteImgButton.type = "button";
                deleteImgButton.className = "btn btn-close";
                deleteImgButton.id = file.url.split("/")[file.url.split("/").length - 2]
                deleteImgButton.addEventListener('click', () => handleDeleteImgClick(del
                filesDeleteDiv.appendChild(deleteImgButton);
```

```
                let img = document.createElement("img");
                img.src = file.file;

                filesDiv.appendChild(img);
                deleteImgButton.style.left = `${(fileCounter % 4) * 191}px`;
                deleteImgButton.style.top = `${Math.floor(fileCounter / 4) * 105}px`;

                if(isFirstImg){
                    currentImageFileElement.src = file.file;
                    isFirstImg = false;
                }
                fileCounter++;
            }
        }

        const otherImageFileElements = document.querySelectorAll(".imgs img");
        const selectedOpacity = 0.6;
        otherImageFileElements[0].style.opacity = selectedOpacity;

        otherImageFileElements.forEach((imageFileElement) => imageFileElement.addEventLi
            //Changes the main image
            currentImageFileElement.src = event.target.src;

            //Adds the fade animation
            currentImageFileElement.classList.add('fade-in')
            setTimeout(() => currentImageFileElement.classList.remove('fade-in'), 500);

            //Sets the opacity of the selected image to 0.4
            otherImageFileElements.forEach((imageFileElement) => imageFileElement.style.
            event.target.style.opacity = selectedOpacity;
        }))

    }
    return workoutData;
}
```

after

```
async function createWorkoutImage(file, fileCounter, filesDiv, deleteDiv){

    /**
     * Generates an html image and delete button and appends it to given div
     */

    let deleteImgButton = document.createElement("input");
    deleteImgButton.type = "button";
    deleteImgButton.className = "btn btn-close";
    deleteImgButton.id = file.url.split("/")[file.url.split("/").length - 2];
    deleteImgButton.addEventListener('click', () => handleDeleteImgClick(deleteImgButtor
    deleteDiv.appendChild(deleteImgButton);

    let img = document.createElement("img");
    img.src = file.file;

    filesDiv.appendChild(img);
    deleteImgButton.style.left = `${(fileCounter % 4) * 191}px`;
    deleteImgButton.style.top = `${Math.floor(fileCounter / 4) * 105}px`;

    return img
}

async function createWorkoutImages(workoutData){
    /**
     * Renders images in a json-response
     */

    let filesDiv = document.getElementById("img-collection");
    let filesDeleteDiv = document.getElementById("img-collection-delete");

    const currentImageFileElement = document.querySelector("#current");
    let isFirstImg = true;

    let fileCounter = 0;

    for (let file of workoutData.files) {
        let a = document.createElement("a");
        a.href = file.file;
        let pathArray = file.file.split("/");
        a.text = pathArray[pathArray.length - 1];
        a.className = "me-2";
        let isImage = ["jpg", "png", "gif", "jpeg", "JPG", "PNG", "GIF", "JPEG"].include

        if(isImage){

            await createWorkoutImage(file, fileCounter, filesDiv, filesDeleteDiv)

            if(isFirstImg){
                currentImageFileElement.src = file.file;
```

```
                isFirstImg = false;
            }
            fileCounter++;
        }
    }
}

async function processWorkoutImages(workoutJson){

    /**
     * Processes a response given to a workout-images request
     */

    let workoutData = await workoutJson.json();

    document.getElementById("workout-title").innerHTML = "Workout name: " + workoutData[
    document.getElementById("workout-owner").innerHTML = "Owner: " + workoutData["owner_

    let hasNoImages = workoutData.files.length == 0;
    let noImageText = document.querySelector("#no-images-text");

    if(hasNoImages){
        noImageText.classList.remove("hide");
        return;
    }

    noImageText.classList.add("hide");

    await createWorkoutImages(workoutData)

    const otherImageFileElements = document.querySelectorAll(".imgs img");
    const selectedOpacity = 0.6;
    otherImageFileElements[0].style.opacity = selectedOpacity;

    otherImageFileElements.forEach((imageFileElement) => imageFileElement.addEventLister
        //Changes the main image
        currentImageFileElement.src = event.target.src;

        //Adds the fade animation
        currentImageFileElement.classList.add('fade-in')
        setTimeout(() => currentImageFileElement.classList.remove('fade-in'), 500);

        //Sets the opacity of the selected image to 0.4
        otherImageFileElements.forEach((imageFileElement) => imageFileElement.style.opac
        event.target.style.opacity = selectedOpacity;
    }))

    return workoutData
}

async function retrieveWorkoutImages(id) {
```

```
    /**
     * Fetches images related to a workout from the server
     */

    let workoutData = null;
    let response = await sendRequest("GET", `${HOST}/api/workouts/${id}/`);
    if (!response.ok) {
        let data = await response.json();
        let alert = createAlert("Could not retrieve workout data!", data);
        document.body.prepend(alert);
    } else {
        workoutData = await processWorkoutImages(response)
    }
    return workoutData;
}
```

Q3

before

```
let workoutData = await retrieveWorkoutImages(id);
```

after

```
await retrieveWorkoutImages(id);
```

exercise.js

Q9

before

after (code removed)

```
async function updateExercise(id)
    /**
     * Read, process, and updates an exercise on the server
     */

function deleteExercise(id)
    /**
     * Makes a server request to delete an exercise on the server with a given id
     */

function handleEditExerciseButtonClick()
    /**
     * Updates the exercise fields from disabled to interactable
     * so that the user gains access to the edit-exercise functionality
     */

function createExercise()
    /**
     * Read, process and post exercise data to the server
     */

function handleCancelButtonDuringEdit()
    /**
     * Event handler for when a user discards changes -> Set input fields to disabled
     */

setMuscleGroupType = (newType) =>
    /**
     * Changes the muscle group of the object
     */
```

users/admin.py

No code smells found.

Q18

before

```
# list_display = UserAdmin.list_display + ('coach',)
```

after

users/views.py

Q14

before

```
    permission_classes = [permissions.IsAuthenticated & (IsCurrentUser | IsReadOnly)]
```

after

```
    permission_classes = [permissions.IsAuthenticated &
                          (IsCurrentUser | IsReadOnly)]
```

Q18

before

```
 #permission_classes = [permissions.IsAuthenticated & (IsCurrentUser | IsReadOnly)]
```

after

Q11

before

```
    def get_queryset(self):
        qs = get_user_model().objects.all()

        if self.request.user:
            # Return the currently logged in user
            status = self.request.query_params.get("user", None)
            if status and status == "current":
                qs = get_user_model().objects.filter(pk=self.request.user.pk)

        return qs
```

after

```
    def get_queryset(self):
        queryset = get_user_model().objects.all()

        if self.request.user:
            # Return the currently logged in user
            status = self.request.query_params.get("user", None)
            if status and status == "current":
                queryset = get_user_model().objects.filter(pk=self.request.user.pk)

        return queryset
```

before

```
    def get_queryset(self):
        qs = Offer.objects.none()
        result = Offer.objects.none()

        if self.request.user:
            qs = Offer.objects.filter(
                Q(owner=self.request.user) | Q(recipient=self.request.user)
            ).distinct()
            qp = self.request.query_params
            u = self.request.user

            # filtering by status (if provided)
            s = qp.get("status", None)
            if s is not None and self.request is not None:
                qs = qs.filter(status=s)
                if qp.get("status", None) is None:
                    qs = Offer.objects.filter(Q(owner=u)).distinct()

            # filtering by category (sent or received)
            c = qp.get("category", None)
            if c is not None and qp is not None:
                if c == "sent":
                    qs = qs.filter(owner=u)
                elif c == "received":
                    qs = qs.filter(recipient=u)
            return qs
        else:
            return result
```

after

```
    def get_queryset(self):
        queryset = Offer.objects.none()
        result = Offer.objects.none()

        if self.request.user:
            queryset = Offer.objects.filter(
                Q(owner=self.request.user) | Q(recipient=self.request.user)
            ).distinct()
            query_params = self.request.query_params
            user = self.request.user

            # filtering by status (if provided)
            status = query_params.get("status", None)
            if status is not None and self.request is not None:
                queryset = queryset.filter(status=status)
                if query_params.get("status", None) is None:
                    queryset = Offer.objects.filter(Q(owner=user)).distinct()

            # filtering by category (sent or received)
            category = query_params.get("category", None)
            if category is not None and query_params is not None:
                if category == "sent":
                    queryset = queryset.filter(owner=user)
                elif category == "received":
                    queryset = queryset.filter(recipient=user)
            return queryset
        else:
            return result
```

before

```
    def get_queryset(self):
        qs = AthleteFile.objects.none()

        if self.request.user:
            qs = AthleteFile.objects.filter(
                Q(athlete=self.request.user) | Q(owner=self.request.user)
            ).distinct()

        return qs
```

after

```
    def get_queryset(self):
        queryset = AthleteFile.objects.none()

        if self.request.user:
            queryset = AthleteFile.objects.filter(
                Q(athlete=self.request.user) | Q(owner=self.request.user)
            ).distinct()

        return queryset
```

Q13

before

```
from django.shortcuts import get_object_or_404
```

after

before

```
from rest_framework.permissions import (
    AllowAny,
    IsAdminUser,
    IsAuthenticated,
    IsAuthenticatedOrReadOnly,
)
```

after

```
from rest_framework.permissions import (
    IsAuthenticatedOrReadOnly
)
```

before

```
import django
```

after

Q9

before

```
class UserList():
    def get_queryset(self):
```

after

```
class UserList():
    def get_queryset(self):
        """
        Return the currently logged in user.
        """
```

before

```
class CoachAthletesList():
    def get_queryset(self):
```

after

```
class CoachAthletesList():
    def get_queryset(self):
        """
        Returns the athletes of the currently logged in user.
        """
```

before

```
class UserDetail():
    def get_object(self):
```

after

```
class UserDetail():
    def get_object(self):
        """
        Returns first object in self.kwargs that exist in lookup_field_options.
        """
```

before

```
class OfferList():
    def get_queryset(self):
```

after

```
class OfferList():
    def get_queryset(self):
        """
        Returns currently logged in user filtered by status (if provided) and category (
        """
```

before

```
class AthleteFileList():
    def get_queryset(self):
```

after

```
class AthleteFileList():
    def get_queryset(self):
        """
        Returns an athlete file.
        """
```

users/urls.py

Q14

before

```
    path("api/users/<str:username>/", views.UserDetail.as_view(), name="user-detail"),
    path("api/reset_password/", auth_views.PasswordResetView.as_view(
        template_name="password_reset.html"),
        name="reset_password"),
    path("api/reset_password_sent/", auth_views.PasswordResetDoneView.as_view(
        template_name="password_reset_sent.html"),
        name="password_reset_done"),
    path("api/reset/<uidb64>/<token>/", auth_views.PasswordResetConfirmView.as_view(
        template_name="password_reset_form.html"),
        name="password_reset_confirm"),
    path("api/reset_password_complete/", auth_views.PasswordResetCompleteView.as_view(
        template_name="password_reset_done.html"),
        name="password_reset_complete")
]
```

after

```
    path("api/users/<str:username>/",
         views.UserDetail.as_view(), name="user-detail"),
    path("api/reset_password/", auth_views.PasswordResetView.as_view(
        template_name="password_reset.html"),
        name="reset_password"),
    path("api/reset_password_sent/", auth_views.PasswordResetDoneView.as_view(
        template_name="password_reset_sent.html"),
        name="password_reset_done"),
    path("api/reset/<uidb64>/<token>/", auth_views.PasswordResetConfirmView.as_view(
        template_name="password_reset_form.html"),
        name="password_reset_confirm"),
    path("api/reset_password_complete/", auth_views.PasswordResetCompleteView.as_view(
        template_name="password_reset_done.html"),
        name="password_reset_complete")
]
```

Q10

before

```
path(
    "api/athlete-files/", views.AthleteFileList.as_view(), name="athlete-file-list"
),
```

after

```
path("api/athlete-files/", views.AthleteFileList.as_view(), name="athlete-file-list"),
```

Q13

before

```
from rest_framework.urlpatterns import format_suffix_patterns
```

after

before

```
from django.urls import path
```

after

```
from django.urls import path
```

workouts/models.py

Q14

before

```
    workout = models.ForeignKey(Workout, on_delete=models.CASCADE, related_name="files")
    workout = models.ForeignKey(Workout, on_delete=models.CASCADE, related_name="data_fi
```

after

```
    workout = models.ForeignKey(
        Workout, on_delete=models.CASCADE, related_name="files")
    workout = models.ForeignKey(
        Workout, on_delete=models.CASCADE, related_name="data_file")
```

- The method workout_data_directory_path had incorrect description

before

```
"""Return path for which workout files should be uploaded on the web server

Args:
    instance (WorkoutFile): WorkoutFile instance
    filename (str): Name of the file

Returns:
    str: Path where workout file is stored
"""
```

after

```
"""Return path for which workout data files should be uploaded on the web server

    Args:
        instance (WorkoutData): WorkoutData instance
        filename (str): Name of the file

    Returns:
        str: Path where workout file is stored
"""
```

workouts/views.py

Q14

before

```
import base64, pickle
```

after

```
import base64
import pickle
```

before

```
(Q(workout__visibility="CO") | Q(workout__visibility="PU"))
& Q(workout__owner__coach=self.request.user)
```

after

```
(Q(workout__visibility="CO") | Q(workout__visibility="PU"))
& Q(workout__owner__coach=self.request.user)
```

before

```
IsOwnerOfWorkout
| (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic))
```

after

```
IsOwnerOfWorkout
| (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic))
```

before

```
Q(workout__visibility="CO")
& Q(workout__owner__coach=self.request.user)
```

after

```
Q(workout__visibility="CO")
& Q(workout__owner__coach=self.request.user)
```

before

```
IsOwner
| IsOwnerOfWorkout
| (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic))
```

after

```
IsOwner
| IsOwnerOfWorkout
| (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic))
```

before

```
IsOwnerOfWorkout | (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic)))
            ]
```

after

```
    IsOwnerOfWorkout | (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPubli
]
```

before

```
permission_classes = [permissions.IsAuthenticated&(IsOwnerOfWorkout|(IsReadOnly & (IsCoa
]
```

after

```
permission_classes = [permissions.IsAuthenticated & (IsOwnerOfWorkout | (IsReadOnly & (I
                        ]
```

before

```
return self.list(request,*args, **kwargs)
```

after

```
return self.list(request, *args, **kwargs)
```

before

```
like = WorkoutLike.objects.get(user = self.request.user, workout_id=kwargs.get("workout_
```

after

```
like = WorkoutLike.objects.get(
    user=self.request.user, workout_id=kwargs.get("workout_pk"))
```

before

```
if WorkoutLike.objects.get(user = self.request.user, workout__id=self.kwargs.get("workou
```

after

```
if WorkoutLike.objects.get(user=self.request.user, workout__id=self.kwargs.get("workout_
```

before

```
serializer.save(user=self.request.user, workout=Workout.objects.filter(id=self.kwargs.ge
```

after

```
serializer.save(user=self.request.user, workout=Workout.objects.filter(
    id=self.kwargs.get("workout_pk", None))[0])
```

Q17

before

```
#"likes": reverse("like-list", request=request, format=format),
```

after

before

```
#ordering_fields = ["timestamp"]
```

after

```
#& (IsOwner
        #        | IsOwnerOfWorkout
        #        | (IsReadOnly & (IsCoachOfWorkoutAndVisibleToCoach | IsWorkoutPublic))
        #)
```

after

Q13

before

```
from rest_framework.response import Response
```

after

before

```
import json
```

after

before

```
from workouts.serializers import WorkoutLikeSerializer, WorkoutSerializer, ExerciseSeria
```

after

```
from workouts.serializers import WorkoutSerializer, ExerciseSerializer, WorkoutDataSeria
```

workouts/serializers.py

Q14

before

```
fields = ["url", "id", "name", "description", "duration", "calories", "muscleGroup", "ur
```

after

```
fields = ["url", "id", "name", "description", "duration",
            "calories", "muscleGroup", "unit", "instances"]
```

before

```
ExerciseInstance.objects.create(workout=workout, **exercise_instance_data)
```

after

```
ExerciseInstance.objects.create(
    workout=workout, **exercise_instance_data)
```

before

```
workout=workout, owner=workout.owner, file=file_data.get("file")
```

after

```
workout=workout, owner=workout.owner, file=file_data.get(
    "file")
```

before

```
workout=workout, owner=workout.owner, file=data_file.get("file")
```

after

```
workout=workout, owner=workout.owner, file=data_file.get(
    "file")
```

before

```
instance.visibility = validated_data.get("visibility", instance.visibility)
```

after

```
instance.visibility = validated_data.get(
    "visibility", instance.visibility)
```

before

```
workout=instance, owner=instance.owner, file=data_file.get("file")
```

after

```
workout=instance, owner=instance.owner, file=data_file.get(
    "file")
```

- Method WorkoutSerializer.update is too long

comments/views.py

Q14

before

```
    permission_classes = [
        permissions.IsAuthenticated & IsCommentVisibleToUser & (IsOwner | IsReadOnly)
    ]
```

after

```
    permission_classes = [
        permissions.IsAuthenticated & IsCommentVisibleToUser & (
            IsOwner | IsReadOnly)
    ]
```

Q13

before

```
from django.shortcuts import render
```

after

Q17

before

```
# queryset = Comment.objects.all()
```

after

```
# queryset = Comment.objects.all()
```