```
"""
Author  :       Thomas Fujise
Date    :       08.04.2022
File    :       __init__.py
Version :       1.0.0
Brief   :       Application initialization
"""

from flask import Flask
from flask_login import LoginManager
from flask_sqlalchemy import SQLAlchemy
from importlib import import_module
from flask_mail import Mail



db = SQLAlchemy()
login_manager = LoginManager()
mail = Mail()

def register_extensions(app):
    """
    Initialize extensions

    - LoginManager
    - SQLAlchemy
    - FlaskMail
    """
    db.init_app(app)
    login_manager.init_app(app)
    mail.init_app(app)


def register_blueprints(app):
    """
    Register all blueprints
    """
    for module_name in ('authentication','client', 'coach'):
        module = import_module('apps.{}.routes'.format(module_name))
        app.register_blueprint(module.blueprint)

def config_database(app):
    """
    Configure database
    """
    @app.before_first_request
    def initialize_database():
        db.create_all()

    @app.teardown_request
    def remove_session(exception=None):
        db.session.remove()

def create_app(config):
    app = Flask(__name__)
    app.config.from_object(config)
    register_extensions(app)
    register_blueprints(app)
    config_database(app)
    return app
```

```
"""
Author  :       Thomas Fujise
Date    :       08.04.2022
File    :       config.py
Version :       1.0.0
Brief   :       Set the different config mode
"""

import os
from decouple import config

```

```
class Config(object):
    """
    Config parent class

    Contains all base config settings
    """
    basedir = os.path.abspath(os.path.dirname(__file__))

    SECRET_KEY = config('SECRET_KEY', default='S3cr3t_k3y')

    #Profile picture

    UPLOAD_FOLDER = config('UPLOAD_FOLDER', default='apps/static/assets/img/profile')

    DEFAULT_PROFILE_PIC = config('DEFAULT_PROFILE_PIC', default='default_profile_pic.
    png')

    #SQL ALCHEMY

    SQLALCHEMY_DATABASE_URI = '{}://{}:{}@{}:{}/{}'.format(
        config('DB_ENGINE', default='mysql+pymysql'),
        config('DB_USERNAME', default='fitjourney_dba'),
        config('DB_PASS', default='Super2012$'),
        config('DB_HOST', default='localhost'),
        config('DB_PORT', default='3306'),
        config('DB_NAME', default='fitjourney')
    )
    SQLALCHEMY_TRACK_MODIFICATIONS = False

    #FLASK MAIL
    MAIL_SERVER = config('MAIL_SERVER', default='smtp.mailtrap.io')
    MAIL_PORT = config('MAIL_PORT', default=2525)
    MAIL_USERNAME = config('MAIL_USERNAME', default='5806e099aa3376')
    MAIL_PASSWORD = config('MAIL_PASSWORD', default='c2857a008d8b60')
    MAIL_USE_TLS = config('MAIL_USE_TLS', default=True)
    MAIL_USE_SSL = config('MAIL_USE_SSL', default=False)
    MAIL_DEFAULT_SENDER = config('MAIL_DEFAULT_SENDER', default='thomas.fjs@eduge.ch')

class ProductionConfig(Config):
    """
    Production config class

    Contains all config settings for Production
    """
    DEBUG = False

    #Security
    SESSION_COOKIE_HTTPONLY = True
    REMEMBER_COOKIE_HTTPONLY = True
    REMEMBER_COOKIE_DURATION = 3600

class DebugConfig(Config):
    """
    Debug config class

    Contains all config settings for Debug
    """
    DEBUG = True


config_dict = {
    'Production': ProductionConfig,
    'Debug': DebugConfig
}
```

```
"""
Author  :       Thomas Fujise
Date    :       08.04.2022
File    :       run.py
Version :       1.0.0
Brief   :       Load the config mode and run the flask app
"""
```

```

from flask_migrate import Migrate
from sys import exit
from decouple import config

from apps.config import config_dict
from apps import create_app, db

# By default config set to debug mode
DEBUG = config('DEBUG', default=False, cast=bool)

#Get the config mode
config_mode = 'Debug' if DEBUG else 'Production'

try:
    app_config = config_dict[config_mode.capitalize()]

except KeyError:
    exit('Error: Invalid config mode')

app = create_app(app_config)
Migrate(app,db)

if DEBUG:
    app.logger.info('DEBUG       = ' + str(DEBUG))
    app.logger.info('Environment = ' + config_mode)
    app.logger.info('DBMS        = ' + app_config.SQLALCHEMY_DATABASE_URI)

if __name__ == "__main__":
    app.run()
```

```
"""
Author  :        Thomas Fujise
Date    :        08.04.2022
File    :        __init__.py
Version :        1.0.0
Brief   :        Authentication blueprint initialization
"""

from flask import Blueprint


blueprint = Blueprint(
    'authentication_blueprint',
    __name__,
    url_prefix=''
)
```

```
"""
Author  :        Thomas Fujise
Date    :        12.04.2022
File    :        forms.py
Version :        1.0.0
Brief   :        Authentication forms
"""

from flask_wtf import FlaskForm
from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
    BooleanField, ValidationError
from wtforms.validators import Email, DataRequired, EqualTo, Length
from wtforms.fields.html5 import DateField, EmailField


class LoginForm(FlaskForm):
    """
    Create a login form
    """
    email = StringField('email', id='email_login', validators=[DataRequired()])

    password = PasswordField('Password', id='pwd_login', validators=[DataRequired()])

    login = SubmitField("login")
```

```
24
25 class RegisterForm(FlaskForm):
26     """
27     Create a register form
28     """
29     name = StringField('Name', id='name_register', validators=[DataRequired()])
30
31     surname = StringField('Surname', id='surname_register', validators=[DataRequired()])
32
33     email = EmailField('Email', id="email_rgister", validators=[DataRequired(), Email()])
34
35     birthdate = DateField('Birthdate', id="date_register", validators=[DataRequired()])
36
37     password = PasswordField('Password', id='password_register', validators=[DataRequired(), EqualTo('confirm_password', message='Passwords Must Match!')])
38
39     confirm_password = PasswordField('Confirm Password', id='password2_register')
40
41     register = SubmitField("register")
42
43
44 class UpdateForm(FlaskForm):
45     """
46     Create a update form
47     """
48
49     name = StringField('Name', id='name_update', validators=[DataRequired()])
50
51     surname = StringField('Surname', id='surname_update', validators=[DataRequired()])
52
53     email = EmailField('Email', id="email_update", validators=[DataRequired(), Email()])
54
55     birthdate = DateField('Birthdate', id="birthdate_update", validators=[DataRequired()])
56
57     profile_pic = FileField('Profile Pic', id="profile_pic_update")
58
59     card_id = IntegerField()
60
61     address = StringField('Address', id='address_update')
62
63     country = StringField('Country', id='country_update')
64
65     city = StringField('City', id='city_update')
66
67     npa = StringField('NPA', id='npa_update')
68
69     update = SubmitField("Update")
```

```
1  """
2  Author  :        Thomas Fujise
3  Date    :        11.04.2022
4  File    :        models.py
5  Version :        1.0.0
6  Brief   :        ALl models needed by the application
7  """
8
9  from flask_login import UserMixin
10 from datetime import datetime
11 from apps import db, login_manager
12
13 from apps.authentication.util import hash_pass
14
15 ACCESS = {
16     'Guest': 0,
17     'Client': 1,
18     'Coach': 2
19 }
20 class User(db.Model, UserMixin):
```

```
21
22      __tablename__ = 'USER'
23
24      id = db.Column(db.Integer, primary_key=True)
25      name = db.Column(db.String(64), nullable=False)
26      surname = db.Column(db.String(64), nullable=False)
27      email = db.Column(db.String(64), unique=True, nullable=False)
28      password = db.Column(db.LargeBinary, nullable=False)
29      birthdate = db.Column(db.Date, nullable=False)
30      card_id = db.Column(db.Integer, unique=True)
31      profile_pic = db.Column(db.String(250), nullable=True)
32      city = db.Column(db.String(64), nullable=True)
33      country = db.Column(db.String(64), nullable=True)
34      address = db.Column(db.String(64), nullable=True)
35      npa = db.Column(db.String(32), nullable=True)
36      is_active = db.Column(db.Boolean)
37      created_at = db.Column(db.DateTime, default=datetime.utcnow)
38      role = db.Column(db.Integer, db.ForeignKey('ROLE.id'))
39
40
41      def __init__(self, **kwargs):
42          for property, value in kwargs.items():
43              # depending on whether value is an iterable or not, we must un pack it's
    value
44              # when **kwargs is request.form, some values will be a 1-element list
45              if hasattr(value, '__iter__') and not isinstance(value, str):
46                  value = value[0]
47
48              if property == 'password':
49                  value = hash_pass(value)  # we need bytes here (not plain str)
50
51              setattr(self, property, value) # set the attribute
52
53      def __repr__(self):
54          return str(self.email)
55
56      def is_authenticated(self):
57          return True
58
59      def is_active(self):
60          return True
61
62      def is_anonymous(self):
63          return False
64
65      def get_id(self):
66          return str(self.id)
67
68      def is_coach(self):
69          return self.role == ACCESS['Coach']
70
71
72  class PhysicalInfo(db.Model):
73      __tablename__ = "PHYSICAL_INFO"
74      id = db.Column(db.Integer, primary_key=True)
75      user_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
76
77      height = db.Column(db.Integer, nullable=False)
78      weight = db.Column(db.Numeric(4,1), nullable=False)
79      age = db.Column(db.Integer, nullable=False)
80      bmi = db.Column(db.Numeric(3,1), nullable=False)
81      bmr = db.Column(db.Numeric(5,1), nullable=False)
82      bodyfat_percentage = db.Column(db.Numeric(3,1), nullable=False)
83      muscle_mass_percentage = db.Column(db.Numeric(3,1), nullable=False)
84      bone_mass_percentage = db.Column(db.Numeric(3,1), nullable=False)
85      water_percentage = db.Column(db.Numeric(3,1), nullable=False)
86      protein_percentage = db.Column(db.Numeric(3,1), nullable=False)
87      bone_mass = db.Column(db.Numeric(3,1), nullable=False)
88      muscle_mass = db.Column(db.Numeric(4,1), nullable=False)
89      bodyfat_mass = db.Column(db.Numeric(4,1), nullable=False)
90      leanbody_mass = db.Column(db.Numeric(4,1), nullable=False)
91      fat_visceral = db.Column(db.Numeric(4,1), nullable=False)
```

```
    body_age = db.Column(db.Integer, nullable=False)
    date = db.Column(db.Date, nullable=False)
    front_photo = db.Column(db.String(250), nullable=True)
    right_side_photo = db.Column(db.String(250), nullable=True)
    left_side_photo = db.Column(db.String(250), nullable=True)
    back_photo = db.Column(db.String(250), nullable=True)


class Role(db.Model):
    __tablename__ = 'ROLE'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(32))

class Workout(db.Model):
    __tablename__ = 'WORKOUT'

    id = db.Column(db.Integer, primary_key=True)
    workout_type = db.Column(db.Integer, db.ForeignKey('WORKOUT_TYPE.id'))
    client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
    date = db.Column(db.DateTime)
    duration = db.Column(db.Time)
    heart_rate_max = db.Column(db.Numeric)
    heart_rate_min = db.Column(db.Numeric)
    heart_rate_avg = db.Column(db.Numeric)
    calories = db.Column(db.Numeric)
    active_calories = db.Column(db.Numeric)
    distance = db.Column(db.Numeric, nullable=True)
    pace_avg = db.Column(db.Time, nullable=True)

class WorkoutType(db.Model):
    __tablename__ = 'WORKOUT_TYPE'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(32))
    description = db.Column(db.String(250))
    logo = db.Column(db.String(250))

class Program(db.Model):
    __tablename__ = 'PROGRAM'

    id = db.Column(db.Integer, primary_key=True)
    type = db.Column(db.String(32))
    pdf = db.Column(db.LargeBinary(length=(2**32)-1))
    date = db.Column(db.Date)
    client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
    coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'))

class Session(db.Model):
    __tablename__ = 'SESSION'

    id = db.Column(db.Integer, primary_key=True)
    start_time = db.Column(db.DateTime, nullable=False)
    end_time = db.Column(db.DateTime, nullable=False)
    duration = db.Column(db.Time, nullable=False)
    workout_type = db.Column(db.Integer, db.ForeignKey('WORKOUT_TYPE.id'), nullable=
    False)
    client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
    coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'))

class Subscription(db.Model):
    __tablename__ = 'SUBSCRIPTION'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(64))
    cost = db.Column(db.Integer)
    duration = db.Column(db.Integer, unique=True)

class Purchase(db.Model):
    __tablename__ = 'PURCHASE'

    id = db.Column(db.Integer, primary_key=True)
```

```
    client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
    date = db.Column(db.Date)
    subscription_id = db.Column(db.Integer, db.ForeignKey('SUBSCRIPTION.id'))

class CoachedBy(db.Model):
    __tablename__ = 'COACHEDBY'

    client_id = db.Column(db.Integer, db.ForeignKey('USER.id'), primary_key=True)
    coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'), primary_key=True)
    starting_date = db.Column(db.Date)
    end_date = db.Column(db.Date, nullable=True)

class CoachingReview(db.Model):
    __tablename__ = 'COACHING_REVIEW'

    id = db.Column(db.Integer, db.ForeignKey('REVIEW.id') , primary_key=True)
    satisfaction = db.Column(db.Integer, nullable=False)
    support = db.Column(db.Integer, nullable=False)
    disponibility = db.Column(db.Integer, nullable=False)
    advice = db.Column(db.Integer, nullable=False)
    target_id = db.Column(db.Integer, db.ForeignKey('USER.id'))

class WorkoutReview(db.Model):
    __tablename__ = 'WORKOUT_REVIEW'

    id = db.Column(db.Integer, db.ForeignKey('REVIEW.id') ,primary_key=True)
    difficulty = db.Column(db.Integer, nullable=False)
    feel = db.Column(db.Integer, nullable=False)
    fatigue = db.Column(db.Integer, nullable=False)
    energy = db.Column(db.Integer, nullable=False)
    target_id = db.Column(db.Integer, db.ForeignKey('WORKOUT.id'))

class Review(db.Model):
    __tablename__ = 'REVIEW'

    id = db.Column(db.Integer, primary_key=True)
    comment = db.Column(db.String(250), nullable=True)
    date = db.Column(db.DateTime)
    type = db.Column(db.String(15), nullable=False)
    id_client = db.Column(db.Integer, db.ForeignKey('USER.id'))

@login_manager.user_loader
def user_loader(id):
    return User.query.filter_by(id=id).first()


@login_manager.request_loader
def request_loader(request):
    email = request.form.get('email')
    user = User.query.filter_by(email=email).first()
    return user if user else None
```

```
"""
Author  :       Thomas Fujise
Date    :       11.04.2022
File    :       routes.py
Version :       1.0.0
Brief   :       Set all authentication routes
"""

from flask import render_template, redirect, request, url_for, flash
from flask_login import (
    current_user,
    login_user,
    logout_user
)
from flask_login import login_required

from apps import db, login_manager
from apps.authentication import blueprint
from apps.authentication.forms import LoginForm, RegisterForm
from apps.authentication.models import User
```

```python
from apps.authentication.util import verify_pass



#Default route
@blueprint.route('/')
def route_default():
    return redirect(url_for('authentication_blueprint.login'))


#Login route
@blueprint.route('/login', methods=['GET', 'POST'])
def login():
    """
    Display login page with the login form
    """
    login_form = LoginForm()
    if login_form.validate_on_submit():
        #read form data
        email = request.form['email']
        password = request.form['password']

        #Locate the user
        user = User.query.filter_by(email=email).first()
        #Check password
        if user and verify_pass(password, user.password):
            login_user(user)
            return redirect(url_for('authentication_blueprint.route_default'))

        # Something is not ok (user or password)
        flash('Wrong user or password', 'danger')
        return redirect(url_for('authentication_blueprint.route_default', msg='Wrong user or password'))

    if not current_user.is_authenticated:

        return render_template('accounts/login.html',form=login_form)

    if current_user.role  == 1:
        #CLIENT
        return redirect(url_for('client_blueprint.index'))
    elif current_user.role == 2:
        #COACH
        return redirect(url_for('coach_blueprint.dashboard'))

    return redirect(url_for('client_blueprint.index'))


#Register route
@blueprint.route('/register', methods=['GET', 'POST'])
def register():
    """
    Display Register page with the register form
    """
    register_form = RegisterForm()

    if register_form.validate_on_submit():

        email = request.form['email']
        #name = request.form['name']

        #Check if user already exists
        user = User.query.filter_by(email=email).first()

        if user:
            flash("Email already registered", 'warning')
            return render_template('accounts/register.html', msg='Email Already registered', success=False, form=register_form)


         #try to create the user
        try:
```

```
            user = User(name=request.form['name'], surname=request.form['surname'], email=request.form['email'], birthdate=request.form['birthdate'], password=request.form['password'], role=2)
            db.session.add(user)
            db.session.commit()
            flash("User has been created please login", 'success')
            return redirect(url_for('authentication_blueprint.login'))
        except:
            db.session.rollback()
            flash("Error while registering new user", 'danger')
            return redirect(url_for('authentication_blueprint.register'))



        return render_template('accounts/register.html', msg='<a href="/login"> LOGIN</a> ', success=True, form=register_form)

    else:
        return render_template('accounts/register.html', form=register_form)


@blueprint.route('/logout')
@login_required
def logout():

    logout_user()

    return redirect(url_for('authentication_blueprint.login'))

# Errors

@login_manager.unauthorized_handler
def unauthorized_handler():
    return render_template('errors/page-403.html'), 403


@blueprint.errorhandler(403)
def access_forbidden(error):
    return render_template('errors/page-403.html'), 403


@blueprint.errorhandler(404)
def not_found_error(error):
    return render_template('errors/page-404.html'), 404


@blueprint.errorhandler(500)
def internal_error(error):
    return render_template('errors/page-500.html'), 500
```

```
"""
Author  :       Thomas Fujise
Date    :       12.04.2022
File    :       util.py
Version :       1.0.0
Brief   :       Util functions file for authentication
"""

import os
import hashlib
import binascii

def hash_pass(password):
    """
    Function that hash a password for storing
    Encodes a provided password in a way that is safe to store on a database

    Parameter(s) :
    NAME      |  TYPE  | DESC
    password  | STRING | The password to hash

    Return :
```

```
    | BYTES | The password hashed with salt

    """
    salt = hashlib.sha256(os.urandom(60)).hexdigest().encode('ascii')
    pwdhash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'), salt, 100000)
    pwdhash = binascii.hexlify(pwdhash)

    return (salt + pwdhash) # return bytes


def verify_pass(provided_password, stored_password):
    """
    Function that verify a stored password against one provided by the user
    Given an encoded password and a plain text one which is provided by the user, it
    verifies whether the provided password matches the encoded one.

    Parameter(s) :
    NAME                |  TYPE   | DESC
    provided_password   | STRING | The password provided by the user
    stored_password     | BYTES | The hashed with salt password stored in db

    Return :
    | BOOLEAN | True if passwords are the same, else False

    """

    stored_password = stored_password.decode('ascii')
    salt = stored_password[:64]
    stored_password = stored_password[64:]
    pwdhash = hashlib.pbkdf2_hmac('sha512', provided_password.encode('utf-8'), salt.
    encode('ascii'), 100000)

    pwdhash = binascii.hexlify(pwdhash).decode('ascii')

    return pwdhash == stored_password
```

```
"""
Author  :       Thomas Fujise
Date    :       08.04.2022
File    :       __init__.py
Version :       1.0.0
Brief   :       Coach blueprint initialization
"""

from flask import Blueprint

blueprint = Blueprint(
    'coach_blueprint',
    __name__,
    url_prefix=''
)
```

```
"""
Author  :       Thomas Fujise
Date    :       24.05.2022
File    :       forms.py
Version :       1.0.0
Brief   :       Coach forms
"""

from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileAllowed, FileRequired
from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
    BooleanField, ValidationError, SelectField, TextAreaField, HiddenField,
    DecimalField
from wtforms.validators import Email, DataRequired, EqualTo, Length, Regexp
from wtforms.fields.html5 import DateField, EmailField, IntegerRangeField, TimeField
from wtforms.ext.dateutil.fields import DateTimeField
from datetime import datetime

class SessionForm(FlaskForm):
    """
```

```
    Create an adding session form
    """

    client = SelectField('Clients', id='clients_list', coerce=int)

    date = DateField('Date', id='datepick', validators=[DataRequired()])

    start_time = TimeField('Start Time', id="start_time", validators=[DataRequired()])

    end_time = TimeField('End Time', id="end_time", validators=[DataRequired()])

    duration = SelectField('Duration', id="duration", coerce=int, validators=[
    DataRequired()])

    type = SelectField('Type', id="type_picker", coerce=int)

    add = SubmitField("Add")

class AddClientForm(FlaskForm):
    """
    Create an adding client form
    """

    name = StringField('Name', id="name", validators=[DataRequired()])

    surname = StringField('Surname', id="surname", validators=[DataRequired()])

    email = EmailField('Email', id="email", validators=[DataRequired()])

    birthdate = DateField('Birthdate', id="birthdate", validators=[DataRequired()])

    height = IntegerField('Height', id="height", validators=[DataRequired()])

    start_date = DateField('Start date', id="start_date",  validators=[DataRequired()])

    subscription = SelectField('Subscription', id="subscription",  validators=[
    DataRequired()])

    end_date = DateField('End date', id="end_date",  validators=[DataRequired()])

    add = SubmitField("Add")

class RenewSubscriptionForm(FlaskForm):
    """
    Create a renew subscription form
    """

    name = StringField('Name', id="name", validators=[DataRequired()])

    surname = StringField('Surname', id="surname", validators=[DataRequired()])

    email = EmailField('Email', id="email", validators=[DataRequired()])

    start_date = DateField('Start date', id="start_date",  validators=[DataRequired()])

    end_date = DateField('End date', id="end_date",  validators=[DataRequired()])

    subscription = SelectField('Subscription', id="subscription",  validators=[
    DataRequired()])

    submit = SubmitField("Submit")

class ClientForm(FlaskForm):
    """
    Create a update form
    """

    name = StringField('Name', id='name', validators=[DataRequired()])

    surname = StringField('Surname', id='surname', validators=[DataRequired()])

    email = EmailField('Email', id="email", validators=[DataRequired(), Email()])
```

```
88
89      birthdate = DateField('Birthdate', id="birthdate", validators=[DataRequired()])
90
91      profile_pic = FileField('Profile Pic', id="profile_pic")
92
93      card_id = StringField('Card ID', id='card_update')
94
95      weight = StringField('Weight', id='weight_update')
96
97      height = StringField('Height', id='height_update')
98
99      subscriptionEnd = DateField('Subscription Until', id='subscription_until')
100
101     address = StringField('Address', id='address')
102
103     country = StringField('Country', id='country')
104
105     city = StringField('City', id='city')
106
107     npa = StringField('NPA', id='npa')
108
109     register_date = DateField('Register date', id='register_date')
110
111     change = SubmitField("Change")
112
113
114 class AddProgramForm(FlaskForm):
115     """
116     Create the adding program form
117     """
118
119     type = SelectField('Type', id="program_type",  validators=[DataRequired()], choices
        =[("Diet", "Diet"), ("Workout", "Workout")])
120
121     file = FileField('Program File', id="program_file", validators=[FileRequired(),
        FileAllowed(['pdf'], 'PDF only !')])
122
123     client = HiddenField('Client', id="program_client")
124
125     add = SubmitField("Upload")
126
127 class AddCheckUpForm(FlaskForm):
128     """
129     Create the check up form
130     """
131
132     name = StringField('Name', id='name', validators=[DataRequired()])
133
134     surname = StringField('Surname', id='surname', validators=[DataRequired()])
135
136     height = IntegerField('Height', id='height', validators=[DataRequired()])
137
138     age = IntegerField('Age', id='age', validators=[DataRequired()])
139
140     weight = DecimalField('Weight', id='weight', validators=[DataRequired()])
141
142     bmi = DecimalField('BMI', id='bmi', validators=[DataRequired()])
143
144     water = DecimalField('Water %', id='water', validators=[DataRequired()])
145
146     protein = DecimalField('Protein %', id='protein', validators=[DataRequired()])
147
148     muscle_mass_percent = DecimalField('Muscle mass %', id='muscle_mass_percent',
        validators=[DataRequired()])
149
150     body_fat_percent = DecimalField('Body Fat %', id='body_fat_percent', validators=[
        DataRequired()])
151
152     bone_mass_percent = DecimalField('Bone Mass %', id='bone_mass_percent', validators
        =[DataRequired()])
153
154     bmr = DecimalField('BMR', id='bmr', validators=[DataRequired()])
```

```
155
156     muscle_mass = DecimalField('Muscle mass', id='muscle_mass', validators=[
        DataRequired()])
157
158     body_fat = DecimalField('Body fat', id='body_fat', validators=[DataRequired()])
159
160     fat_visceral = DecimalField('Fat Visceral', id='fat_visceral', validators=[
        DataRequired()])
161
162     bone_mass = DecimalField('Bone Mass', id='bone_mass', validators=[DataRequired()])
163
164     lean_body_mass = DecimalField('Lean body mass', id='lean_body_mass', validators=[
        DataRequired()])
165
166     body_age = DecimalField('Body afe', id='body_age', validators=[DataRequired()])
167
168     add = SubmitField("Add")
```

```
1  from smartcard.CardRequest import CardRequest
2  from smartcard.Exceptions import CardRequestTimeoutException
3  from smartcard.CardType import AnyCardType
4  from smartcard import util
5  import time
6
7  def get_card_id():
8      card_type = AnyCardType()
9
10     request = CardRequest(timeout=30, cardType=card_type)
11
12     card = None
13
14     while card == None:
15         time.sleep(0.1)
16
17         service = None
18         try:
19             service = request.waitforcard()
20         except:
21             print("ERROR: No card detected")
22             break
23
24         conn = service.connection
25         conn.connect()
26
27         get_uid = util.toBytes("FF CA 00 00 00")
28
29         data, sw1, sw2 = conn.transmit(get_uid)
30
31         card = ' '.join(str(e) for e in data)
32
33     return card
```

```
1  """
2  Author  :        Thomas Fujise
3  Date    :        23.05.2022
4  File    :        routes.py
5  Version :        1.0.0
6  Brief   :        Set all the coach routes
7  """
8
9  # APP
10 from apps.coach import blueprint
11 from apps import db, login_manager, mail
12 from apps.authentication.models import User, PhysicalInfo, Subscription, CoachingReview
       , WorkoutReview, Review, Workout, WorkoutType, Session, Program, CoachedBy,
       Purchase
13 from apps.coach.forms import SessionForm, AddClientForm, ClientForm, AddProgramForm,
       AddCheckUpForm, RenewSubscriptionForm
14 from apps.config import Config
15
16 # FLASK
17 from flask import render_template, redirect, request, url_for, flash, send_file
```

```
from flask_login import login_required
from jinja2 import TemplateNotFound
from flask_mail import Message
from flask_login import (
    current_user
)

# UTILS
from io import BytesIO
import uuid as uuid
import os
from werkzeug.utils import secure_filename
from datetime import date, datetime

from apps.coach.util import *
from apps.client.util import *
from apps.coach.reader import get_card_id

@blueprint.route('/dashboard')
@login_required
def dashboard():
    #Check if user is coach
    if not current_user.is_coach():
        return redirect(url_for('authentication_blueprint.login'))

    nextClient = get_coach_next_session(current_user.id)
    clientLastWorkout = get_last_workout(nextClient.id) if nextClient != None else None

    clients = get_clients(current_user.id)
    return render_template('coach/dashboard.html', segment='coach_dashboard',
    nextClient=nextClient, clientLastWorkout=clientLastWorkout, clients=clients)


@blueprint.route('/calendar', methods=['POST', 'GET'])
@login_required
def calendar():
    #Check if user is coach
    if not current_user.is_coach():
        return redirect(url_for('authentication_blueprint.login'))

    form = SessionForm(request.form)
    # Set select input choices
    form.client.choices = [(client['id'], str(client['name'] + " " + client['surname'])
    ) for client in get_clients(current_user.id)]
    form.duration.choices = [(i,i) for i in range(1,4)]
    form.type.choices = [(type.id, type.title) for type in get_all_workout_types()]

    sessions = get_sessions(current_user.id)
    #print(sessions)

    if request.method == 'POST':
        #Check if client there is client in the select field
        if 'client' not in request.form:
            flash("You don't have client to select, please add a client before", '
    danger')
            return redirect( url_for('coach_blueprint.calendar') )

        startDate = request.form['date']
        startTime = request.form['start_time']
        endTime = request.form['end_time']
        durationSelected = request.form['duration']

        # Set datetime value
        startDateTime = datetime.strptime(startDate + " " + startTime, '%Y-%m-%d %H:%M'
    )
        endDateTime = datetime.strptime(startDate + " " + endTime, '%Y-%m-%d %H:%M:%S')

        # Set duration format
        duration = "0{durationSelected}:00:00".format(durationSelected=durationSelected
    )
        client = get_client_details(request.form['client'])
        try :
```

```
85              newSession = Session(start_time=startDateTime, end_time=endDateTime,
    duration=duration, workout_type=request.form['type'], client_id=request.form['
    client'], coach_id=current_user.id)
86              db.session.add(newSession)
87              db.session.commit()
88              flash("The session is registered !", 'success')
89              #set and send mail
90              msg = Message('New session register', recipients =[client.email])
91              msg.body = "Hey "+ client.name + ", your coach just registered a new
    session with you on  "+ startDateTime.strftime("%d %B, %Y") + " at " +
    startDateTime.strftime("%I%p") + " for " + durationSelected +" hour(s)"
92              mail.send(msg)
93              return redirect( url_for('coach_blueprint.calendar') )
94          except:
95              db.session.rollback()
96              flash("Error while trying to add a new session, please try again", 'danger'
    )
97              return redirect( url_for('coach_blueprint.calendar') )
98
99      return render_template('coach/calendar.html', form=form, sessions=sessions)
100
101 @blueprint.route('/client', methods=['POST', 'GET'])
102 @login_required
103 def client():
104     #Check if user is coach
105     if not current_user.is_coach():
106         return redirect(url_for('authentication_blueprint.login'))
107
108     form = ClientForm(request.form)
109     #Get client id
110     client_id = request.args.get('clientId')
111
112     #Get client infos to display
113     reviews = get_reviews(client_id)
114     clientDetails = get_client_details(client_id)
115     subscriptionUntil = get_subscription_end_date(client_id)
116     physicalInfo = get_physical_infos(client_id)
117
118     # Get each workout type and how many client have made in 2 separate array to use
    them in js
119     wrktTypeList = get_workout_type_count(client_id)[0]
120     wrktTypeCount = get_workout_type_count(client_id)[1] if get_workout_type_count(
    client_id)[1] != [] else 0
121
122     nbWorkoutPerMonth = get_workout_count_per_month(client_id)
123
124     programs = get_program(client_id)
125     workoutProgram = programs[0]
126     dietProgram = programs[1]
127
128     # Get the average of heart rate during this week
129     avgHeartRate = get_average_heart_rate_last_week(client_id)
130   # Get the average of calories burned this week
131     avgCalories = get_average_calories_last_week(client_id)
132     # Get total time of training this week
133     totalTime = get_time_working_out_last_week(client_id)
134     # Get the average weight recorded each month
135     weightUpdate = get_weight_update(client_id)
136
137     # Get all check ups
138     checkUps = get_all_check_up(client_id)
139     return render_template('coach/client.html', segment='client', form=form, reviews=
    reviews, client=clientDetails, subscriptionUntil=subscriptionUntil,
140      wrktTypeCount=wrktTypeCount, wrktTypeList=wrktTypeList, nbWorkoutPerMonth=
    nbWorkoutPerMonth, workoutProgram=workoutProgram, dietProgram=dietProgram,
    physicalInfo=physicalInfo,
141      avgCalories=avgCalories, avgHeartRate=avgHeartRate, totalTime=totalTime,
    weightUpdate=weightUpdate, checkUps=checkUps)
142
143
144 @blueprint.route('/add_client', methods=['POST', 'GET'])
145 @login_required
```

```
146 def add_client():
147     #Check if user is coach
148     if not current_user.is_coach():
149         return redirect(url_for('authentication_blueprint.login'))
150 
151     form = AddClientForm(request.form)
152 
153     form.subscription.choices = [(subscription.duration, subscription.title + " - " +
    str(subscription.cost) + "CHF") for subscription in get_all_subscriptions()]
154 
155     if request.method == 'POST':
156         height = request.form['height']
157         surname = request.form['surname']
158         email = request.form['email']
159         name = request.form['name']
160         birthdate = request.form['birthdate']
161         date = request.form['start_date']
162         end_date = request.form['end_date']
163         subscription = request.form['subscription']
164 
165         try:
166             #Add client
167             newClient = User(name=name,surname=surname, email=email, birthdate=
    birthdate, profile_pic=Config.DEFAULT_PROFILE_PIC,role=1, password=name+"123")
168             db.session.add(newClient)
169             db.session.flush()
170             #Assign coach and add purchase of the subscription selected
171             newCoachAssign = CoachedBy(client_id=newClient.id, coach_id=current_user.id
    , starting_date=date, end_date=end_date)
172             newPurchase = Purchase(client_id=newClient.id, date=date, subscription_id=
    get_subscription_id(subscription).id)
173 
174             db.session.add(newPurchase)
175             db.session.add(newCoachAssign)
176             db.session.commit()
177             flash("New client is registered !", 'success')
178             #Set and send mail
179             msg = Message('Your account is now created ', recipients =[newClient.email
    ])
180             msg.body = "Hey "+ newClient.name + ", your Fitjourney account just been
    created. You can access to it by filling the login form with : \r\n Email : "+
    newClient.email  + " \r\n Password : " + newClient.name + "123 \r\n Thank you. \r\n
     Fitjourney"
181             mail.send(msg)
182             return redirect( url_for('coach_blueprint.dashboard') )
183         except:
184             db.session.rollback()
185             flash("Error, while trying to register new client, please try again", '
    danger')
186             return redirect(url_for('coach_blueprint.dashboard'))
187 
188 
189     return render_template('coach/add_client.html', segment='add_client', form=form)
190 
191 @blueprint.route('/add_program', methods=['POST', 'GET'])
192 @login_required
193 def add_program():
194     #Check if user is coach
195     if not current_user.is_coach():
196         return redirect(url_for('authentication_blueprint.login'))
197 
198     form=AddProgramForm()
199     #Get client id
200     client_id = request.args.get('clientId')
201 
202     if request.method == "POST":
203         if form.validate_on_submit():
204 
205             file_name = form.file.data
206             client = get_client_details(request.form['client'])
207             try:
208                 newProgram = Program(type=request.form['type'], pdf=file_name.read(),
```

```
    date=date.today(), client_id=request.form['client'], coach_id=current_user.id)
209                 db.session.add(newProgram)
210                 db.session.commit()
211                 flash("Program Added !", 'success')
212                 #Set and send mail
213                 msg = Message('New program available', recipients =[client.email])
214                 msg.body = "Hey "+ client.name + ", your coach just added your new "+
    request.form['type']  + " program \r\n Fitjourney"
215                 mail.send(msg)
216                 return redirect(url_for('coach_blueprint.client', clientId=request.form
    ['client']))
217             except:
218                 db.session.rollback()
219                 flash("Error while adding the new program", 'danger')
220                 return redirect(url_for('coach_blueprint.client', clientId=request.form
    ['client']))
221
222     return render_template('coach/add_program.html', segment='add_program', form=form,
    clientId=client_id)
223
224
225 @blueprint.route('/program', methods=['GET'])
226 @login_required
227 def program():
228
229     program_id = request.args.get('programId')
230     program_type = request.args.get('programType')
231
232     program = get_program_by_id(program_id)
233
234     return send_file(BytesIO(program.pdf), attachment_filename=str(program_type+'.pdf')
    , as_attachment=True)
235
236
237 @blueprint.route('/check_up', methods=['POST', 'GET'])
238 @login_required
239 def check_up():
240     #Check if user is coach
241     if not current_user.is_coach():
242         return redirect(url_for('authentication_blueprint.login'))
243
244     form = AddCheckUpForm(request.form)
245     client_id = request.args.get('clientId')
246
247     # Get the client details
248     infos = get_client_details(client_id)
249     today = date.today()
250     # Set the static infos for the check up forms
251     static_infos = {
252         "id" : infos.id,
253         "name" : infos.name,
254         "surname" : infos.surname,
255         "age" : (today.year - infos.birthdate.year - ((today.month, today.day) < (infos
    .birthdate.month, infos.birthdate.day))) #Get age with birthdate and date.today()
256     }
257
258     if request.method == "POST":
259         if form.validate_on_submit():
260             try:
261                 newCheckup = PhysicalInfo(user_id=client_id, height=request.form['
    height'], weight=request.form['weight'], age=request.form['age'],
262                     bmi=request.form['bmi'], bmr=request.form['bmr'],
    bodyfat_percentage=request.form['body_fat_percent'], muscle_mass_percentage=request
    .form['muscle_mass_percent'],
263                     bone_mass_percentage=request.form['bone_mass_percent'],
    water_percentage=request.form['water'], protein_percentage=request.form['protein'],
264                     bone_mass=request.form['bone_mass'], muscle_mass=request.form['
    muscle_mass'], bodyfat_mass=request.form['body_fat'], leanbody_mass=request.form['
    lean_body_mass'],
265                     fat_visceral=request.form['fat_visceral'], body_age=request.form['
    body_age'], date=today)
266                 db.session.add(newCheckup)
```

```
                db.session.commit()
                flash("Check up added !", 'success')
                return redirect(url_for('coach_blueprint.client', clientId=client_id))
            except:
                flash("Error while trying to add check up", 'danger')
                return redirect(url_for('coach_blueprint.client', clientId=client_id))

    return render_template('coach/add_checkup.html', segment='add_checkup', form=form, infos=static_infos)

@blueprint.route('/new_subscription', methods=['POST', 'GET'])
@login_required
def new_subscription():
    #Check if user is coach
    if not current_user.is_coach():
        return redirect(url_for('authentication_blueprint.login'))

    form=RenewSubscriptionForm(request.form)

    #Set the subscription choices
    form.subscription.choices = [(subscription.duration, subscription.title + " - " + str(subscription.cost) + "CHF") for subscription in get_all_subscriptions()]

    client_id = request.args.get('clientId')
    # Get the client details
    client = get_client_details(client_id)
    if request.method == "POST":
        if form.validate_on_submit():
            try:
                print(get_subscription_id(request.form['subscription']).id)
                newCoachAssign = CoachedBy(client_id=client_id, coach_id=current_user.id, starting_date=request.form['start_date'], end_date=request.form['end_date'])
                newPurchase = Purchase(client_id=client_id, date=request.form['start_date'], subscription_id=get_subscription_id(request.form['subscription']).id)
                print("OK")
                db.session.add(newPurchase)
                db.session.add(newCoachAssign)
                db.session.commit()
                flash("New subscription is purchased !", 'success')
                msg = Message('New subscription purchased', recipients =[client.email])
                msg.body = "Hey "+ client.name + ", you just purchased a new subscription for " + request.form['subscription'] + " month(s).  \r\n Thank you. \r\n Fitjourney"
                mail.send(msg)
                return redirect( url_for('coach_blueprint.client', clientId=client_id) )
            except:
                db.session.rollback()
                flash("Error, while trying to purchase new subscription", 'danger')
                return redirect(url_for('coach_blueprint.client', clientId=client_id))

    return render_template('coach/new_subscription.html', segment='new_subscription', form=form, client=client)

@blueprint.route('/new_card', methods=['GET'])
@login_required
def new_card():
    #Check if user is coach
    if not current_user.is_coach():
        return redirect(url_for('authentication_blueprint.login'))

    client_id = request.args.get('clientId')
    card = get_card_id()
    client = get_client_details(client_id)
    if card == None:
        flash("No new card was detected", 'danger')
        return redirect(url_for('coach_blueprint.client', clientId=client_id))

    if update_card_id(client_id, card):
        flash("Member card updated", 'success')
    else:
        flash("Error while updating the member card, please try again", 'danger')
```

```
331
332     return redirect(url_for('coach_blueprint.client', clientId=client_id))
333
334 @blueprint.route('/cancel_subscription', methods=['GET'])
335 @login_required
336 def cancel_subscription():
337     #Check if user is coach
338     if not current_user.is_coach():
339         return redirect(url_for('authentication_blueprint.login'))
340
341     client_id = request.args.get('clientId')
342
343     if cancel_last_subscription(client_id):
344         flash("Subscription canceled", 'success')
345     else:
346         flash("Error while trying to cancel subscription", 'danger')
347
348     return redirect(url_for('coach_blueprint.client', clientId=client_id))
```

```
1  """
2  Author  :        Thomas Fujise
3  Date    :        18.05.2022
4  File    :        util.py
5  Version :        1.0.0
6  Brief   :        All the functions needed to get datas for coach templates
7  """
8
9  # APP
10 from apps import db
11 from apps.authentication.models import User, PhysicalInfo, Subscription, Purchase,
       CoachingReview, WorkoutReview, Review, Workout, WorkoutType, Session, CoachedBy,
       Program
12
13 # SQL ALCHEMY
14 from sqlalchemy import union, func
15
16 # UTILS
17 from dateutil.relativedelta import relativedelta
18
19 import json
20
21 from datetime import date
22
23 from apps.client.util import *
24
25 def get_last_workout(userId):
26     """
27     Get a user last workout done
28
29      Parameter(s):
30      NAME      |  TYPE   | DESC
31      userId    |  INT    | the id of the user
32
33      Return :
34      | Workout() | Workout object only with the properties selected with the query (
       date)
35     """
36     lastWorkout = db.session.query(Workout.date).filter(Workout.client_id==userId).
       order_by(Workout.date.desc()).first()
37
38     return lastWorkout
39
40 def get_coach_next_session(coachId):
41     """
42     Get the next session of a coach
43
44     Parameter(s):
45      NAME      |  TYPE   | DESC
46      coachId    |   INT    | the id of the coach
47
48      Return :
49      | Query() | Query object with all the properties selected
```

```
50
51      SQL :
52          SELECT USER.id, USER.name, USER.surname, USER.card_id, USER.profile_pic SESSION
        .start_time, SESSION.end_time, SESSION.duration, WORKOUT_TYPE.logo
53          FROM SESSION
54          JOIN USER ON SESSION.client_id = USER.id
55          JOIN WORKOUT_TYPE ON WORKOUT_TYPE.id = SESSION.workout_type
56          WHERE SESSION.start_time >= current_timestamp()
57          ORDER BY SESSION.start_time ASC
58      """
59
60      nextSession = db.session.query(User.id, User.name, User.surname, User.card_id, User
        .profile_pic, Session.start_time, Session.end_time, Session.duration, WorkoutType.
        logo.label("workoutLogo")).join(User, User.id==Session.client_id).join(WorkoutType,
         WorkoutType.id==Session.workout_type).filter(Session.coach_id==coachId).filter(
        Session.start_time>=func.current_timestamp()).order_by(Session.start_time.asc()).
        first()
61      return nextSession
62
63
64
65  def is_active(userId):
66      """
67      Check if user was active in the last 24h (by checking if he done a workout)
68
69      Parameter(s):
70       NAME      |  TYPE   | DESC
71       userId    |  INT    | the id of the user
72
73       Return :
74       | Boolean | True if the user done a workout in the last 24h, else False
75      """
76      isActive = db.session.query(Workout.date).filter(Workout.client_id==userId).filter(
        Workout.date>=func.current_date()-1).first() is not None
77
78      return isActive
79
80  def get_clients(coachId):
81      """
82      Get all clients coached by a coach
83
84      Parameter(s):
85       NAME      |  TYPE   | DESC
86       coachId    |  INT    | the id of the coach
87
88       Return :
89       | Array[Dict] | Array of dict, dict contains all properties from user that we need
90
91      """
92      clients = db.session.query(User.id, User.name, User.surname, User.card_id, User.
        profile_pic).distinct(User.id).join(CoachedBy, CoachedBy.client_id==User.id).filter
        (CoachedBy.coach_id==coachId)
93
94
95      result = []
96
97      for client in clients:
98          result.append({
99              'id': client.id,
100             'name' : client.name,
101             'surname' : client.surname,
102             'card_id' : client.card_id,
103             'profile_pic' : str(client.profile_pic),
104             'subscriptionEnd' : get_subscription_end_date(client.id),
105             'lastWorkout' : get_last_workout(client.id),
106             'isActive' : is_active(client.id)
107         })
108
109
110     return result
111
112 def get_client_details(clientId):
```

```
    """
    Get client details with his id

    Parameter(s):
     NAME     |  TYPE  | DESC
     clientId |  INT   | the id of the client

     Return :
     | Dict | Dict, contains all properties (details) from user that we need
    """

    client = db.session.query(User.id, User.name, User.surname, User.email, User.
    birthdate, User.profile_pic, User.created_at, User.card_id, User.address, User.city
    , User.country, User.npa).filter(User.id==clientId).first()


    return client


def get_all_workout_types():
    """
    Get all types of workouts

    Parameter(s):
    NAME     |  TYPE  | DESC
    /        |  /     |  /

    Return :
    | Array[Query()] | Array of query object with all properties selected
    """

    types = db.session.query(WorkoutType.id, WorkoutType.title).order_by(WorkoutType.
    title.asc())
    return types


def get_sessions(coachId):
    """
    Get all sessions as event and create array to be used with FullCalendar.io

    Parameter(s):
     NAME     |  TYPE  | DESC
     coachId   |  INT   | the id of the coach

    Return :
    | Array[Obj()] | Array of object contains properties required to be an event with
    FullCalendar (title, start, end)
    """

    sessions = db.session.query(User.name, User.surname, Session.start_time, Session.
    end_time).join(User, Session.client_id==User.id).filter(Session.coach_id==coachId)

    result = []

    for session in sessions:
        result.append(
            {
                'title': session.name + " " + session.surname,
                'start' : session.start_time.strftime("%Y-%m-%dT%H:%M:%S"),
                'end' : session.end_time.strftime("%Y-%m-%dT%H:%M:%S")
            })

    return result

def get_all_subscriptions():
    """
    Get all types of subscriptions available

    Parameter(s):
     NAME     |  TYPE  | DESC
        /     |   /    | /

```

```
    Return :
    | Array[Query()] | Array of query object with all subscription properties selected
    """

    subscription = db.session.query(Subscription.id, Subscription.title, Subscription.
    cost, Subscription.duration).order_by(Subscription.title.asc())

    return subscription


def get_subscription_id(duration):
    """
    Get the id of a subscription by his duration

    Parameter(s):
     NAME     |  TYPE  | DESC
     duration | STRING | the duration of the subscription

    Return :
    | Query | Array of object contains properties required to be an event with
    FullCalendar (title, start, end)
    """
    id = db.session.query(Subscription.id).filter(Subscription.duration==duration).
    first()

    return id


def get_program(clientId):
    """
    Get the actual program of a client

    Parameter(s):
     NAME     |  TYPE  | DESC
     clientId |   INT  | the id of the client

    Return :
    | Array[Query()] | Array with 2 Query object 1 represent the latest workout program
     and the other the latest diet program.
    """
    # Get the latest workout program uploaded for this user (Program type : 1 => Diet |
     2 => Workout)
    workoutProgram = db.session.query(Program.id, Program.date, User.name, User.surname
    ).join(User, Program.coach_id==User.id).filter(Program.type=="Workout").filter(
    Program.client_id==clientId).order_by(Program.date.desc()).first()
    dietProgram = db.session.query(Program.id, Program.date, User.name, User.surname).
    join(User, Program.coach_id==User.id).filter(Program.type=="Diet").filter(Program.
    client_id==clientId).order_by(Program.date.desc()).first()

    result = []

    result.append(workoutProgram)
    result.append(dietProgram)

    return result


def get_program_by_id(programId):
    """
    Get a specific program

    Parameter(s):
     NAME     |  TYPE  | DESC
     programId |   INT  | the id of the program

    Return :
    | Program | Program object with the selected properties
    """
    program = db.session.query(Program.date, Program.type, Program.pdf).filter(Program.
    id==programId).first()
    return program

```

```
def update_card_id(clientId, cardId):
    """
    Update the card id of a client

    Parameter(s):
     NAME     |  TYPE  | DESC
     clientId |   INT  | the id of the client
     cardId   |  INT   | the new card id

    Return :
    | Boolean | True if the update success, else False
    """
    try:
        db.session.query(User).filter(User.id==clientId).update({'card_id':cardId})
        db.session.commit()
        return True
    except:
        return False

def cancel_last_subscription(clientId):
    """
    Cancel the last subscription purchased for a user

    Parameter(s):
     NAME     |  TYPE  | DESC
     clientId |   INT  | the id of the client

    Return :
    | Boolean | True if the delete success, else False
    """
    try:
        #Set the last subscription
        last_subscription = db.session.query(Purchase).filter(Purchase.client_id==
    clientId).order_by(Purchase.date.desc()).first()
        db.session.query(Purchase).filter(Purchase.id==last_subscription.id).delete()

        db.session.commit()
        return True
    except:
        return False
```

```
"""
Author  :       Thomas Fujise
Date    :       08.04.2022
File    :       __init__.py
Version :       1.0.0
Brief   :       Client blueprint initialization
"""

from flask import Blueprint

blueprint = Blueprint(
    'client_blueprint',
    __name__,
    url_prefix=''
)
```

```
"""
Author  :       Thomas Fujise
Date    :       12.04.2022
File    :       forms.py
Version :       1.0.0
Brief   :       Client forms
"""

from flask_wtf import FlaskForm
from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
    BooleanField, ValidationError, SelectField, TextAreaField, HiddenField,
    PasswordField
from wtforms.validators import Email, DataRequired, EqualTo, Length
from wtforms.fields.html5 import DateField, EmailField, IntegerRangeField

```

```
14
15 class UpdateForm(FlaskForm):
16     """
17     Create a update form
18     """
19
20     name = StringField('Name', id='name_update', validators=[DataRequired()])
21
22     surname = StringField('Surname', id='surname_update', validators=[DataRequired()])
23
24     email = EmailField('Email', id="email_update", validators=[DataRequired(), Email()
       ])
25
26     birthdate = DateField('Birthdate', id="birthdate_update", validators=[DataRequired
       ()])
27
28     profile_pic = FileField('Profile Pic', id="profile_pic_update")
29
30     card_id = StringField('Card ID', id='card_update')
31
32     weight = StringField('Weight', id='weight_update')
33
34     height = StringField('Height', id='height_update')
35
36     subscriptionEnd = DateField('Subscription Until', id='subscription_until')
37
38     address = StringField('Address', id='address_update')
39
40     country = StringField('Country', id='country_update')
41
42     city = StringField('City', id='city_update')
43
44     npa = StringField('NPA', id='npa_update')
45
46     update = SubmitField("Update")
47
48
49 class AddReviewForm(FlaskForm):
50     """
51     Create form to add review
52     """
53
54     field1 = IntegerRangeField('Field1', id='field1', default=0)
55
56     field2 = IntegerRangeField('Field2', id='field2', default=0)
57
58     field3 = IntegerRangeField('Field3', id='field3', default=0)
59
60     field4 = IntegerRangeField('Field4', id='field4', default=0)
61
62     comment = TextAreaField('Comment', id='comment')
63
64     type = HiddenField('Type', id='type')
65
66     target = HiddenField('Target', id='target')
67
68     submit = SubmitField("Submit")
69
70 class ChangePasswordForm(FlaskForm):
71     """
72     Create form to change password
73     """
74
75     oldPassword = PasswordField('Old Password', id='oldPassword')
76
77     newPassword = PasswordField('New Password', id='newPassword')
78
79     confirmPassword = PasswordField('Confirm New Password', id='confirmPassword')
80
81     change = SubmitField("Change")
```

```
1 """
```

```
Author  :       Thomas Fujise
Date    :       11.04.2022
File    :       routes.py
Version :       1.0.0
Brief   :       Set all the client routes
"""

# APP
from apps.client import blueprint
from apps import db, login_manager
from apps.authentication.models import User, PhysicalInfo, Subscription, CoachingReview, WorkoutReview, Review, Workout, WorkoutType, Session
from apps.client.forms import UpdateForm, AddReviewForm, ChangePasswordForm
from apps.config import Config
from apps.authentication.util import verify_pass, hash_pass

# FLASK
from flask import render_template, redirect, request, url_for, flash
from flask_login import login_required
from jinja2 import TemplateNotFound

from flask_login import (
    current_user
)


# UTILS
import uuid as uuid
import os
from werkzeug.utils import secure_filename
from datetime import date, datetime

from apps.client.util import *
from apps.coach.util import get_program

@blueprint.route('/index')
@login_required
def index():
  nextSessions = get_next_session(current_user.id)
  return render_template('client/index.html', segment='index', nextSessions=nextSessions)

# Create Profile Page
@blueprint.route('/profile', methods=['GET', 'POST'])
@login_required
def profile():
  update_form = UpdateForm(request.form)
  # Get the last physical information of the user
  physicalInfo = get_physical_infos(current_user.id)

  # Get the reviews posted by user
  reviews = get_reviews(current_user.id)

  # Get each workout type and how many client have made in 2 separate array to use them in js
  wrktTypeList = get_workout_type_count(current_user.id)[0]

  wrktTypeCount = get_workout_type_count(current_user.id)[1] if get_workout_type_count(current_user.id)[1] != [] else 0

  # Get the number of workout per month during this year
  nbWorkoutPerMonth = get_workout_count_per_month(current_user.id)

  # Get the average of heart rate during this week
  avgHeartRate = get_average_heart_rate_last_week(current_user.id)

  # Get the average of calories burned this week
  avgCalories = get_average_calories_last_week(current_user.id)
  # Get the total time training this week
  totalTime = get_time_working_out_last_week(current_user.id)

  # Get the average weight recorded for each month
```

```
70    weightUpdate = get_weight_update(current_user.id)
71
72    #Get the latest programs
73    programs = get_program(current_user.id)
74
75    # Get Coaching review fields name
76    coachingReviewFields = get_coaching_review_field()
77
78    # Get coach id
79    coachId =get_actual_coach_id(current_user.id)
80
81    #Get all the check up made for this user
82    current_user.checkUps = get_all_check_up(current_user.id)
83
84
85    #set the programs
86    current_user.workoutProgram = programs[0]
87    current_user.dietProgram = programs[1]
88
89    # Set the average heartRate for all workout made this week
90    current_user.avgHeartRate = avgHeartRate
91
92    # Set the average heartRate for all workout made this week
93    current_user.avgCalories = avgCalories
94
95    # Set the total time of working out this week
96    current_user.totalTime = totalTime
97
98    # Set the number of workout per month
99    current_user.nbWorkoutPerMonth = nbWorkoutPerMonth
100
101   # Set the average weight per month
102   current_user.weightUpdate = weightUpdate
103
104   # Set the 2 array for workout type
105   current_user.workoutTypeList = wrktTypeList
106   current_user.workoutTypeCount = wrktTypeCount
107
108   #Set all the reviews
109   current_user.reviews = reviews
110
111   # Set the physical value to the user
112   current_user.physicalInfo = physicalInfo
113
114   # Get last subscription end date
115   current_user.subscriptionEnd = get_subscription_end_date(current_user.id)
116
117   if request.method == "POST":
118     current_user.name = request.form['name']
119     current_user.surname = request.form['surname']
120     current_user.email = request.form['email']
121     current_user.birthdate = request.form['birthdate']
122     current_user.address = request.form['address']
123     current_user.city = request.form['city']
124     current_user.country = request.form['country']
125     current_user.npa = request.form['npa']
126
127     # Check if new profile pic
128     if request.files['profile_pic']:
129       current_user.profile_pic = request.files['profile_pic']
130       # Grab Image name
131       pic_filename = secure_filename(current_user.profile_pic.filename)
132       pic_name = str(uuid.uuid1()) + "_" + pic_filename
133
134       saver = request.files['profile_pic']
135
136       # Change it to a string to save to db
137       current_user.profile_pic = pic_name
138
139       try:
140         db.session.commit()
141         saver.save(os.path.join(Config.UPLOAD_FOLDER, pic_name))
```

```
142         flash("Account Updated successfully !", 'success')
143         return render_template("client/profile.html",
144             form=update_form,
145             reviewFields=coachingReviewFields,
146             reviewTargetId=coachId
147         )
148       except:
149         flash("Error! Looks like there was a problem.. try again!", 'danger')
150         return render_template("client/profile.html",
151             form=update_form,
152             reviewFields=coachingReviewFields,
153             reviewTargetId=coachId
154             )
155     else:
156       db.session.commit()
157       flash("User Updated successfully !", 'success')
158       return render_template("client/profile.html",
159           form=update_form,
160           reviewFields=coachingReviewFields,
161           reviewTargetId=coachId
162           )
163   else:
164     return render_template("client/profile.html", segment='profile',
165         form=update_form,
166         id = id,
167         physicalInfo=physicalInfo,
168         reviewFields=coachingReviewFields,
169         reviewTargetId=coachId)
170
171   return render_template("client/profile.html", segment='profile')
172
173
174 @blueprint.route('/review/', methods=['GET'])
175 @login_required
176 def review():
177   # Put all parameters into dict
178   reviewDetails = request.args.to_dict()
179   review = get_review_details(reviewDetails['Id'], reviewDetails['Type'])[0]
180   # Client who post the review
181   client = get_review_author(review.id_client)
182
183   # Target of the review may be workout or a coach depends on review type
184   target = get_review_details(reviewDetails['Id'], reviewDetails['Type'])[1]
185
186   return render_template('client/review.html', segment='review', review=review, client=
      client, target=target)
187
188 @blueprint.route('/checkup/', methods=['GET'])
189 @login_required
190 def checkup():
191   checkUpId = request.args.get('checkUpId')
192   #Get the check up values
193   checkUp = get_check_up(checkUpId)
194
195   return render_template('client/checkup.html', segment='checkup', checkUp=checkUp)
196
197 # Create Workouts Page
198 @blueprint.route('/workouts')
199 @login_required
200 def workouts():
201   # Get all data from workout
202   workouts = get_workouts(current_user.id)
203   return render_template('client/workouts.html', segment='workouts', workouts=workouts)
204
205
206 @blueprint.route('/workout')
207 @login_required
208 def workout():
209   # put all parameters into dict
210   params = request.args.to_dict()
211
212   workoutDetails = get_workout_details(params['Id'])
```

```
213
214   workoutReview = get_workout_review_field()
215
216   return render_template('client/workout.html', segment="workout", workoutDetails=
        workoutDetails, workoutReview=workoutReview)
217
218 @blueprint.route('/add_review', methods=['POST', 'GET'])
219 @login_required
220 def add_review():
221   add_review_form = AddReviewForm(request.form)
222
223   # put all parameters into dict
224   reviewFields = request.args.to_dict()
225   print(reviewFields)
226   if request.method == 'POST':
227     newReview = Review(comment=request.form['comment'], date=datetime.now(),type=
        request.form['type'],id_client=current_user.id)
228     db.session.add(newReview)
229     db.session.flush()
230     if request.form['type'] == "WORKOUT":
231       exists = is_workout_reviewed(request.form['target'])
232       if exists :
233         db.session.rollback()
234         flash("You already add a review on this workout", 'warning')
235         return redirect( url_for('client_blueprint.workouts') )
236       else :
237         # Try to add and commit the workout review
238         try:
239           newWorkoutReview = WorkoutReview(id= newReview.id,difficulty=request.form['
        field1'], feel=request.form['field2'], fatigue=request.form['field3'], energy=
        request.form['field4'], target_id=request.form['target'])
240           db.session.add(newWorkoutReview)
241           db.session.commit()
242           flash("Your workout review is added", 'success')
243         except :
244           db.session.rollback()
245           flash("Error, please try again", 'danger')
246
247         return redirect( url_for('client_blueprint.workouts') )
248
249     elif request.form['type'] == "COACHING" :
250
251       # Try to add and commit the coaching review
252       try:
253         newCoachingReview = CoachingReview(id=newReview.id,satisfaction=request.form['
        field1'], support=request.form['field2'], disponibility=request.form['field3'],
        advice=request.form['field4'], target_id=request.form['target'])
254         db.session.add(newCoachingReview)
255         db.session.commit()
256         flash("Your coaching review is added", 'success')
257       except:
258         flash("Error, please try again", 'danger')
259
260
261       return redirect( url_for('client_blueprint.profile') )
262
263
264   return render_template('client/add_review.html', segment="add_review", form=
        add_review_form, reviewFields=reviewFields)
265
266
267 @blueprint.route('/change_password', methods=['POST', 'GET'])
268 @login_required
269 def change_password():
270   form = ChangePasswordForm(request.form)
271
272   if request.method == 'POST':
273     # Check if the old password is correct
274     if verify_pass(request.form['oldPassword'], current_user.password):
275       # Check if the confirmation field match the new password
276       if request.form['newPassword'] == request.form['confirmPassword'] and request.
        form['newPassword'] != None:
```

```
277
278         try:
279           # Set the new password
280           current_user.password = hash_pass(request.form['newPassword'])
281           db.session.flush()
282           db.session.commit()
283           flash("Password updated", 'success')
284           return redirect( url_for('client_blueprint.profile') )
285         except:
286           flash("Error, please try again", 'danger')
287       else :
288         flash ("Passwords must match !", 'warning')
289     else:
290       flash("Old password isn't correct !", 'warning')
291   return render_template('client/change_password.html', segment="change_password", form
      =form)
292
293
294 # Extract current page name from request
295 def get_segment(request):
296
297     try:
298
299         segment = request.path.split('/')[-1]
300
301         if segment == '':
302             segment = 'index'
303
304         return segment
305     except:
306         return None
```

```
1  """
2  Author  :        Thomas Fujise
3  Date    :        18.05.2022
4  File    :        util.py
5  Version :        1.0.0
6  Brief   :        All the functions needed to get datas for client templates
7  """
8
9  # APP
10 from apps import db
11 from apps.authentication.models import User, PhysicalInfo, Subscription, Purchase,
       CoachingReview, WorkoutReview, Review, Workout, WorkoutType, Session, CoachedBy
12
13 # SQL ALCHEMY
14 from sqlalchemy import union, func
15
16 # UTILS
17 from dateutil.relativedelta import relativedelta
18 import numpy as np
19 from datetime import date, datetime
20
21 def get_next_session(userId):
22     """
23     Get all the next sessions
24
25     SQL :
26    SELECT SESSION.start_time, SESSION.end_time, SESSION.duration, WORKOUT_TYPE.title,
       USER.name, USER.surname
27    FROM SESSION
28    JOIN WORKOUT_TYPE ON SESSION.workout_type = WORKOUT_TYPE.id
29    JOIN USER ON USER.id = SESSION.coach_id
30    WHERE SESSION.client_id = 1 AND SESSION.start_time > curdate()
31    ORDER BY SESSION.start_time
32
33     Parameter(s) :
34     NAME      |  TYPE   | DESC
35     userId    | INT     | The id of the user
36
37     Return :
38     | ARRAY[] | Array with all the next sessions planed for a user
```

```
    """
    sessions = db.session.query(Session.start_time, Session.end_time, Session.duration,
     WorkoutType.title, WorkoutType.logo, User.name, User.surname).join(WorkoutType,
    Session.workout_type==WorkoutType.id).join(User, Session.coach_id==User.id).filter(
    Session.client_id==userId).filter(Session.start_time>datetime.now()).order_by(
    Session.start_time)

    return sessions

def get_review_author(clientId):
    """
    Get the client name and surname who add the review with his id

     Parameter(s):
     NAME     |  TYPE  | DESC
     clientId |  INT   | the id of the client who post the review.

     Return :
     | USER() | User name and surname
    """
    result = db.session.query(User.name, User.surname).filter(User.id==clientId).first
    ()

    return result


def get_review_details(reviewId, reviewType):
    """
    Get the details of a review depends on his type

    Parameter(s) :
     NAME      |  TYPE  | DESC
     reviewId  |  INT   | The id of the review
     reviewType| STRING | The type of the review ("WORKOUT" or "COACHING")

    Return :
    | ARRAY[REVIEW(), ARRAY[]] | Array with the review details and an array that
    contains the target details of the review
    """
    #Queries to get all Review from user
  #coachingReviewQuery = db.session.query(CoachingReview.id, CoachingReview.
    satisfaction.label("Field1"), CoachingReview.support.label("Field2"),
    CoachingReview.disponibility.label("Field3"), CoachingReview.advice.label("Field4")
    , CoachingReview.target_id.label("target_id"))
  #workoutReviewQuery = db.session.query(WorkoutReview.id, WorkoutReview.difficulty,
    WorkoutReview.feel, WorkoutReview.fatigue, WorkoutReview.energy, WorkoutReview.
    target_id)

  #UNION with the 2 queries
  #reviewsUnion = union(coachingReviewQuery,workoutReviewQuery).alias()

  # Query to get field from Review table and from the review union made before
  #reviewsQuery = db.session.query(Review.id, Review.comment, Review.date, Review.type,
     reviewsUnion.c.Field1, reviewsUnion.c.Field2, reviewsUnion.c.Field3, reviewsUnion.
    c.Field4, Review.id_client, reviewsUnion.c.target_id).select_from(reviewsUnion).
    join(Review,Review.id==reviewsUnion.c.COACHING_REVIEW_id).filter(Review.id_client==
    id).order_by(Review.date.desc())
    target = {}
    if reviewType == "WORKOUT":
        review = db.session.query(Review.id, Review.comment, Review.date, Review.type,
    WorkoutReview.difficulty.label("Field1"), WorkoutReview.feel.label("Field2"),
         WorkoutReview.fatigue.label("Field3"), WorkoutReview.energy.label("Field4"),
          Review.id_client, WorkoutReview.target_id).join(Review, Review.id==
    WorkoutReview.id).filter(Review.id==reviewId).first()

        targetQuery = db.session.query(WorkoutType.title, Workout.date, Workout.
    duration).join(WorkoutType, WorkoutType.id==Workout.workout_type).filter(Workout.id
    ==review.target_id).first()

        target['Type'] = targetQuery.title
        target['Date'] = targetQuery.date
        target['Duration'] = targetQuery.duration
```

```
92
93      elif reviewType == "COACHING":
94          review = db.session.query(Review.id, Review.comment, Review.date, Review.type,
        CoachingReview.satisfaction.label("Field1"), CoachingReview.support.label("Field2")
        , CoachingReview.disponibility.label("Field3"), CoachingReview.advice.label("Field4
        "), Review.id_client, CoachingReview.target_id).join(Review, Review.id==
        CoachingReview.id).filter(Review.id==reviewId).first()
95
96          targetQuery = db.session.query(User.name, User.surname).filter(User.id==review.
        target_id).first()
97          target['Name'] = targetQuery.name
98          target['Surname'] = targetQuery.surname
99
100     return [review, target]
101
102 def get_reviews(clientId):
103     """
104     Get all the reviews added by user
105
106     Parameter(s):
107      NAME      |  TYPE   | DESC
108      clientId  |  INT    | the id of the client
109
110     | ARRAY[REVIEW()] | Array with all reviews added by the user
111     """
112
113     reviews = db.session.query(Review.id, Review.type, Review.date).filter(Review.
        id_client==clientId).order_by(Review.date.desc())
114
115     return reviews
116
117 def get_workouts(clientId):
118     """
119     Get all the workouts done by a client
120
121     Parameter(s) :
122      NAME       |  TYPE   | DESC
123      clientId   |  INT    | The id of the client
124
125     Return :
126     | ARRAY[WORKOUT()] | Array with all workouts made by user
127     """
128     workouts = db.session.query(WorkoutType.title,Workout.id, Workout.date, Workout.
        duration, Workout.heart_rate_avg, Workout.calories).join(WorkoutType, Workout.
        workout_type == WorkoutType.id).filter(Workout.client_id==clientId).order_by(
        Workout.date.desc())
129
130     return workouts
131
132 def get_workout_details(workoutId):
133     """
134     Get all the details of a workout
135
136     Parameter(s):
137     NAME       |  TYPE   | DESC
138     workoutId  |  INT    | The id of the workout
139
140     | OBJECT | Object with all details of the workout as properties
141     """
142
143     workout =  db.session.query(WorkoutType.title, WorkoutType.logo, Workout.id,
        Workout.date, Workout.duration, Workout.heart_rate_max, Workout.heart_rate_min,
144         Workout.heart_rate_avg, Workout.calories, Workout.active_calories, Workout.
        distance, Workout.pace_avg).join(WorkoutType, Workout.workout_type == WorkoutType.
        id).filter(Workout.id==workoutId).first()
145
146     return workout
147
148 def get_workout_review_field():
149     """
150     Get the fields name for a workout review (the fields names are set in the database)
151
```

```
    Parameter(s):
    /

    Return :
    | ARRAY[STRING] | Array with the workout review fields name
    """
    fields = db.session.query(WorkoutReview).statement.columns.keys()

    return fields

def get_coaching_review_field():
    """
    Get the fields name for a coaching review (the fields names are set in the database
    )

    Parameter(s):
    /

    Return :
    | ARRAY[STRING] | Array with the coaching review fields name
    """
    fields = db.session.query(CoachingReview).statement.columns.keys()
    return fields

def is_workout_reviewed(workoutId):
    """
    Check if a workout is already reviewed


    Parameter(s):
    NAME      |  TYPE  | DESC
    workoutId  |  INT   | The id of the workout

    Return :
    | BOOLEAN | True if the workout is already reviewed, else False
    """
    result = db.session.query(WorkoutReview.target_id).filter(WorkoutReview.target_id==
    workoutId).first() is not None
    return result

def get_user(userId):
    """
    Get user object with id

    Parameter(s):
    NAME      |  TYPE  | DESC
    userId    |  INT   | The id of the user

    Return :
    | USER() | the user
    """
    user = User.query.filter_by(id=userId).first()

    return user

def get_physical_infos(userId):
    """
    Get the last added physical infos for a user

    Parameter(s):
    NAME      |  TYPE  | DESC
    userId    |  INT   | The id of the user

    Return :
    | PHYSICALINFO() | the last updated physical infos of the user
    """
    infos = PhysicalInfo.query.filter_by(user_id=userId).order_by(PhysicalInfo.date.
    desc()).first()

    return infos

def get_subscription_end_date(userId):
```

```
    """
    Get the last subsciption purchased end date

    Parameter(s):
    NAME      |  TYPE  | DESC
    userId    |  INT   | The id of the user

    Request in SQL :
     SELECT `SUBSCRIPTION`.DURATION AS `SUBSCRIPTION_DURATION`, `PURCHASE`.DATE AS `
    PURCHASE_DATE`
   FROM `PURCHASE`
   INNER JOIN `SUBSCRIPTION` ON `PURCHASE`.SUBSCRIPTION_ID = `SUBSCRIPTION`.ID
   WHERE `PURCHASE`.CLIENT_ID = userId
   ORDER BY `PURCHASE`.DATE

    Return :
    | DATE | the end date of the last purchased subscription

    """
    subscription = db.session.query(Subscription.duration, Purchase.date).join(
    Subscription, Purchase.subscription_id==Subscription.id).filter(Purchase.client_id
    ==userId).order_by(Purchase.date.desc()).first()

    endDate=''
    if subscription != None:
        #Add duration to the purchase date to get end date
        endDate = (subscription[1] + relativedelta(months=subscription[0]))


    return endDate


def get_workout_type_count(clientId):
    """
    Get the count of each type of workout made by user and split workout type count in
    2 array (1 for the title and 1 for the count) to use them
    in javascript with Chart.JS

    Parameter(s) :
     NAME      |  TYPE  | DESC
     clientId  |  INT   | The id of the client

    Return :
    | ARRAY[ARRAY[STRING], ARRAY[INT]] | Array with 2 differents array inside. 1 for
    all the titles and 1 for all the counts
    """
    workoutTypeCount = db.session.query(WorkoutType.title, func.count(Workout.
    workout_type).label("count")).join(Workout, Workout.workout_type==WorkoutType.id).
    filter(Workout.client_id==clientId).group_by(Workout.workout_type)

    wrktTypeList = []
    wrktTypeCount = []

    for wtCount in workoutTypeCount:
        wrktTypeList.append('"' +wtCount.title+ '"')
        wrktTypeCount.append(wtCount.count)

    return [wrktTypeList, wrktTypeCount]

def get_workout_count_per_month(clientId):
    """
    Get the number of workout made each month during this year

    Parameter(s) :
     NAME      |  TYPE  | DESC
     clientId  |  INT   | The id of the client

    Return :
    | ARRAY[INT] | Array with 12 int values represent the 12 months of a year. (Each
    values is the number of workout made for the month)
    """

```

```
285     count = db.session.query(func.month(Workout.date).label("month"), func.count(
        Workout.id).label("count")).filter(Workout.client_id==clientId).filter(func.year(
        date.today())==func.year(Workout.date)).group_by(func.month(Workout.date))
286 
287     # Create a year array with 12 values (each value represent a month)
288     nbWorkoutPerMonth = [0,0,0,0,0,0,0,0,0,0,0,0]
289     for i in range(12):
290         for wtCountbyMonth in count:
291             if i == wtCountbyMonth.month-1: # If month got result from query set the
        value else keep 0
292                 nbWorkoutPerMonth[i] = wtCountbyMonth.count
293                 break
294 
295 
296     return nbWorkoutPerMonth
297 
298 def get_actual_coach_id(clientId):
299     """
300     Get the actual coach id for a client
301 
302     Parameter(s) :
303      NAME       |  TYPE   | DESC
304      clientId   |  INT    | The id of the client
305 
306     Return :
307     | INT | The coach id
308     """
309 
310     coachId = db.session.query(CoachedBy.coach_id).filter(CoachedBy.client_id==clientId
        ).order_by(CoachedBy.end_date.desc()).first()
311 
312     return coachId
313 
314 def get_average_calories_last_week(clientId):
315     """
316     Get the average of calories burned this week
317 
318     SQL :
319    SELECT AVG(calories) FROM WORKOUT WHERE WORKOUT.client_id = clientId AND WEEK(
        WORKOUT.date) = WEEK(CURDATE()) - 1;
320 
321     Parameter(s) :
322      NAME       |  TYPE   | DESC
323      clientId   |  INT    | The id of the client
324 
325     Return :
326     | DECIMAL | the average of calories burned rounded to 1 decimal
327     """
328     calories = db.session.query(func.avg(Workout.calories).label("avg")).filter(Workout
        .client_id==clientId).filter(func.week(Workout.date)>=func.week(date.today()) -1).
        first()
329 
330     return round(calories.avg,1) if calories.avg != None else 0.0
331 
332 def get_average_heart_rate_last_week(clientId):
333     """
334     Get the average heart rate recorded this week
335 
336     SQL :
337      SELECT AVG(heart_rate_avg) FROM WORKOUT WHERE WORKOUT.client_id = clientId AND
        WEEK(WORKOUT.date) = WEEK(CURDATE()) - 1;
338 
339     Parameter(s) :
340      NAME       |  TYPE   | DESC
341      clientId   |  INT    | The id of the client
342 
343     Return :
344     | DECIMAL | the average of heart rate recorded rounded to 1 decimal
345     """
346     heart_rate = db.session.query(func.avg(Workout.heart_rate_avg).label("avg")).filter
        (Workout.client_id==clientId).filter(func.week(Workout.date)>=func.week(date.today
        ()) -1).first()
```

```
347
348     return round(heart_rate.avg,1) if heart_rate.avg != None else 0.0
349
350 def get_time_working_out_last_week(clientId):
351     """
352     Get the total time working out recorded this week
353
354     Parameter(s) :
355      NAME       |  TYPE   | DESC
356      clientId   |  INT    | The id of the client
357
358     Return :
359     | DECIMAL | the total time in minute
360     """
361
362     subquery = db.session.query((func.extract("hour",Workout.duration)*60*60 + func.
        extract("minute",Workout.duration)*60+ func.extract("second",Workout.duration)).
        label("time")).filter(Workout.client_id==clientId).filter(func.week(Workout.date)>
        func.week(date.today())-1)
363     result = 0.0
364
365     for workout in subquery:
366         result += workout.time
367
368
369     return round(result /60,1)
370
371 def get_weight_update(clientId):
372     """
373     Get the average weight each month of a client during the whole year (months without
         values will show 0)
374
375     Parameter(s) :
376      NAME       |  TYPE   | DESC
377      clientId   |  INT    | The id of the client
378
379     Return :
380     | ARRAY[INT] | Array with 12 int values represent the 12 months of a year. (Each
        values is the average of all the weight recorded for the month in kg)
381     """
382
383     weights = db.session.query(func.month(PhysicalInfo.date).label("month"), func.avg(
        PhysicalInfo.weight).label("avg")).filter(PhysicalInfo.user_id==clientId).filter(
        func.year(date.today())==func.year(PhysicalInfo.date)).group_by(func.month(
        PhysicalInfo.date))
384
385     # Create a year array with 12 values (each value represent a month)
386     weightPerMonth = [0,0,0,0,0,0,0,0,0,0,0,0]
387     for i in range(12):
388         for weight in weights:
389             if i == weight.month-1: # If month got result from query set the value else
         keep 0
390                 weightPerMonth[i] = float(round(weight.avg,1))
391                 break
392
393     return weightPerMonth
394
395 def get_all_check_up(clientId):
396     """
397     Get all the check ups made for a client
398
399     Parameter(s) :
400      NAME       |  TYPE   | DESC
401      clientId   |  INT    | The id of the client
402
403     Return :
404     | ARRAY(QUERY()) | Array contains query object with field selected as properties (
        query object as PhysicalInfo)
405     """
406     checkUp = db.session.query(PhysicalInfo.id, PhysicalInfo.date, PhysicalInfo.weight)
        .filter(PhysicalInfo.user_id==clientId).order_by(PhysicalInfo.date.desc())
407     return checkUp
```

```
408
409 def get_check_up(checkUpId):
410     """
411     Get all the values of a check up
412
413     Parameter(s) :
414      NAME       |  TYPE  | DESC
415      checkUpId  |  INT   | The id of the checkup
416
417     Return :
418     | PhysicalInfo() | PhysicalInfo object contains all value of the check up selected
419     """
420     checkUp = db.session.query(PhysicalInfo).filter(PhysicalInfo.id==checkUpId).first()
421     return checkUp
```

```
1  #!/usr/bin/env python
2
3  """
4  Author  :        Thomas Fujise
5  Date    :        02.06.2022
6  File    :        accessink.py
7  Version :        1.0.0
8  Brief   :        Polar accesslink class to made all request and get the data
9  """
10
11 from __future__ import print_function
12
13 from utils import load_config, save_config, pretty_print_json
14 from accesslink import AccessLink
15 from db import cursor, db
16 from datetime import datetime
17 from config import WORKOUT_TYPE
18
19 try:
20     input = raw_input
21 except NameError:
22     pass
23
24
25 CONFIG_FILENAME = "config.yml"
26
27
28 class PolarAccessLink(object):
29     """Polar Open AccessLink v3."""
30
31     def __init__(self):
32         self.config = load_config(CONFIG_FILENAME)
33
34         if "access_token" not in self.config:
35             print("Authorization is required. Run authorization.py first.")
36             return
37
38         self.accesslink = AccessLink(client_id=self.config["client_id"],
39                                      client_secret=self.config["client_secret"])
40
41         self.running = True
42         #self.get_exercises()
43
44
45
46     def show_menu(self):
47         while self.running:
48             print("\nChoose an option:\n" +
49                   "-----------------------\n" +
50                   "1) Get user information\n" +
51                   "2) Check available data\n" +
52                   "3) Revoke access token\n" +
53                   "4) Exit\n" +
54                   "-----------------------")
55             self.get_menu_choice()
56
57     def get_menu_choice(self):
```

```
58          choice = input("> ")
59          {
60              "1": self.get_user_information,
61              "2": self.check_available_data,
62              "3": self.revoke_access_token,
63              "4": self.exit
64          }.get(choice, self.get_menu_choice)()
65
66      def get_user_information(self):
67          user_info = self.accesslink.users.get_information(user_id=self.config["user_id"],
68                                                            access_token=self.config["access_token"])
69          pretty_print_json(user_info)
70
71      def check_available_data(self):
72          available_data = self.accesslink.pull_notifications.list()
73
74          if not available_data:
75              print("No new data available.")
76              return false
77
78          print("Available data:")
79          pretty_print_json(available_data)
80
81          for item in available_data["available-user-data"]:
82              if item["data-type"] == "EXERCISE":
83                  self.get_exercises()
84              elif item["data-type"] == "ACTIVITY_SUMMARY":
85                  self.get_daily_activity()
86              elif item["data-type"] == "PHYSICAL_INFORMATION":
87                  self.get_physical_info()
88
89      def revoke_access_token(self):
90          self.accesslink.users.delete(user_id=self.config["user_id"],
91                                       access_token=self.config["access_token"])
92
93          del self.config["access_token"]
94          del self.config["user_id"]
95          save_config(self.config, CONFIG_FILENAME)
96
97          print("Access token was successfully revoked.")
98
99          self.exit()
100
101     def exit(self):
102         self.running = False
103
104     def get_exercises(self, userId):
105         # Query to insert
106         insertWorkoutQuery = "INSERT INTO WORKOUT (workout_type, client_id, date, duration, heart_rate_max, heart_rate_avg, calories, distance) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
107
108         #Init values
109         values = ()
110
111         workout_type = ''
112         client_id = userId
113         date = ''
114         duration = ''
115         heart_rate_max = ''
116         heart_rate_avg = ''
117         calories = ''
118         distance = 0.0
119
120         transaction = self.accesslink.training_data.create_transaction(user_id=self.config["user_id"],
121                                                                        access_token=self.config["access_token"])
122         if not transaction:
123             print("No new exercises available.")
```

```
124             return
125
126         resource_urls = transaction.list_exercises()["exercises"]
127
128
129         for url in resource_urls:
130             exercise_summary = transaction.get_exercise_summary(url)
131
132             print("Exercise summary:")
133             pretty_print_json(exercise_summary)
134             if WORKOUT_TYPE[exercise_summary['detailed-sport-info']] is not None:
135                 workout_type = WORKOUT_TYPE[exercise_summary['detailed-sport-info']]
136
137             date = exercise_summary["start-time"]
138             start = datetime.strptime(exercise_summary["start-time"], '%Y-%m-%dT%H:%M:%
    S')
139             end = datetime.strptime(exercise_summary["upload-time"], '%Y-%m-%dT%H:%M:%S
    .%fZ')
140
141             #Try different format (depends on duration exemple : PT1H30M)
142             if 'H' in exercise_summary["duration"]:
143                 duration = datetime.strptime(exercise_summary["duration"], 'PT%HH%MM')
144             elif 'M' in exercise_summary["duration"]:
145                 duration = datetime.strptime(exercise_summary["duration"], 'PT%MM%S.%fS
    ')
146             elif 'S' in exercise_summary["duration"]:
147                 duration = datetime.strptime(exercise_summary["duration"], 'PT%S.%fS')
148
149             heart_rate_max = exercise_summary["heart-rate"]["maximum"]
150             heart_rate_avg = exercise_summary["heart-rate"]["average"]
151             calories = exercise_summary["calories"]
152
153             if 'distance' in exercise_summary:
154                 distance = exercise_summary['distance']
155
156             values = (workout_type, client_id, date, duration, heart_rate_max,
    heart_rate_avg, calories, distance)
157
158         print(values)
159         cursor.execute(insertWorkoutQuery, values)
160         db.commit()
161         print(cursor.rowcount, "was inserted.")
162         transaction.commit()
163
164     def get_daily_activity(self):
165         transaction = self.accesslink.daily_activity.create_transaction(user_id=self.
    config["user_id"],
166                                                                         access_token=
    self.config["access_token"])
167         if not transaction:
168             print("No new daily activity available.")
169             return
170
171         resource_urls = transaction.list_activities()["activity-log"]
172
173         for url in resource_urls:
174             activity_summary = transaction.get_activity_summary(url)
175
176             print("Activity summary:")
177             pretty_print_json(activity_summary)
178
179         transaction.commit()
180
181     def get_physical_info(self):
182         transaction = self.accesslink.physical_info.create_transaction(user_id=self.
    config["user_id"],
183                                                                        access_token=
    self.config["access_token"])
184         if not transaction:
185             print("No new physical information available.")
186             return
187
```

```
        resource_urls = transaction.list_physical_infos()["physical-informations"]

        for url in resource_urls:
            physical_info = transaction.get_physical_info(url)

            print("Physical info:")
            pretty_print_json(physical_info)

        transaction.commit()
```

```
#!/usr/bin/env python
# encoding: utf-8
"""
Author  :       Thomas Fujise
Date    :       02.06.2022
File    :       config.py
Version :       1.0.0
Brief   :       Config constants
"""

#DATABASE
DB_HOST = "localhost"
DB_USER = "fitjourney_dba"
DB_PASSWORD = "Super2012$"
DB = "fitjourney"

#WORKOUT TYPE
WORKOUT_TYPE = {
    "STRENGTH_TRAINING": 1,
    "CYCLING" : 2,
    "RUNNING" : 3
}
```

```
#!/usr/bin/env python

"""
Author  :       Thomas Fujise
Date    :       02.06.2022
File    :       db.py
Version :       1.0.0
Brief   :       Database connector
"""

import config
import mysql.connector

db = mysql.connector.connect(
  host=config.DB_HOST,
  user=config.DB_USER,
  password=config.DB_PASSWORD,
  database=config.DB
)

cursor = db.cursor()

def getUserIdByCard(cardId):
    sql = "SELECT id FROM USER WHERE card_id = %s"
    card = (cardId,)

    cursor.execute(sql, card)

    result = cursor.fetchone()

    return result
```

```
"""
Author  :       Thomas Fujise
Date    :       02.06.2022
File    :       main.py
Version :       1.0.0
Brief   :       Python script to check if card scanned are allowed (if they are assign
    to a client) and to save exercise data with Polar API
```

```
"""
from smartcard.CardRequest import CardRequest
from smartcard.Exceptions import CardRequestTimeoutException
from smartcard.CardType import AnyCardType
from smartcard import util
from accesslink_polar import PolarAccessLink
import time

from db import cursor, getUserIdByCard

if __name__ == '__main__':

    #Init Polar class
    polar = PolarAccessLink()

    #Query to get the card id from user
    selectQuery = "SELECT name, surname, card_id FROM USER"


    cursor.execute(selectQuery)

    myresult = cursor.fetchall()
    # respond to the insertion of any type of smart card
    card_type = AnyCardType()

    # create the request. Wait for up to x seconds for a card to be attached
    request = CardRequest(timeout=None, cardType=card_type)

    entry = False
    while True:
                time.sleep(0.1)
                # listen for the card
                service = None
                try:
                    service = request.waitforcard()
                except CardRequestTimeoutException:
                    print("ERROR: No card detected")
                    exit(-1)

                # when a card is attached, open a connection
                conn = service.connection
                conn.connect()

                # get and print the ATR and UID of the card
                get_uid = util.toBytes("FF CA 00 00 00")

                data, sw1, sw2 = conn.transmit(get_uid)

                print(data)
                card_scanned = ' '.join(str(e) for e in data)
                print(card_scanned)
                card_found = False

                for client in myresult:

                    if client[2] == card_scanned:
                        card_found = True
                        if entry:
                            print("-------------GOODBYE " + client[0] + " " + client
    [1]+ "---------------")
                            if polar.check_available_data():
                                polar.get_exercises(getUserIdByCard(client[2])[0])
                            entry = False
                        else:
                            print("-------------WELCOME " + client[0] + " " + client
    [1]+ "---------------")
                            entry = True


                if not card_found:
                    print("-------------UNRECOGNIZED CARD---------------")

```

```
                time.sleep(2)
```

```
<!--
============================================================
Author          :           Thomas Fujise
Date            :           11.04.2022
File            :           base.html
Version         :           1.0.0
Brief           :           HTML base for each page of the application
============================================================
-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"
    >
  <title>
        FitJourney - {% block title %} {% endblock %}
  </title>
  {% include "includes/css.html" %}
  <!-- Specific CSS HERE -->
  {% block stylesheets %}{% endblock stylesheets %}
</head>
<body class="">
  {% include "includes/nav.html" %}
  <div class="main-content" id="panel">
        <!-- Header -->
        <div class="header bg-gradient-purple py-7 py-lg-8">
          <div class="container">
            <div class="header-body text-center mb-7">
              <div class="row justify-content-center">
                <div class="col-lg-5 col-md-6">
                  <h1 class="text-white">
                    <a target="_blank" class="text-white"
                       href="">FitJourney</a>
                  </h1>
                  <p class="text-lead text-light">
                    The application to manage your coaching
                  </p>
                </div>
              </div>
            </div>
          </div>
        </div>
    {% block content %}{% endblock content %}
  </div>
  {% include "includes/scripts.html" %}
  <!-- Specific JS HERE -->
  {% block javascripts %}{% endblock javascripts %}
</body>
</html>
```

```
  <!-- Fonts -->
  <link href="https://fonts.googleapis.com/css?family=Poppins:200,300,400,600,700,800"
    rel="stylesheet">

  <!-- Icons -->
  <link href="/static/assets/vendor/nucleo/css/nucleo.css" rel="stylesheet">

  <!-- Argon CSS -->
  <link type="text/css" href="/static/assets/css/argon.min.css" rel="stylesheet">
  <link rel="stylesheet" href="/static/assets/css/argon.css?v=1.2.0" type="text/css">
```

```
<nav class="sidenav navbar navbar-vertical  fixed-left  navbar-expand-xs navbar-light
    bg-white" id="sidenav-main">
    <div class="scrollbar-inner">
      <!-- Brand -->
      <div class="sidenav-header  align-items-center">
        <a class="navbar-brand" href="/">
            <span class="nav-link-text">FitJourney</span>
        </a>
```

```
      </div>
      <div class="navbar-inner">
        <!-- Collapse -->
        <div class="collapse navbar-collapse" id="sidenav-collapse-main">
          <!-- Nav items -->
          <ul class="navbar-nav">
            {% if not current_user.is_authenticated %}
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
    ="{{ url_for('authentication_blueprint.login') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Login</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
    ="{{ url_for('authentication_blueprint.register') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Register</span>
              </a>
            </li>
            {% elif current_user.is_authenticated and current_user.role == 1 %}
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
    ="{{ url_for('client_blueprint.profile') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Profile</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'index' in segment %} active {% endif %}" href="
    /">
                <i class="ni ni-calendar-grid-58 text-orange"></i>
                <span class="nav-link-text">Next Sessions</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'workouts' in segment %} active {% endif %}"
    href="{{ url_for('client_blueprint.workouts') }}">
                <i class="ni ni-planet text-orange"></i>
                <span class="nav-link-text">Workouts</span>
              </a>
            </li>
            {% elif current_user.is_authenticated and current_user.role == 2 %}
            <li class="nav-item">
              <a class="nav-link {% if 'dashboard' in segment %} active {% endif %}"
    href="{{ url_for('coach_blueprint.dashboard') }}">
                <i class="ni ni-tv-2 text-primary"></i>
                <span class="nav-link-text">Dashboard</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
    ="{{ url_for('client_blueprint.profile') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Profile</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'calendar' in segment %} active {% endif %}"
    href="{{ url_for('coach_blueprint.calendar') }}">
                <i class="ni ni-calendar-grid-58 text-orange"></i>
                <span class="nav-link-text">Calendar</span>
              </a>
            </li>
            {% endif %}
          </ul>
          <!-- Divider -->
          <hr class="my-3">
          {% if current_user.is_authenticated %}
          <!-- Heading -->
          <ul class="navbar-nav""">
```

```
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('authentication_blueprint.logout') }}"
    >
            <i class="ni ni-user-run text-red"></i><i class="fa fa-sign-out" aria-
    hidden="true"></i>
            <span class="nav-link-text">Logout</span>
          </a>
        </li>
        </ul>
        {% endif %}
        </div>
      </div>
    </div>
  </nav>
```

```
<!-- Core -->
<script src="https://code.jquery.com/jquery-3.6.0.slim.js" integrity="sha256-
    HwWONEZrpuoh951cQD1ov2HUK5zA5DwJ1DNUXaM6FsY=" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
    integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
     crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
    integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
     crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.1/chart.min.js"
    integrity="sha512-QSkVNOCYLtj73J4hbmVoOV6KVZuMluZlioC+
    trLpewV8qMjsWqlIQvkn1KGX2StWvPMdWGBqim1xlC8krl1EKQ==" crossorigin="anonymous"
    referrerpolicy="no-referrer"></script>


<!-- Theme JS -->
<script src="/static/assets/js/argon.js?v=1.2.0"></script>
```

```
{% extends 'layout/base.html' %}

{% block title %} Add check up {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

{% endfor %}



    <div class="row justify-content-center">

      <div class="col-xl-8 ">
        <div class="card">
          <div class="card-header">
            <div class="row align-items-center">
              <div class="col-8">
                <h3 class="mb-0">Add check up </h3>
              </div>
              <div class="col-4 text-right">

              </div>
            </div>
          </div>
          <div class="card-body">
            <form action="{{ url_for('coach_blueprint.check_up', clientId=infos.id)}}"
    method="POST">
                {{ form.hidden_tag() }}
```

```
38              <div class="pl-lg-4">
39                <div class="row justify-content-center">
40                    <div class="col-lg-4">
41                        <div class="form-group">
42                          <label class="form-control-label" for="name">Name</label>
43                          {{form.name(placeholder="Name",class="form-control",type="
    text", value=infos.name)}}
44                        </div>
45                    </div>
46                    <div class="col-lg-4">
47                      <div class="form-group">
48                        <label class="form-control-label" for="surname">Surname</
    label>
49                        {{form.surname(placeholder="Surname",class="form-control",
    type="text", value=infos.surname)}}
50                        </div>
51                    </div>
52                    <div class="col-lg-4">
53                      <div class="form-group">
54                        <label class="form-control-label" for="surname">Age</label>
55                        {{form.age(placeholder="Age",class="form-control", value=
    infos.age)}}
56                        </div>
57                    </div>
58
59
60              </div>
61              <hr class="my-4" />
62              <div class="row justify-content-center">
63                <div class="col-lg-3">
64                  <div class="form-group">
65                    <label class="form-control-label" for="weight">Weight (kg)</label
    >
66                    {{form.weight(placeholder="Weight",class="form-control", oninput=
    "this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
67                  </div>
68                </div>
69                <div class="col-lg-3">
70                  <div class="form-group">
71                    <label class="form-control-label" for="height">Height (cm)</label
    >
72                    {{form.height(placeholder="Height",class="form-control", oninput=
    "this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
73                  </div>
74                </div>
75                <div class="col-lg-3">
76                  <div class="form-group">
77                    <label class="form-control-label" for="water">Water %</label>
78                    {{form.water(placeholder="Water %",class="form-control", oninput=
    "this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
79                  </div>
80                </div>
81                <div class="col-lg-3">
82                  <div class="form-group">
83                    <label class="form-control-label" for="protein">Protein %</label>
84                    {{form.protein(placeholder="Protein %",class="form-control",
    oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1')
    ;")}}
85                  </div>
86                </div>
87              </div>
88                  <div class="row justify-content-center">
89                    <div class="col-lg-3">
90                      <div class="form-group">
91                        <label class="form-control-label" for="muscle_mass_percent">
    Muscle Mass %</label>
92                        {{form.muscle_mass_percent(placeholder="Muscle mass %",class=
    "form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace
    (/(\..*)\./g, '$1');")}}
93                        </div>
94                    </div>
95                    <div class="col-lg-3">
```

```
                        <div class="form-group">
                          <label class="form-control-label" for="body_fat_percent">Body fat %</label>
                          {{form.body_fat_percent(placeholder="Body fat %",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="bone_mass_percent">Bone mass %</label>
                          {{form.bone_mass_percent(placeholder="Bone mass %",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="bmr">BMR</label>
                          {{form.bmr(placeholder="BMR",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                    </div>

                    <div class="row justify-content-center">
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="muscle_mass">Muscle Mass (kg)</label>
                          {{form.muscle_mass(placeholder="Muscle mass (kg)",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="body_fat_percent">Body fat (kg)</label>
                          {{form.body_fat(placeholder="Body fat (kg)",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="bone_mass_percent">Visceral fat (kg)</label>
                          {{form.fat_visceral(placeholder="Visceral fat (kg)",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="bmr">Bone mass (kg)</label>
                          {{form.bone_mass(placeholder="Bone mass (kg)",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                    </div>
                    <div class="row justify-content-center">
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="lean_body_mass">Lean body mass (kg)</label>
                          {{form.lean_body_mass(placeholder="Lean body mass (kg)",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
```

```
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="lean_body_mass">Body age</label>
                          {{form.body_age(placeholder="Body age",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                      <div class="col-lg-3">
                        <div class="form-group">
                          <label class="form-control-label" for="bmi">BMI</label>
                          {{form.bmi(placeholder="BMI",class="form-control", oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
                        </div>
                      </div>
                    </div>
                <div class="row">
                  <div class="col-md-12">
                    <div class="form-group">
                        {{form.add(class="form-control btn btn-primary")}}
                    </div>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Add client {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
  </div>

  {% endfor %}



    <div class="row justify-content-center">

      <div class="col-xl-8 ">
        <div class="card">
          <div class="card-header">
            <div class="row align-items-center">
              <div class="col-8">
                <h3 class="mb-0">Add new client </h3>
              </div>
              <div class="col-4 text-right">
```

```
31
32              </div>
33            </div>
34          </div>
35          <div class="card-body">
36            <form action="/add_client" method="POST">
37                {{ form.hidden_tag() }}
38              <div class="pl-lg-4">
39                <div class="row justify-content-center">
40                    <div class="col-lg-6">
41                        <div class="form-group">
42                          <label class="form-control-label" for="name">Name</label>
43                          {{form.name(placeholder="Name",class="form-control",type="
    text")}}
44                        </div>
45                      </div>
46                      <div class="col-lg-6">
47                        <div class="form-group">
48                          <label class="form-control-label" for="surname">Surname</
    label>
49                          {{form.surname(placeholder="Surname",class="form-control",
    type="text")}}
50                        </div>
51                      </div>
52
53
54                </div>
55                <div class="row justify-content-center">
56                    <div class="col-lg-4">
57                      <div class="form-group">
58                        <label class="form-control-label" for="email">Email</label>
59                        {{form.email(placeholder="Email", class="form-control", type="
    email")}}
60                      </div>
61                    </div>
62                    <div class="col-lg-4">
63                        <div class="form-group">
64                          <label class="form-control-label" for="birthdate">Date of
    birth</label>
65                          {{form.birthdate(placeholder="Date of birth", class="form-
    control")}}
66                        </div>
67                      </div>
68                    <div class="col-lg-4">
69                        <div class="form-group">
70                          <label class="form-control-label" for="height">Height (cm)</
    label>
71                          {{form.height(placeholder="Height", class="form-control",
    oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\..*)\./g, '$1')
    ;")}}
72                        </div>
73                      </div>
74                </div>
75                    <div class="row justify-content-center">
76                        <div class="col-lg-4">
77                            <div class="form-group">
78                              <label class="form-control-label" for="start_date">
    Starting date</label>
79                              {{form.start_date(class="form-control", onchange="
    updateEndTime()")}}
80                            </div>
81                          </div>
82                        <div class="col-lg-4">
83                            <div class="form-group">
84                              <label class="form-control-label" for="subscription">
    Subscription type</label>
85                              {{form.subscription(class="form-control", onchange="
    updateEndTime()")}}
86                            </div>
87                          </div>
88
89                          <div class="col-lg-4">
```

```
                            <div class="form-group">
                              <label class="form-control-label" for="end_date">
    Subscription until</label>
                              {{form.end_date(class="form-control", readonly=readonly)
    }}
                            </div>
                          </div>
                      </div>

                  <div class="row">
                    <div class="col-md-12">
                      <div class="form-group">
                          {{form.add(class="form-control btn btn-primary")}}
                      </div>
                    </div>
                  </div>
                </div>
              </form>
            </div>
          </div>
        </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}
<script>
    function updateEndTime(){
      var subscription = document.getElementById("subscription").value;
      var start_date = document.getElementById("start_date").value;

      if(subscription != "" && start_date != ""){
        //Check if the fields aren't empty, then set a end time depends on values
    selected before
        var dt = new Date(start_date);
        dt.setMonth(dt.getMonth() + parseInt(subscription));
        let [end_date] = dt.toISOString().split('T');
        document.getElementById("end_date").value = end_date
      }

    }
</script>
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Add new program {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

{% endfor %}



    <div class="row justify-content-center">

      <div class="col-xl-8 ">
        <div class="card">
          <div class="card-header">
```

```
            <div class="row align-items-center">
              <div class="col-8">
                <h3 class="mb-0">Add new program </h3>
              </div>
              <div class="col-4 text-right">

              </div>
            </div>
          </div>
          <div class="card-body">
            <form action="{{url_for('coach_blueprint.add_program')}}" method="POST"
    enctype="multipart/form-data">
                {{ form.csrf_token() }}
              <div class="pl-lg-4">
                <div class="row justify-content-center">
                    <div class="col-lg-6">
                        <div class="form-group">
                          <label class="form-control-label" for="program_type">Type</
    label>
                          {{form.type(placeholder="Type",class="form-control")}}
                        </div>
                      </div>
                      <div class="col-lg-6">
                        <div class="form-group">
                          <label class="form-control-label" for="program_file">File</
    label>
                          {{form.file(placeholder="Select your file",class="form-
    control", accept="application/pdf")}}
                        </div>
                      </div>
                </div>

                <div class="row">
                  <div class="col-md-12">
                    <div class="form-group">
                        {{form.client(class="form-control", value=clientId)}}
                        {{form.add(class="form-control btn btn-primary")}}
                    </div>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Add review {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

  {% endfor %}
```

```
18
19 <div class="row">
20   <div class="card card-body h-auto" style="width: 18rem; float: right;">
21     <form action="/add_review" method="POST" >
22     <h5 class="card-header">Add review {{form.type(class="form-control", value=
       reviewFields.type)}}{{form.target(class="form-control", value=reviewFields.target)
       }}</h5>
23     <div class="row justify-content-center" >
24         <div class="col-lg-4">
25             <div class="card border-light mb-3" style="max-width: 18rem;">
26                 <div class="card-header">{{reviewFields.field1}}</div>
27                 <div class="card-body">
28                   <h5 class="card-title">{{ form.field1(class="form-control form-range"
       , type="range", min="0", max="10",  oninput="outputUpdate(id, value)")}}<output for
       ="field1" id="field1-value">{{form.field1.data }}</output></h5>
29                 </div>
30               </div>
31               </div>
32
33               <div class="col-lg-4">
34                 <div class="card border-light mb-3" style="max-width: 18rem;">
35                     <div class="card-header">{{reviewFields.field2}}</div>
36                     <div class="card-body">
37                       <h5 class="card-title">{{ form.field2(class="form-control form-
       range", type="range", min="0", max="10",  oninput="outputUpdate(id, value)")}}<
       output for="field2" id="field2-value">{{form.field2.data }}</output></h5>
38                     </div>
39                   </div>
40               </div>
41     </div>
42     <div class="row justify-content-center">
43       <div class="col-lg-4">
44         <div class="card border-light mb-3" style="max-width: 18rem;">
45             <div class="card-header">{{reviewFields.field3}}</div>
46             <div class="card-body">
47               <h5 class="card-title">{{ form.field3(class="form-control form-range",
       type="range", min="0", max="10",  oninput="outputUpdate(id, value)")}}<output for="
       field3" id="field3-value">{{form.field3.data }}</output></h5>
48             </div>
49           </div>
50           </div>
51
52           <div class="col-lg-4">
53             <div class="card border-light mb-3" style="max-width: 18rem;">
54                 <div class="card-header">{{reviewFields.field4}}</div>
55                 <div class="card-body">
56                   <h5 class="card-title">{{ form.field4(class="form-control form-range"
       , type="range", min="0", max="10",  oninput="outputUpdate(id, value)")}}<output for
       ="field4" id="field4-value">{{form.field4.data }}</output></h5>
57                 </div>
58               </div>
59           </div>
60     </div>
61     <div class="row justify-content-center">
62       <div class="col-lg-8">
63         <div class="card border-light mb-3" style="max-width: 54rem;">
64             <div class="card-header">Comment</div>
65             <div class="card-body">
66               <h5 class="card-title">{{ form.comment(class="form-control")}}</h5>
67             </div>
68           </div>
69       </div>
70
71     </div>
72     <div class="pl-lg-4">
73       <div class="row">
74         <div class="col-md-12">
75           <div class="form-group">
76               {{ form.submit(class="btn btn-primary btn-block") }}
77           </div>
78         </div>
79       </div>
```

```
    </div>
    </form>
  </div>
</div>

{% endblock content %}
{% block javascripts %}
<script>
  // Update output value for each range input
    function outputUpdate(id, val){
        document.getElementById(id.concat("-value")).value = val;
    }
</script>
{% endblock javascripts %}
```

```
div{% extends 'layout/base.html' %}

{% block title %} Calendar {% endblock title %}

{% block stylesheets %}
<link type="text/css"  rel="stylesheet" href="{{ url_for('static', filename='assets/css/calendar.css') }}"/>

<style>

  body {
    font-family: Arial, Helvetica Neue, Helvetica, sans-serif;
    font-size: 14px;
  }

  #calendar {
    max-width: 1100px;
    margin: 0 ;
  }

  </style>
{% endblock stylesheets %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
  </div>

{% endfor %}
<div id="sessions" data-sessions="{{sessions}}"></div>
<div class="row">
    <div class="card card-body h-auto" style="width: 18rem; float: right;">
        <h3 class="card-header">Calendar :</h3>

        <div class="row">

          <div class="col-lg-6">
            <div class="card border-light mb-3" style="max-width: 50rem;">
              <div class="card-header">Sessions</div>
                <div class="card-body">
                  <div id='calendar'></div>
              </div>
            </div>
          </div>
          <div class="col-lg-6">
            <div class="card border-light mb-3" style="max-width: 50rem;">
              <div class="card-header">Add a session</div>
                <div class="card-body">
                  <form method="POST">
```

```
56                      {{ form.hidden_tag() }}
57                      <div class="row">
58                        <div class="col-lg-4">
59                          <div class="form-group">
60                            <label class="form-control-label" for="clients_list">Client :
      </label>
61                            {{form.client(class="form-control")}}
62                        </div>
63                      </div>
64                        <div class="col-lg-4">
65                          <div class="form-group">
66                            <label class="form-control-label" for="datepick">Date :</
      label>
67                               {{ form.date(placeholder="Date", class="form-control",
      type="date")}}
68                        </div>
69                        </div>
70
71                        <div class="col-lg-4">
72                          <div class="form-group">
73                            <label class="form-control-label" for="start_time">Time for
      the session</label>
74                            {{form.start_time(class="form-control" , type="time", min="
      07:00:00", max="20:00:00", onchange="updateEndTime()")}}
75                        </div>
76                      </div>
77                      </div>
78                      <div class="row">
79                        <div class="col-lg-4">
80                          <div class="form-group">
81                            <label class="form-control-label" for="type_picker">Type </
      label>
82                            {{form.type(class="form-control")}}
83                          </div>
84                        </div>
85                        <div class="col-lg-4">
86                          <div class="form-group">
87                            <label class="form-control-label" for="duration">Duration (h)
      </label>
88                            {{form.duration(class="form-control", onchange="updateEndTime
      ()")}}
89                        </div>
90                      </div>
91                      <div class="col-lg-4">
92                        <div class="form-group">
93                          <label class="form-control-label" for="end_time">End time :</
      label>
94                          {{form.end_time(class="form-control" , type="time")}}
95                        </div>
96                      </div>
97                    </div>
98                    <div class="row">
99                      <div class="col-lg-12">
100                       <div class="form-group">
101                         {{form.add(class="form-control btn btn-primary")}}
102                       </div>
103                     </div>
104
105                   </div>
106                   </div>
107                 </div>
108                 </form>
109             </div>
110           </div>
111         </div>
112
113       </div>
114
115   </div>
116 </div>
117 {% endblock content %}
118
```

```
<!-- Specific JS goes HERE -->
{% block javascripts %}
<script type="text/javascript" src="{{ url_for('static', filename='assets/js/calendar.
    js') }}"></script>
<script>
  function updateEndTime(){
    var duration = document.getElementById("duration").value;
    var start = document.getElementById("start_time").value;
    var date = document.getElementById("datepick").value;

    if(duration != "" && start != "" && date != ""){
      //Check if the fields aren't empty, then set a end time depends on values
    selected before
      var dt = new Date(date+"T"+start);
      dt.setHours(dt.getHours() + parseInt(duration));
      document.getElementById("end_time").value = dt.toTimeString().slice(0, 8);
    }

  }

  document.addEventListener('DOMContentLoaded', function() {
    var calendarEl = document.getElementById('calendar');

    var calendar = new FullCalendar.Calendar(calendarEl, {
      initialDate: new Date(Date.now()),
      initialView: 'timeGridWeek',
      headerToolbar: {
        left: 'prev,next today',
        center: 'title',
        right: 'dayGridMonth,timeGridWeek,timeGridDay,listWeek'
      },
      height: "600px",
      navLinks: true, // can click day/week names to navigate views
      editable: false,
      selectable: true,
      selectMirror: true,
      nowIndicator: true,
      events: {{ sessions | safe }}
    });

    calendar.render();
  });

  </script>
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Profile {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

  {% endfor %}



    <div class="row justify-content-center">

      <div class="col-xl-8 ">
        <div class="card">
          <div class="card-header">
```

```
            <div class="row align-items-center">
              <div class="col-8">
                <h3 class="mb-0">Change password </h3>
              </div>
              <div class="col-4 text-right">
                <a href="#!" class="btn btn-sm btn-warning">Password</a>
              </div>
            </div>
          </div>
          <div class="card-body">
            <form action="/change_password" method="POST">
                {{ form.hidden_tag() }}
              <div class="pl-lg-4">
                <div class="row justify-content-center">

                    <div class="col-lg-4">
                      <div class="form-group">
                        <label class="form-control-label" for="oldPassword">Old password</label>
                        {{ form.oldPassword(placeholder="Old password",class="form-control") }}
                      </div>
                    </div>
                </div>
                <div class="row justify-content-center">
                    <div class="col-lg-4">
                      <div class="form-group">
                        <label class="form-control-label" for="newPassowrd">New password</label>
                        {{ form.newPassword(placeholder="New password", class="form-control")}}
                      </div>
                    </div>
                </div>
                    <div class="row justify-content-center">
                        <div class="col-lg-4">
                            <div class="form-group">
                              <label class="form-control-label" for="confirmPassword">Confirm new password</label>
                              {{ form.confirmPassword(placeholder="Confirm new password", class="form-control")}}
                            </div>
                          </div>
                    </div>

                <div class="row">
                  <div class="col-md-12">
                    <div class="form-group">
                        {{ form.change(class="btn btn-primary btn-block") }}
                    </div>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Add check up {% endblock title %}

{% block content %}
```

```
6
7
8
9  <div class="container-fluid mt--8" style="margin: auto;">
10     {% for category, message in get_flashed_messages(with_categories=true) %}
11
12   <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
13     {{ message }}
14     <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
       "Close">Close</button>
15   </div>
16
17 {% endfor %}
18
19
20
21     <div class="row justify-content-center">
22
23       <div class="col-xl-8 ">
24         <div class="card">
25           <div class="card-header">
26             <div class="row align-items-center">
27               <div class="col-8">
28                 <h3 class="mb-0">Check up of the : {{checkUp.date.strftime("%d/%m/%Y")
       }}</h3>
29               </div>
30               <div class="col-4 text-right">
31
32               </div>
33             </div>
34           </div>
35           <div class="card-body">
36
37
38               <div class="pl-lg-4">
39
40                 <div class="row justify-content-center">
41                   <div class="col-lg-3">
42                     <div class="form-group">
43
44                       <div class="card-header text-center">Weight (kg)</div>
45                       <div class="card-body">
46                         <h5 class="card-title text-center"> {{checkUp.weight}}</h5>
47                       </div>
48                     </div>
49                   </div>
50                   <div class="col-lg-3">
51                     <div class="form-group">
52                         <div class="card-header text-center">Height (cm)</div>
53                         <div class="card-body">
54                           <h5 class="card-title text-center"> {{checkUp.height}}</h5>
55                         </div>
56                     </div>
57                   </div>
58                   <div class="col-lg-3">
59                     <div class="form-group">
60                         <div class="card-header text-center">Water (%)</div>
61                         <div class="card-body">
62                           <h5 class="card-title text-center"> {{checkUp.
       water_percentage}}</h5>
63                         </div>
64                     </div>
65                   </div>
66                   <div class="col-lg-3">
67                     <div class="form-group">
68                         <div class="card-header text-center">Protein (%)</div>
69                         <div class="card-body">
70                           <h5 class="card-title text-center"> {{checkUp.
       protein_percentage}}</h5>
71                         </div>
72                     </div>
73                   </div>
```

```
74                </div>
75                    <div class="row justify-content-center">
76                      <div class="col-lg-3">
77                        <div class="form-group">
78                            <div class="card-header text-center">Muscle Mass (%)</div>
79                            <div class="card-body">
80                              <h5 class="card-title text-center"> {{checkUp.
    muscle_mass_percentage}}</h5>
81                            </div>
82                        </div>
83                      </div>
84                      <div class="col-lg-3">
85                        <div class="form-group">
86                            <div class="card-header text-center">Body Fat (%)</div>
87                            <div class="card-body">
88                              <h5 class="card-title text-center"> {{checkUp.
    bodyfat_percentage}}</h5>
89                            </div>
90                        </div>
91                      </div>
92                      <div class="col-lg-3">
93                        <div class="form-group">
94                            <div class="card-header text-center">Bone Mass (%)</div>
95                            <div class="card-body">
96                              <h5 class="card-title text-center"> {{checkUp.
    bone_mass_percentage}}</h5>
97                            </div>
98                        </div>
99                      </div>
100                     <div class="col-lg-3">
101                       <div class="form-group">
102                           <div class="card-header text-center">BMR </div>
103                           <div class="card-body">
104                             <h5 class="card-title text-center"> {{checkUp.bmr}}</h5>
105                           </div>
106                       </div>
107                     </div>
108                   </div>
109
110                   <div class="row justify-content-center">
111                     <div class="col-lg-3">
112                       <div class="form-group">
113                           <div class="card-header text-center">Muscle Mass (kg)</div>
114                           <div class="card-body">
115                             <h5 class="card-title text-center"> {{checkUp.muscle_mass
    }}</h5>
116                           </div>
117                       </div>
118                     </div>
119                     <div class="col-lg-3">
120                       <div class="form-group">
121                           <div class="card-header text-center">Body Fat (kg)</div>
122                           <div class="card-body">
123                             <h5 class="card-title text-center"> {{checkUp.
    bodyfat_mass}}</h5>
124                           </div>
125                       </div>
126                     </div>
127                     <div class="col-lg-3">
128                       <div class="form-group">
129                           <div class="card-header text-center">Visceral fat (kg)</div
    >
130                           <div class="card-body">
131                             <h5 class="card-title text-center"> {{checkUp.
    fat_visceral}}</h5>
132                           </div>
133                       </div>
134                     </div>
135                     <div class="col-lg-3">
136                       <div class="form-group">
137                           <div class="card-header text-center">Bone Mass (kg)</div>
138                           <div class="card-body">
```

```
                                <h5 class="card-title text-center"> {{checkUp.bone_mass}}
    </h5>
                            </div>
                        </div>
                    </div>
                    <div class="row justify-content-center">
                        <div class="col-lg-3">
                            <div class="form-group">
                                <div class="card-header text-center">Lean body mass (kg)</
    div>
                                <div class="card-body">
                                    <h5 class="card-title text-center"> {{checkUp.
    leanbody_mass}}</h5>
                                </div>
                            </div>
                        </div>
                        <div class="col-lg-3">
                            <div class="form-group">
                                <div class="card-header text-center">Body age</div>
                                <div class="card-body">
                                    <h5 class="card-title text-center"> {{checkUp.body_age}}<
    /h5>
                                </div>
                            </div>
                        </div>
                        <div class="col-lg-3">
                            <div class="form-group">
                                <div class="card-header text-center">BMI </div>
                                <div class="card-body">
                                    <h5 class="card-title text-center"> {{checkUp.bmi}}</h5>
                                </div>
                            </div>
                        </div>
                    </div>

                </div>

            </div>
          </div>
        </div>


{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Client {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

{% endfor %}



```

```
21      <div class="row">
22
23        <div class="col-xl-8 ">
24          <div class="card">
25            <div class="card-header">
26              <div class="row align-items-center">
27                <div class="col-8">
28                  <h3 class="mb-0">Edit profile </h3>
29                </div>
30                <div class="col-4 text-right">
31                  <a href="{{ url_for('coach_blueprint.check_up', clientId=client.id)}}"
    class="btn btn-sm btn-primary">Check up</a>
32                </div>
33              </div>
34            </div>
35            <div class="card-body">
36              <form action="/profile" method="POST" enctype="multipart/form-data">
37                  {{ form.hidden_tag() }}
38                <h6 class="heading-small text-muted mb-4">User information</h6>
39                <div class="pl-lg-4">
40                  <div class="row">
41                      <div class="col-lg-4">
42                        <div class="form-group">
43                          <label class="form-control-label" for="name">First name</label>
44                          {{ form.name(placeholder="Name",class="form-control",type="text
    ", value=client.name) }}
45                        </div>
46                      </div>
47                      <div class="col-lg-4">
48                        <div class="form-group">
49                          <label class="form-control-label" for="surname">Last name</
    label>
50                          {{ form.surname(placeholder="Surname", class="form-control",
    type="text", value=client.surname)}}
51                        </div>
52                      </div>
53
54                </div>
55                <div class="row">
56                  <div class="col-lg-4">
57                    <div class="form-group">
58                      <label class="form-control-label" for="birthdate"> Birthdate</
    label>
59                      {{ form.birthdate(placeholder="Birthdate", class="form-control",
    type="date", value=client.birthdate)}}
60                    </div>
61                  </div>
62                  <div class="col-lg-4">
63                    <div class="form-group">
64                      <label class="form-control-label" for="email">Email address</
    label>
65                      {{ form.email(placeholder="Email", class="form-control", type="
    email", value=client.email)}}
66                    </div>
67                  </div>
68                  <div class="col-lg-4">
69                    <div class="form-group">
70                        <div class="card-profile-image">
71                            <a href="#">
72                              {% if client.profile_pic %}
73                                 <img src="{{ url_for('static', filename='/assets/img/
    profile/' + client.profile_pic)}}"  align="right" id="profile_pic" class="rounded-
    circle" width="120" height="120">
74
75                              {% endif %}
76                            </a>
77                      </div>
78                    </div>
79                  </div>
80                </div>
81                <div class="row">
82                    <div class="col-lg-4">
```

```
                        <div class="form-group">
                            <label class="form-control-label" for="weight">Weight (kg)</label>
                            {{ form.weight(placeholder="Weight", class="form-control", type="number", value=physicalInfo.weight, disabled=true)}}
                        </div>
                    </div>
                    <div class="col-lg-4">
                        <div class="form-group">
                            <label class="form-control-label" for="height">Height (cm)</label>
                            {{ form.height(placeholder="Height", class="form-control", type="number", value=physicalInfo.height, disabled=true)}}
                        </div>
                    </div>
                    <div class="col-lg-4">
                      <div class="form-group">
                        <label class="form-control-label" for="register_date">Date of register </label>
                            {{ form.register_date(placeholder="Register Date", class="form-control", type="date", value=client.created_at.strftime("%Y-%m-%d"), readonly=readonly)}}
                      </div>
                  </div>
                </div>

                <div class="row">
                    <div class="col-lg-6">
                        <div class="form-group">
                          <label class="form-control-label" for="card_update">Card ID #</label>
                            {{ form.card_id(placeholder="Card ID", class="form-control", type="text", value=client.card_id if client.card_id != None, disabled=true)}}
                            <a href="{{ url_for('coach_blueprint.new_card', clientId=client.id) }}" class="btn btn-sm btn-primary">Change</a>

                        </div>
                    </div>
                    <div class="col-lg-6">
                        <div class="form-group">
                          <label class="form-control-label" for="subscription_until">Subscription Until</label>
                            {{ form.subscriptionEnd(placeholder="Subscription Until", class="form-control", type="date", value=subscriptionUntil, disabled=true)}}
                            <a href="{{url_for('coach_blueprint.new_subscription', clientId=client.id)}}" class="btn btn-sm btn-success">Renew subscription</a>
                            <a href="{{url_for('coach_blueprint.cancel_subscription', clientId=client.id)}}" class="btn btn-sm btn-danger">Cancel</a>
                        </div>
                    </div>
                </div>
            </div>
            <hr class="my-4" />
            <!-- Address -->
            <h6 class="heading-small text-muted mb-4">Contact information</h6>
            <div class="pl-lg-4">
              <div class="row">
                <div class="col-md-12">
                  <div class="form-group">
                    <label class="form-control-label" for="address_update">Address</label>
                    {{ form.address(placeholder="Address", class="form-control", type="text", value=client.address if client.address != None)}}
                  </div>
                </div>
              </div>
              <div class="row">
                <div class="col-lg-4">
                  <div class="form-group">
                    <label class="form-control-label" for="city_update">City</label>
                    {{ form.city(placeholder="City", class="form-control", type="text", value=client.city if client.address != None)}}
```

```
                    </div>
                  </div>
                  <div class="col-lg-4">
                    <div class="form-group">
                      <label class="form-control-label" for="country_update">Country</label>
                      {{ form.country(placeholder="Country", class="form-control", type="text", value=client.country if client.country != None)}}
                    </div>
                  </div>
                  <div class="col-lg-4">
                    <div class="form-group">
                      <label class="form-control-label" for="input-country">Postal code</label>
                      {{ form.npa(placeholder="Postal code", class="form-control", type="number", value=client.npa if client.npa != None)}}
                    </div>
                  </div>
                </div>
              </div>
              <hr class="my-4" />
              <div class="row">
                <div class="col-lg-12">
                  <div class="form-group">
                      {{ form.change(class="btn btn-primary btn-block") }}
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>
      <div class="card card-body h-auto" style="width: 18rem; float: right;">
        <div class="card-header">
        <div class="row align-items-center">
          <div class="col-8">
            <h3 class="mb-0">Programs </h3>
          </div>
          <div class="col-4 text-right">
            <a href="{{ url_for('coach_blueprint.add_program', clientId=client.id) }}" class="btn btn-sm btn-primary">Upload new program</a>
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-lg-12">
            <div class="form-group">
              {% if workoutProgram != None %}
              <label class="form-control-label">Workout Program added the {{ workoutProgram.date.strftime('%d/%m/%Y')}}: </label>
          </br>
            <a href="{{ url_for('coach_blueprint.program', programId=workoutProgram.id, programType="workout") }}" class="btn btn-sm btn-warning">Workout Program</a> added by <strong>{{workoutProgram.name}} {{workoutProgram.surname}}</strong>
            {% else %}
            <h4>No workout plan available yet</h4>
            {% endif %}
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col-lg-12">
            <div class="form-group">
              {% if dietProgram != None %}
              <label class="form-control-label">Diet Program added the {{dietProgram.date.strftime('%d/%m/%Y')}}: </label>
            </br>
              <a href="{{ url_for('coach_blueprint.program', programId=dietProgram.id, programType="diet") }}" class="btn btn-sm btn-success">Diet Program</a> added by <strong>{{dietProgram.name}} {{dietProgram.surname}}</strong>
              {% else %}
              <h4>No diet plan available yet</h4>
```

```
                {% endif %}
              </div>
            </div>
          </div>

          <hr class="my-4" />
          <div class="row">
            <div class="col-8">
              <h3 class="mb-0">Check up :</h3>
            </div>

          </div>
          {% if checkUps.count() >0 %}
          <table class="table">
            <thead>
                <tr>
                    <th scope="col">Type</th>
                    <th scope="col">Date</th>
                    <th scope="col">Details</th>
                </tr>
            </thead>

            {% for checkUp in checkUps %}
            <tr>
                <td>
                    {{ checkUp.date.strftime("%d/%m/%Y, %H:%M:%S") }}
                </td>
                <td>
                    {{ checkUp.weight }}KG
                </td>
                <td>
                  <a class="btn btn-sm btn-primary" href="{{ url_for('client_blueprint.checkup', checkUpId=checkUp.id) }}">Details</a>
                </td>
            </tr>
            {% endfor %}
          </table>
          {%else%}
          <h4>No check up done yet</h4>
          {%endif%}
    </div>
  </div>

  <div class="row">
    <div class="col-xl-8">
      <div class="card" >
        <div class="card-header">
          <div class="row align-items-center">
            <div class="col-8">
              <h3 class="mb-0">Analytics  {% if wrktTypeCount == 0 %} - No workout done yet {%endif%}</h3>
            </div>
            <div class="col-4 text-right">
              <a href="#!" class="btn btn-sm btn-warning">Statistic</a>
            </div>
          </div>
        </div>
        <div class="card-body">
          <div class="row">
          <div class="col-lg-4">
          <div class="chart-container" style="max-width: 25rem; position: relative;">

            <canvas id="workoutTypes"></canvas>

          </div>
        </div>
          <div class="col-lg-4">
            <div class="chart-container" style="max-width: 40rem; position: relative;">
              <canvas id="workoutCount"></canvas>
            </div>
          </div>
```

```
268             <div class="col-lg-4">
269                 <div class="card" style="max-width: 12rem;">
270                   <div class="card-header">Calories burned
271                     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
    viewBox="0 0 24 24" fill="currentColor" class="text-orange"> <g> <path fill="none"
    d="M0 0h24v24H0z"/> <path d="M12 23a7.5 7.5 0 0 1-5.138-12.963C8.204 8.774 11.5 6.5
     11 1.5c6 4 9 8 3 14 1 0 2.5 0 5-2.47.27.773.5 1.604.5 2.47A7.5 7.5 0 0 1 12 23z"/>
     </g> </svg>
272                   </div>
273                   <div class="card-body">
274                     <h5 class="card-title">{{ avgCalories }}</h5>
275                   </div>
276                 </div>
277                 <div class="card" style="max-width: 11rem;">
278                   <div class="card-header">Heart rate avg
279                     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
    ="currentColor" class="bi bi-heart-pulse-fill text-red" viewBox="0 0 16 16">
280                       <path fill-rule="evenodd" d="M1.475 9C2.702 10.84 4.779 12.871 8
    15c3.221-2.129 5.298-4.16 6.525-6H12a.5.5 0 0 1-.464-.314l-1.457-3.642-1.598 5.593a
    .5.5 0 0 1-.945.049L5.889 6.568l-1.473 2.21A.5.5 0 0 1 4 9H1.475ZM.879 8C-2.426
    1.68 4.41-2 7.824 1.143c.06.055.119.112.176.171a3.12 3.12 0 0 1 .176-.17C11.59-2
    18.426 1.68 15.12 8h-2.783l-1.874-4.686a.5.5 0 0 0-.945.049L7.921 8.956 6.464 5.314
    a.5.5 0 0 0-.88-.091L3.732 8H.88Z"/>
281                     </svg>
282                   </div>
283                   <div class="card-body">
284                     <h5 class="card-title">{{ avgHeartRate }}</h5>
285                   </div>
286                 </div>
287                 <div class="card" style="max-width: 11rem;">
288                   <div class="card-header">Time working out
289                     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
    ="currentColor" class="bi bi-clock" viewBox="0 0 16 16">
290                       <path d="M8 3.5a.5.5 0 0 0-1 0V9a.5.5 0 0 0 .252.434l3.5 2a.5.5 0
     0 0 .496-.868L8 8.71V3.5z"/>
291                       <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm7-8A7 7 0 1 1 1 8a7
    7 0 0 1 14 0z"/>
292                     </svg>
293                   </div>
294                   <div class="card-body">
295                     <h5 class="card-title">{{ totalTime }}</h5>
296                   </div>
297                 </div>
298           </div>
299         </div>
300         <div class="row">
301           <div class="col-lg-12">
302             <div class="chart-container" style="max-width: 40rem; position: relative;
    ">
303               <canvas id="weightUpdate"></canvas>
304             </div>
305           </div>
306         </div>
307         </div>
308       </div></div>
309       <div class="card card-body h-auto" style="width: 18rem; float: right;">
310         <div class="card-header">
311         <div class="row align-items-center">
312           <div class="col-8">
313             <h3 class="mb-0">Reports </h3>
314           </div>
315           <div class="col-4 text-right">
316           </div>
317         </div>
318       </div>
319         <table class="table">
320             <thead>
321                 <tr>
322                     <th scope="col">Type</th>
323                     <th scope="col">Date</th>
324                     <th scope="col">Details</th>
325                 </tr>
```

```
326                 </thead>
327
328                 {% for review in reviews %}
329                 <tr>
330                     <td>
331                         {{ review.type }}
332                     </td>
333                     <td>
334                         {{ review.date.strftime("%d/%m/%Y, %H:%M:%S") }}
335                     </td>
336                     <td>
337                       <a class="btn btn-sm btn-primary" href="{{ url_for('
    client_blueprint.review', Id=review.id, Type=review.type) }}">Details</a>
338                     </td>
339                 </tr>
340                 {% endfor %}
341               </table>
342         </div>
343       </div>
344       </div>
345     </div>
346
347
348 {% endblock content %}
349
350 <!-- Specific JS goes HERE -->
351 {% block javascripts %}
352 <script>
353   var workoutList = []
354   {% for workoutType in wrktTypeList %}
355     workoutList.push('{{workoutType}}'.replaceAll('&#34;',''));
356   {% endfor %}
357   const ctx = document.getElementById('workoutTypes').getContext('2d');
358   const ctx2 = document.getElementById('workoutCount').getContext('2d');
359   const ctx3 = document.getElementById('weightUpdate').getContext('2d');
360   const weightUpdateChart = new Chart(ctx3,{
361       type: 'line',
362       data:{
363         labels: [
364           'Jan',
365           'Feb',
366           'Mar',
367           'Apr',
368           'Mai',
369           'Jun',
370           'Jul',
371           'Aug',
372           'Sept',
373           'Oct',
374           'Nov',
375           'Dec'
376         ],
377         datasets: [{
378           label: 'Weight update',
379           data: {{weightUpdate}},
380           fill: false,
381           borderColor: 'rgb(0, 128, 128)',
382           tension: 0.1
383         }]
384       },
385       options: {
386         scales: {
387             y: {
388                 min: 40,
389                 suggestedMin: 50,
390                 suggestedMax: 100
391             }
392         },
393         elements: {
394         line: {
395           borderWidth: 3
396         }
```

```
      }
      }
  });
  const workoutCountChart = new Chart(ctx2, {
      type: 'line',
      data: {
        labels: [
          'Jan',
          'Feb',
          'Mar',
          'Apr',
          'Mai',
          'Jun',
          'Jul',
          'Aug',
          'Sept',
          'Oct',
          'Nov',
          'Dec'
        ],
        datasets: [{
          label: 'Number of workouts',
          data: {{nbWorkoutPerMonth}},
          fill: false,
          borderColor: 'rgb(128, 0, 128)',
          tension: 0.1
        }]
      },
      options: {
        r: {
          suggestedMin: 0
        },
        elements: {
        line: {
          borderWidth: 3
        }
      }
      }
    });
  const workoutTypeChart = new Chart(ctx, {
    type: 'doughnut',
    data: {
      labels: workoutList,
      datasets: [{
        label: 'Stats',
        data: {{ wrktTypeCount }},
        backgroundColor: [
          'rgb(255, 99, 132)',
          'rgb(54, 162, 235)',
          'rgb(255, 205, 86)'
        ],
        hoverOffset: 4
      }]

    }
  });

  </script>
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Payment {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
```

```
13      {{ message }}
14      <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
        "Close">Close</button>
15    </div>
16
17  {% endfor %}
18
19  <div class="row">
20        <div class="card card-body h-auto" style="width: 18rem; float: right;">
21          <div class="card-header">
22              <div class="row align-items-center">
23                <div class="col-8">
24                  <h3 class="mb-0">Next Session : </h3>
25                </div>
26
27              </div>
28            </div>
29
30          {% if nextClient != None %}
31          <div class="row justify-content-center">
32
33              <div class="col-lg-2">
34                  <div class="card border-light mb-3" style="max-width: 18rem;">
35                      <div class="card-header">Client</div>
36
37                      <div class="card-body">
38                        <h5 class="card-title">
39
40                          <img src="{{ url_for('static', filename='/assets/img/profile/'
        + nextClient.profile_pic)}}"  align="left" id="profile_pic" class="rounded-circle"
        width="50" height="50">
41                              {{nextClient.name}} {{nextClient.surname}}
42
43                        </h5>
44                      </div>
45                    </div>
46          </div>
47          <div class="col-lg-4">
48
49            <div class="card border-light mb-3" style="max-width: 36rem;">
50                <div class="card-header">Date</div>
51                <div class="card-body">
52
53                    {{nextClient.start_time.strftime("%d %B, %Y")}}
54
55                </div>
56              </div>
57          </div>
58          <div class="col-lg-2">
59
60            <div class="card border-light mb-3" style="max-width: 18rem;">
61                <div class="card-header">Card ID</div>
62                <div class="card-body">
63                  <h5 class="card-title">{{nextClient.card_id}}</h5>
64                </div>
65              </div>
66          </div>
67        </div>
68        <div class="row justify-content-center">
69            <div class="col-lg-2">
70                <div class="card border-light mb-3" style="max-width: 18rem;">
71                    <div class="card-header">Session time</div>
72                    <div class="card-body">
73                      <h5 class="card-title">
74
75                          {{nextClient.start_time.strftime("%H:%M:%S")}}
76
77                        </h5>
78                    </div>
79                  </div>
80            </div>
81            <div class="col-lg-2">
```

```
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">Duration</div>
                    <div class="card-body">
                      <h5 class="card-title">

                        {{nextClient.duration}}

                    </h5>
                    </div>
                  </div>
            </div>
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">Type</div>
                    <div class="card-body">
                      <h5 class="card-title">

                        <img src="{{ url_for('static', filename='/assets/img/sports/' + nextClient.workoutLogo)}}" alt="{{nextClient.workoutLogo}}" class="rounded mx-auto d-block" width="50" height="50">

                    </h5>
                    </div>
                  </div>
            </div>
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">Last Workout </div>
                    <div class="card-body">
                      <h5 class="card-title">   {{clientLastWorkout.date.strftime("%d/%m/%Y, %H:%M:%S") if clientLastWorkout != None else "No workout yet"}}</h5>
                    </div>
                  </div>
            </div>
        </div>
        {% else %}
        <h2 align="center">No session registered</h2>
        {% endif %}
        </div>
      </div>
    </div>

    <div class="row" style="margin:auto;">
      <div class="col-xl-12" >
        <div class="card" style="max-width: 114rem;"">
          <div class="card-header">
            <div class="row align-items-center">
              <div class="col-8">
                <h3 class="mb-0">Clients</h3>
              </div>
              <div class="col-4 text-right">
                <a href="{{ url_for('coach_blueprint.add_client') }}" class="btn btn-sm btn-primary">Add new client</a>
              </div>
            </div>
          </div>


        <table class="table">
        <thead>
            <tr>
                <th scope="col"></th>
                <th scope="col">Name</th>
                <th scope="col">Subscription end date</th>
                <th scope="col">Last Workout </th>
                <th scope="col">Card ID</th>
                <th scope="col">Active (Last 24h)</th>
            </tr>
        </thead>
        {% for client in clients %}

        <tr>
```

```
            <td>
                <img src="{{ url_for('static', filename='/assets/img/profile/' + client.profile_pic)}}"  align="left" id="profile_pic" class="rounded-circle" width="50" height="50">

            </td>
            <td>
                <a href="{{ url_for('coach_blueprint.client', clientId=client.id) }}" class="btn btn-sm btn-primary"> {{client.name}} {{client.surname}}</a>
            </td>
            <td>
               {{client.subscriptionEnd.strftime("%d/%m/%Y")}}
            </td>
            <td>
                {{client.lastWorkout.date.strftime("%d/%m/%Y, %H:%M:%S") if client.lastWorkout != None else "No workout yet"}}
            </td>
            <td>
                {{client.card_id}}
            </td>
            <td>
                {% if client.isActive %}
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-check" viewBox="0 0 16 16">
                  <path d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.061l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z"/>
                </svg>
                {% endif %}

            </td>
        </tr>
        {% endfor %}
        </table>


        </div>
      </div>
    </div>

{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Home {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}
{% for category, message in get_flashed_messages(with_categories=true) %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
  {{ message }}
  <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
</div>

{% endfor %}

<div class="row">
  <div class="card card-body h-auto" style="width: 18rem; float: right;">
    <h5 class="card-header">Next sessions</h5>
    <table class="table">
        <thead>
            <tr>
                <th scope="col"></th>
                <th scope="col">Type</th>
```

```
                <th scope="col">Date</th>
                <th scope="col">Duration</th>
                <th scope="col">Hour</th>
                <th scope="col">Notif</th>
            </tr>
        </thead>
        {% for session in nextSessions %}
        <br>
        <tr>
            <td>
              {% if session.logo %}
                <img src="{{ url_for('static', filename='/assets/img/sports/' + session.logo)}}" alt="{{session.logo}}" class="rounded mx-auto d-block">
              {% endif %}
            </td>
            <td>
                {{ session.title }}
            </td>
            <td>
                {{ session.start_time.strftime('%d/%m/%Y') }}
            </td>
            <td>
                {{ session.duration }}
            </td>
            <td>
                {{ session.start_time.strftime('%X') }}
            </td>
            <td>
              <button type="button" class="btn btn-primary">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-bell" viewBox="0 0 16 16">
                  <path d="M8 16a2 2 0 0 0 2-2H6a2 2 0 0 0 2 2zM8 1.918l-.797.161A4.002 4.002 0 0 0 4 6c0 .628-.134 2.197-.459 3.742-.16.767-.376 1.566-.663 2.258h10.244c-.287-.692-.502-1.49-.663-2.258C12.134 8.197 12 6.628 12 6a4.002 4.002 0 0 0-3.203-3.92L8 1.917zM14.22 12c.223.447.481.801.78 1H1c.299-.199.557-.553.78-1C2.68 10.2 3 6.88 3 6c0-2.42 1.72-4.44 4.005-4.901a1 1 0 1 1 1.99 0A5.002 5.002 0 0 1 13 6c0 .88.32 4.2 1.22 6z"></path>
                </svg>
              </button>
            </td>
        </tr>
        {% endfor %}
        </table>
  </div>
</div>

{% endblock content %}
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Profile {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
  </div>

  {% endfor %}



```

```
21      <div class="row">
22
23        <div class="col-xl-8 ">
24          <div class="card">
25            <div class="card-header">
26              <div class="row align-items-center">
27                <div class="col-8">
28                  <h3 class="mb-0">Edit profile </h3>
29                </div>
30                <div class="col-4 text-right">
31                  <a href="#!" class="btn btn-sm btn-primary">Settings</a>
32                </div>
33              </div>
34            </div>
35            <div class="card-body">
36              <form action="/profile" method="POST" enctype="multipart/form-data">
37                  {{ form.hidden_tag() }}
38                <h6 class="heading-small text-muted mb-4">User information</h6>
39                <div class="pl-lg-4">
40                  <div class="row">
41                      <div class="col-lg-4">
42                        <div class="form-group">
43                          <label class="form-control-label" for="name_update">First name<
    /label>
44                          {{ form.name(placeholder="Name",class="form-control",type="text
    ", value= current_user.name) }}
45                        </div>
46                      </div>
47                      <div class="col-lg-4">
48                        <div class="form-group">
49                          <label class="form-control-label" for="surname_update">Last
    name</label>
50                          {{ form.surname(placeholder="Surname", class="form-control",
    type="text", value=current_user.surname)}}
51                        </div>
52                      </div>
53                      <div class="col-lg-4">
54                          <div class="form-group">
55                              <div class="card-profile-image">
56                                  <a href="#">
57                                      {% if current_user.profile_pic %}
58
59                                      <img src="{{ url_for('static', filename='/assets/
    img/profile/' + current_user.profile_pic)}}"  align="right" id="profile_pic" class=
    "rounded-circle" width="120" height="120">
60
61                                      {% else %}
62
63                                      <img src="{{ url_for('static', filename='/assets/
    img/profile/default_profile_pic.png')}}" align="right" id="profile_pic" class="
    rounded-circle" width="120" height="120">
64
65                                      {% endif %}
66                                  </a>
67                              </div>
68                          </div>
69                      </div>
70                  </div>
71                  <div class="row">
72                    <div class="col-lg-4">
73                      <div class="form-group">
74                        <label class="form-control-label" for="birthdate_update">
    Birthdate</label>
75                        {{ form.birthdate(placeholder="Birthdate", class="form-control",
    type="date", value=current_user.birthdate)}}
76                      </div>
77                    </div>
78                    <div class="col-lg-4">
79                      <div class="form-group">
80                        <label class="form-control-label" for="email_update">Email
    address</label>
81                        {{ form.email(placeholder="Email", class="form-control", type="
```

```
    email", value=current_user.email)}}
82                      </div>
83                  </div>
84                  <div class="col-lg-4">
85                    <div class="form-group">
86                  <label class="form-control-label" for="input-file">
87                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
    ="currentColor" class="bi bi-file-earmark-person " viewBox="0 0 16 16">
88                      <path d="M11 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>
89                      <path d="M14 14V4.5L9.5 0H4a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h8a2 2
    0 0 0 2-2zM9.5 3A1.5 1.5 0 0 0 11 4.5h2v9.255S12 12 8 12s-5 1.755-5 1.755V2a1 1 0 0
     1 1-1h5.5v2z"/>
90                    </svg>
91                  </label>
92                  {{ form.profile_pic(class="form-control", type="file") }}
93                </div>
94              </div>
95                </div>
96                {% if current_user.role == 1%}
97                <div class="row">
98                    <div class="col-lg-4">
99                        <div class="form-group">
100                           <label class="form-control-label" for="weight_update">
    Weight (kg)</label>
101                           {{ form.weight(placeholder="Weight", class="form-control",
    type="number", value=current_user.physicalInfo.weight, disabled=true)}}
102                       </div>
103                   </div>
104                   <div class="col-lg-4">
105                       <div class="form-group">
106                           <label class="form-control-label" for="height_update">
    Height (cm)</label>
107                           {{ form.height(placeholder="Height", class="form-control",
    type="number", value=current_user.physicalInfo.height, disabled=true)}}
108                       </div>
109                   </div>
110             </div>
111
112             <div class="row">
113                 <div class="col-lg-6">
114                     <div class="form-group">
115                         <label class="form-control-label" for="card_update">Card ID
     #</label>
116                         {{ form.card_id(placeholder="Card ID", class="form-control"
    , type="text", value=current_user.card_id if current_user.card_id != None, disabled
    =true)}}
117                     </div>
118                 </div>
119                 <div class="col-lg-6">
120                     <div class="form-group">
121                         <label class="form-control-label" for="subscription_until">
    Subscription Until</label>
122                         {{ form.subscriptionEnd(placeholder="Subscription Until",
    class="form-control",  value=current_user.subscriptionEnd if current_user.
    subscriptionEnd != None, disabled=true)}}
123                     </div>
124                 </div>
125             </div>
126           </div>
127           {%endif%}
128           <hr class="my-4" />
129           <!-- Address -->
130           <h6 class="heading-small text-muted mb-4">Contact information</h6>
131           <div class="pl-lg-4">
132             <div class="row">
133               <div class="col-md-12">
134                 <div class="form-group">
135                   <label class="form-control-label" for="address_update">Address</
    label>
136                   {{ form.address(placeholder="Address", class="form-control", type
    ="text", value=current_user.address if current_user.address != None)}}
137                 </div>
```

```
                </div>
              </div>
              <div class="row">
                <div class="col-lg-4">
                  <div class="form-group">
                    <label class="form-control-label" for="city_update">City</label>
                    {{ form.city(placeholder="City", class="form-control", type="text", value=current_user.city if current_user.city != None)}}
                  </div>
                </div>
                <div class="col-lg-4">
                  <div class="form-group">
                    <label class="form-control-label" for="country_update">Country</label>
                    {{ form.country(placeholder="Country", class="form-control", type="text", value=current_user.country if current_user.country != None)}}
                  </div>
                </div>
                <div class="col-lg-4">
                  <div class="form-group">
                    <label class="form-control-label" for="input-country">Postal code</label>
                    {{ form.npa(placeholder="Postal code", class="form-control", type="number", value=current_user.npa if current_user.npa != None)}}
                  </div>
                </div>
              </div>
            </div>
            <hr class="my-4" />
            <div class="pl-lg-4">
              <div class="row">
                <div class="col-md-12">
                  <div class="form-group">
                      {{ form.update(class="btn btn-primary btn-block") }}
                      <a class="btn btn-sm btn-warning" href="{{url_for('client_blueprint.change_password')}}">Change password</a>
                  </div>
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>

    {% if current_user.role == 1%}
    <div class="card card-body h-auto" style="width: 18rem; float: right;">
      <div class="card-header">
      <div class="row align-items-center">
        <div class="col-8">
          <h3 class="mb-0">Programs </h3>
        </div>
        <div class="col-4 text-right">
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-lg-12">
          <div class="form-group">
            {% if current_user.workoutProgram != None %}
            <label class="form-control-label">Workout Program added the {{current_user.workoutProgram.date.strftime('%d/%m/%Y')}}: </label>
            </br>
            <a href="{{ url_for('coach_blueprint.program', programId=current_user.workoutProgram.id, programType="workout") }}" class="btn btn-sm btn-warning">Workout Program</a> added by <strong>{{current_user.workoutProgram.name}} {{current_user.workoutProgram.surname}}</strong>
            {% else %}
            <h4>No workout plan available yet</h4>
            {% endif %}
          </div>
        </div>
```

```
        </div>
        <div class="row">
          <div class="col-lg-12">
              <div class="form-group">
                {% if current_user.dietProgram != None %}
                <label class="form-control-label">Diet Program added the {{current_user
.dietProgram.date.strftime('%d/%m/%Y')}}: </label>
              </br>
                <a href="{{ url_for('coach_blueprint.program', programId=current_user.
dietProgram.id, programType="diet") }}" class="btn btn-sm btn-success">Diet Program
</a> added by <strong>{{current_user.dietProgram.name}} {{current_user.dietProgram.
surname}}</strong>
              {% else %}
              <h4>No diet plan available yet</h4>
              {% endif %}
            </div>
          </div>
        </div>
        <hr class="my-4" />
        <div class="row">
          <div class="col-8">
            <h3 class="mb-0">Check up :</h3>
          </div>

        </div>
        {% if current_user.checkUps.count() >0 %}
        <table class="table">
          <thead>
              <tr>
                  <th scope="col">Type</th>
                  <th scope="col">Date</th>
                  <th scope="col">Details</th>
              </tr>
          </thead>


          {% for checkUp in current_user.checkUps %}
          <tr>
              <td>
                  {{ checkUp.date.strftime("%d/%m/%Y, %H:%M:%S") }}
              </td>
              <td>
                  {{ checkUp.weight }}KG
              </td>
              <td>
                <a class="btn btn-sm btn-primary" href="{{ url_for('client_blueprint.
checkup', checkUpId=checkUp.id) }}">Details</a>
              </td>
          </tr>
          {% endfor %}

        </table>
        {%else%}
          <h4>No check up done yet</h4>
          {%endif%}
    </div>
  </div>



  <div class="row">
    <div class="col-xl-8">
      <div class="card" >
        <div class="card-header">
          <div class="row align-items-center">
            <div class="col-8">
              <h3 class="mb-0">Analytics {% if current_user.workoutTypeCount == 0 %}
- No workout done yet {%endif%}</h3>
            </div>
            <div class="col-4 text-right">
              <a href="#!" class="btn btn-sm btn-warning">Statistic</a>
            </div>
```

```
            </div>
          </div>
          <div class="card-body">
            <div class="row">
            <div class="col-lg-4">
            <div class="chart-container" style="max-width: 25rem; position: relative;">
              <canvas id="workoutTypes" ></canvas>
            </div>
          </div>
            <div class="col-lg-4">
              <div class="chart-container" style="max-width: 40rem; position: relative;
">
                <canvas id="workoutCount"></canvas>
              </div>
            </div>
            <div class="col-lg-4">
                <div class="card" style="max-width: 12rem;">
                  <div class="card-header">Calories Burn
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
viewBox="0 0 24 24" fill="currentColor" class="text-orange"> <g> <path fill="none"
d="M0 0h24v24H0z"/> <path d="M12 23a7.5 7.5 0 0 1-5.138-12.963C8.204 8.774 11.5 6.5
 11 1.5c6 4 9 8 3 14 1 0 2.5 0 5-2.47.27.773.5 1.604.5 2.47A7.5 7.5 0 0 1 12 23z"/>
 </g> </svg>


                  </div>
                  <div class="card-body">
                    <h5 class="card-title">{{ current_user.avgCalories }}</h5>
                  </div>
                </div>
                <div class="card" style="max-width: 10rem;">
                  <div class="card-header">Heart rate avg
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
="currentColor" class="bi bi-heart-pulse-fill text-red" viewBox="0 0 16 16">
                      <path fill-rule="evenodd" d="M1.475 9C2.702 10.84 4.779 12.871 8
15c3.221-2.129 5.298-4.16 6.525-6H12a.5.5 0 0 1-.464-.314l-1.457-3.642-1.598 5.593a
.5.5 0 0 1-.945.049L5.889 6.568l-1.473 2.21A.5.5 0 0 1 4 9H1.475ZM.879 8C-2.426
1.68 4.41-2 7.824 1.143c.06.055.119.112.176.171a3.12 3.12 0 0 1 .176-.17C11.59-2
18.426 1.68 15.12 8h-2.783l-1.874-4.686a.5.5 0 0 0-.945.049L7.921 8.956 6.464 5.314
a.5.5 0 0 0-.88-.091L3.732 8H.88Z"/>
                    </svg>

                  </div>
                  <div class="card-body">
                    <h5 class="card-title">{{ current_user.avgHeartRate }}</h5>
                  </div>
                </div>
                <div class="card" style="max-width: 10rem;">
                  <div class="card-header">Time working out (min)
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
="currentColor" class="bi bi-clock" viewBox="0 0 16 16">
                      <path d="M8 3.5a.5.5 0 0 0-1 0V9a.5.5 0 0 0 .252.434l3.5 2a.5.5 0
 0 0 .496-.868L8 8.71V3.5z"/>
                      <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm7-8A7 7 0 1 1 1 8a7
7 0 0 1 14 0z"/>
                    </svg>

                  </div>
                  <div class="card-body">
                    <h5 class="card-title">{{ current_user.totalTime }}</h5>
                  </div>
                </div>
            </div>
          </div>
          <div class="row">
            <div class="col-lg-12">
              <div class="chart-container" style="max-width: 40rem; position: relative;
">
                <canvas id="weightUpdate"></canvas>
              </div>
            </div>
          </div>
```

```
323           </div>
324         </div>
325       </div>
326
327       <div class="card card-body h-auto" style="width: 18rem; float: right;">
328         <div class="card-header">
329         <div class="row align-items-center">
330           <div class="col-8">
331             <h3 class="mb-0">Reports </h3>
332           </div>
333           <div class="col-4 text-right">
334             <a href="{{ url_for('client_blueprint.add_review', field1=reviewFields[1],
    field2=reviewFields[2], field3=reviewFields[3], field4=reviewFields[4] , type="
    COACHING", target=reviewTargetId) }}" class="btn btn-sm btn-secondary">Add new
    review</a>
335           </div>
336         </div>
337       </div>
338         <table class="table">
339             <thead>
340                 <tr>
341                     <th scope="col">Type</th>
342                     <th scope="col">Date</th>
343                     <th scope="col">Details</th>
344                 </tr>
345             </thead>
346
347             {% for review in current_user.reviews %}
348             <tr>
349                 <td>
350                     {{ review.type }}
351                 </td>
352                 <td>
353                     {{ review.date.strftime("%d/%m/%Y, %H:%M:%S") }}
354                 </td>
355                 <td>
356                   <a class="btn btn-sm btn-primary" href="{{ url_for('client_blueprint.
    review', Id=review.id, Type=review.type) }}">Details</a>
357                 </td>
358             </tr>
359             {% endfor %}
360           </table>
361       </div>
362     </div>
363     </div>
364   </div>
365   {%endif%}
366
367 {% endblock content %}
368
369 <!-- Specific JS goes HERE -->
370 {% block javascripts %}
371 {% if current_user.role == 1%}
372 <script>
373
374   var workoutList = []
375   {% for workoutType in current_user.workoutTypeList %}
376     workoutList.push('{{workoutType}}'.replaceAll('&#34;',''));
377   {% endfor %}
378   const ctx = document.getElementById('workoutTypes').getContext('2d');
379   const ctx2 = document.getElementById('workoutCount').getContext('2d');
380   const ctx3 = document.getElementById('weightUpdate').getContext('2d');
381   const weightUpdateChart = new Chart(ctx3,{
382       type: 'line',
383       data:{
384         labels: [
385           'Jan',
386           'Feb',
387           'Mar',
388           'Apr',
389           'Mai',
390           'Jun',
```

```
          'Jul',
          'Aug',
          'Sept',
          'Oct',
          'Nov',
          'Dec'
        ],
        datasets: [{
          label: 'Weight update',
          data: {{current_user.weightUpdate}},
          fill: false,
          borderColor: 'rgb(0, 128, 128)',
          tension: 0.1
        }]
      },
      options: {
        scales: {
            y: {
                min: 40,
                suggestedMin: 50,
                suggestedMax: 100
            }
        },
        elements: {
        line: {
          borderWidth: 3
        }
      }
      }
  });
  const myChart2 = new Chart(ctx2, {
      type: 'line',
      data: {
        labels: [
          'Jan',
          'Feb',
          'Mar',
          'Apr',
          'Mai',
          'Jun',
          'Jul',
          'Aug',
          'Sept',
          'Oct',
          'Nov',
          'Dec'
        ],
        datasets: [{
          label: 'Number of workouts',
          data: {{current_user.nbWorkoutPerMonth}},
          fill: false,
          borderColor: 'rgb(128, 0, 128)',
          tension: 0.1
        }]
      },
      options: {
        r: {
          suggestedMin: 0
        },
        elements: {
        line: {
          borderWidth: 3
        }
      }
      }
    });
  const myChart = new Chart(ctx, {
    type: 'doughnut',
    data: {
      labels: workoutList,
      datasets: [{
        label: 'Stats',
```

```
        data: {{ current_user.workoutTypeCount }},
        backgroundColor: [
          'rgb(255, 99, 132)',
          'rgb(54, 162, 235)',
          'rgb(255, 205, 86)',
          'rgb(128, 0, 128)',
          'rgb(0,128,128)',
          'rgb(128,128,0)'
        ],
        hoverOffset: 4
      }]

    }
  });

  </script>
  {%endif%}
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Login {% endblock title %}

<!-- Specific CSS HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block content %}


  <!-- Page content -->
  <div class="container mt--8 pb-5">
    <div class="row justify-content-center">
      <div class="col-lg-5 col-md-7">
        <div class="card bg-secondary shadow border-0">
          <div class="card-header bg-transparent pb-5">

            <div class="text-muted text-center mt-2 mb-3">
              LOGIN
              <br />
              {% if msg %}
                <span class="text-danger">{{ msg | safe }}</span>
              {% else %}
              <span>
                Add your credentials
              </span>
              {% endif %}
            </div>

          </div>
          {% for category, message in get_flashed_messages(with_categories=true) %}

          <div class="alert alert-{{category}} alert-dismissible fade show" role="alert
    ">
            {{ message }}
            <button type="button" class="btn btn-secondary" data-bs-dismiss="alert"
    aria-label="Close">Close</button>
          </div>

          {% endfor %}
          <div class="card-body px-lg-5 py-lg-5">

            <form role="form" method="post" action="">

              {{ form.hidden_tag() }}

              <div class="form-group mb-3">
                <div class="input-group input-group-alternative">
                  <div class="input-group-prepend">
                    <span class="input-group-text"><i class="ni ni-single-02"></i></
    span>
                  </div>
                  {{ form.email(placeholder="Email",class="form-control") }}
```

```
                </div>
              </div>
              <div class="form-group">
                <div class="input-group input-group-alternative">
                  <div class="input-group-prepend">
                    <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
                  </div>
                  {{ form.password(placeholder="Password",class="form-control",type="password") }}
                </div>
              </div>

              <div class="text-center">
                {{ form.login(class="btn btn-primary my-4") }}
              </div>
            </form>
          </div>
        </div>
        <div class="row mt-3 text-center">
          <div class="col-12">
            <a href="{{ url_for('authentication_blueprint.register') }}" class="text-light"><small>Create new account </small></a>
          </div>
        </div>
      </div>
    </div>
  </div>
  {% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```

```
<nav class="sidenav navbar navbar-vertical  fixed-left  navbar-expand-xs navbar-light bg-white" id="sidenav-main">
    <div class="scrollbar-inner">
      <!-- Brand -->
      <div class="sidenav-header  align-items-center">
        <a class="navbar-brand" href="/">
            <span class="nav-link-text">FitJourney</span>
        </a>
      </div>
      <div class="navbar-inner">
        <!-- Collapse -->
        <div class="collapse navbar-collapse" id="sidenav-collapse-main">
          <!-- Nav items -->
          <ul class="navbar-nav">
            {% if not current_user.is_authenticated %}
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href="{{ url_for('authentication_blueprint.login') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Login</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href="{{ url_for('authentication_blueprint.register') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Register</span>
              </a>
            </li>
            {% elif current_user.is_authenticated and current_user.role == 1 %}
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href="{{ url_for('client_blueprint.profile') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Profile</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'index' in segment %} active {% endif %}" href="
```

```
    /">
                <i class="ni ni-calendar-grid-58 text-orange"></i>
                <span class="nav-link-text">Next Sessions</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'workouts' in segment %} active {% endif %}"
    href="{{ url_for('client_blueprint.workouts') }}">
                <i class="ni ni-planet text-orange"></i>
                <span class="nav-link-text">Workouts</span>
              </a>
            </li>
            {% elif current_user.is_authenticated and current_user.role == 2 %}
            <li class="nav-item">
              <a class="nav-link {% if 'dashboard' in segment %} active {% endif %}"
    href="{{ url_for('coach_blueprint.dashboard') }}">
                <i class="ni ni-tv-2 text-primary"></i>
                <span class="nav-link-text">Dashboard</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
    ="{{ url_for('client_blueprint.profile') }}">
                <i class="ni ni-single-02 text-yellow"></i>
                <span class="nav-link-text">Profile</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link {% if 'calendar' in segment %} active {% endif %}"
    href="{{ url_for('coach_blueprint.calendar') }}">
                <i class="ni ni-calendar-grid-58 text-orange"></i>
                <span class="nav-link-text">Calendar</span>
              </a>
            </li>
            {% endif %}
          </ul>
          <!-- Divider -->
          <hr class="my-3">
          {% if current_user.is_authenticated %}
          <!-- Heading -->
          <ul class="navbar-nav""">
          <li class="nav-item">
            <a class="nav-link" href="{{ url_for('authentication_blueprint.logout') }}"
    >
            <i class="ni ni-user-run text-red"></i><i class="fa fa-sign-out" aria-
    hidden="true"></i>
            <span class="nav-link-text">Logout</span>
          </a>
        </li>
        </ul>
        {% endif %}
        </div>
      </div>
    </div>
  </nav>
```

```
{% extends 'layout/base.html' %}

{% block title %} Renew subscription {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

<div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
  {{ message }}
  <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="
    Close">Close</button>
</div>
```

```
16
17 {% endfor %}
18
19
20
21     <div class="row justify-content-center">
22
23       <div class="col-xl-8 ">
24         <div class="card">
25           <div class="card-header">
26             <div class="row align-items-center">
27               <div class="col-8">
28                 <h3 class="mb-0">Renew subscription </h3>
29               </div>
30               <div class="col-4 text-right">
31
32               </div>
33             </div>
34           </div>
35           <div class="card-body">
36             <form action="{{url_for('coach_blueprint.new_subscription', clientId=client
    .id)}}" method="POST">
37                 {{ form.hidden_tag() }}
38               <div class="pl-lg-4">
39                 <div class="row justify-content-center">
40                     <div class="col-lg-4">
41                         <div class="form-group">
42                           <label class="form-control-label" for="name">Name</label>
43                           {{form.name(placeholder="Name",class="form-control",type="
    text", value=client.name, readonly=readonly)}}
44                         </div>
45                     </div>
46                     <div class="col-lg-4">
47                       <div class="form-group">
48                         <label class="form-control-label" for="surname">Surname</
    label>
49                         {{form.surname(placeholder="Surname",class="form-control",
    type="text", value=client.surname,readonly=readonly)}}
50                         </div>
51                     </div>
52                     <div class="col-lg-4">
53                       <div class="form-group">
54                         <label class="form-control-label" for="email">Email</label>
55                         {{form.email(placeholder="Email", class="form-control", type=
    "email", value=client.email, readonly=readonly)}}
56                       </div>
57                     </div>
58
59               </div>
60                   <div class="row justify-content-center">
61                       <div class="col-lg-4">
62                           <div class="form-group">
63                             <label class="form-control-label" for="start_date">
    Starting date</label>
64                             {{form.start_date(class="form-control", onchange="
    updateEndTime()")}}
65                           </div>
66                         </div>
67                       <div class="col-lg-4">
68                           <div class="form-group">
69                             <label class="form-control-label" for="subscription">
    Subscription type</label>
70                             {{form.subscription(class="form-control", onchange="
    updateEndTime()")}}
71                           </div>
72                         </div>
73
74                         <div class="col-lg-4">
75                           <div class="form-group">
76                             <label class="form-control-label" for="end_date">
    Subscription until</label>
77                             {{form.end_date(class="form-control", readonly=readonly)
```

```
    }}
                            </div>
                          </div>
                    </div>

                <div class="row">
                  <div class="col-md-12">
                    <div class="form-group">
                        {{form.submit(class="form-control btn btn-primary")}}
                    </div>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}
<script>
    function updateEndTime(){
      var subscription = document.getElementById("subscription").value;
      var start_date = document.getElementById("start_date").value;

      if(subscription != "" && start_date != ""){
        //Check if the fields aren't empty, then set a end time depends on values
    selected before
        var dt = new Date(start_date);
        dt.setMonth(dt.getMonth() + parseInt(subscription));
        let [end_date] = dt.toISOString().split('T');
        document.getElementById("end_date").value = end_date
      }

    }
</script>
{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Error 403 {% endblock title %}

<!-- Specific CSS goes HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

    <!-- Page content -->
    <div class="container mt--8 pb-5">
      <div class="row justify-content-center">
        <div class="col-lg-5 col-md-7">
          <div class="card bg-secondary shadow border-0">
            <div class="card-header bg-transparent pb-5">

              <div class="text-muted text-center mt-2 mb-3">
                Oups! Error 403
              </div>

              <div class="text-center mb-4">
                Access denied - Please authenticate <a href="{{ url_for('
    authentication_blueprint.login') }}">Login</a>
                USER : {{ current_user }}
              </div>

            </div>
          </div>
        </div>
      </div>
```

```
    </div>

{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Error 404 {% endblock title %}

<!-- Specific CSS goes HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

    <!-- Page content -->
    <div class="container mt--8 pb-5">
      <div class="row justify-content-center">
        <div class="col-lg-5 col-md-7">
          <div class="card bg-secondary shadow border-0">
            <div class="card-header bg-transparent pb-5">

              <div class="text-muted text-center mt-2 mb-3">
                Oups! Error 404
              </div>

              <div class="text-center mb-4">
                Page not found - <a href="/">Home</a>
              </div>

            </div>
          </div>
        </div>
      </div>
    </div>

{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Error 404 {% endblock title %}

<!-- Specific CSS goes HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

    <!-- Page content -->
    <div class="container mt--8 pb-5">
      <div class="row justify-content-center">
        <div class="col-lg-5 col-md-7">
          <div class="card bg-secondary shadow border-0">
            <div class="card-header bg-transparent pb-5">

              <div class="text-muted text-center mt-2 mb-3">
                Oups! Error 500
              </div>

              <div class="text-center mb-4">
                Server error, please contact support - <a href="/">Home</a>.
              </div>

            </div>
          </div>
        </div>
      </div>
    </div>

```

```
{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Register {% endblock title %}

<!-- Specific CSS HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block content %}


<!-- Page content -->
<div class="container mt--8 pb-5">
    <div class="row justify-content-center">
      <div class="col-lg-5 col-md-7">
        <div class="card bg-secondary shadow border-0">
          <div class="card-header bg-transparent pb-5">

            <div class="text-muted text-center mt-2 mb-3">
              REGISTER
              <br />
              {% if msg and success%}
                <span class="text-success">{{ msg | safe }}</span>
              {% else %}
                Add your credentials
              {% endif %}

              {% for error in form.password.errors %} <p class="text-danger">{{ error | safe }}</p>  {% endfor%}
            </div>

          </div>
          {% for category, message in get_flashed_messages(with_categories=true) %}

          <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
            {{ message }}
            <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
          </div>

          {% endfor %}
          <div class="card-body px-lg-5 py-lg-5">

            {% if not success %}

              <form role="form" method="post" action="">

                {{ form.hidden_tag() }}

                <div class="form-group mb-3">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-single-02"></i></span>
                    </div>
                    {{ form.name(placeholder="Name",class="form-control") }}
                  </div>
                </div>
                <div class="form-group mb-3">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-single-02"></i></span>
                    </div>
                    {{ form.surname(placeholder="Surname",class="form-control") }}
                  </div>
                </div>
```

```
                <div class="form-group mb-3">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-email-83"></i></span>
                    </div>
                    {{ form.email(placeholder="Email",class="form-control") }}
                  </div>
                </div>
                <div class="form-group mb-3">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-single-02"></i></span>
                    </div>
                    {{ form.birthdate(placeholder="Date of Birth",class="form-control") }}
                  </div>
                </div>
                <div class="form-group">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
                    </div>

                    {{ form.password(placeholder="Password",class="form-control") }}
                  </div>
                </div>
                <div class="form-group">
                  <div class="input-group input-group-alternative">
                    <div class="input-group-prepend">
                      <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
                    </div>
                    {{ form.confirm_password(placeholder="Confirm Password",class="form-control") }}
                  </div>
                </div>

                <div class="text-center">
                  {{ form.register(class="btn btn-primary my-4") }}
                </div>
              </form>
            {% endif %}

          </div>
        </div>
        <div class="row mt-3 text-center">
          <div class="col-12">
            <a href="{{ url_for('authentication_blueprint.login') }}" class="text-light"><small>Use existing account</small></a>
          </div>
        </div>
      </div>
    </div>
  </div>

{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Review {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
```

```
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
  </div>

  {% endfor %}

<div class="row">
      <div class="card card-body h-auto" style="width: 18rem; float: right;">
        <h5 class="card-header">Review added by {{client.name}} {{client.surname}}</h5>
        <div class="row justify-content-center">

            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">Date</div>
                    <div class="card-body">
                      <h5 class="card-title">{{review.date}}</h5>
                    </div>
                  </div>
          </div>
          <div class="col-lg-4">

            <div class="card border-light mb-3" style="max-width: 36rem;">
                <div class="card-header">Review on</div>
                <div class="card-body">
                    {% if review.type == 'WORKOUT'  %}
                    <h5 class="card-title">{{target['Type'] + " - " + target['Date'].strftime("%m/%d/%Y, %H:%M:%S") + " - " + target['Duration'].strftime("%X")}}</h5>
                    {% elif review.type == 'COACHING'  %}
                    <h5 class="card-title">{{target['Name'] + " " + target['Surname']}}</h5>
                    {% endif %}
                </div>
              </div>
          </div>
          <div class="col-lg-2">

            <div class="card border-light mb-3" style="max-width: 18rem;">
                <div class="card-header">Type</div>
                <div class="card-body">
                  <h5 class="card-title">{{review.type}}</h5>
                </div>
              </div>
          </div>
        </div>
        <div class="row justify-content-center">
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">{{"Satisfaction" if review['Type'] == 'COACHING' else "Difficulty"}}</div>
                    <div class="card-body">
                      <h5 class="card-title">{{review.Field1}}/10</h5>
                    </div>
                  </div>
            </div>
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">{{"Support" if review['Type'] == 'COACHING' else "Feel"}}</div>
                    <div class="card-body">
                      <h5 class="card-title">{{review.Field2}}/10</h5>
                    </div>
                  </div>
            </div>
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">{{"Disponibility" if review['Type'] == 'COACHING' else "Fatigue (at the end)"}}</div>
                    <div class="card-body">
```

```
                        <h5 class="card-title">{{review.Field3}}/10</h5>
                      </div>
                    </div>
            </div>
            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">{{"Advice quality" if review['Type'] == '
    COACHING' else "Energy (at the start)"}}</div>
                    <div class="card-body">
                      <h5 class="card-title">{{review.Field4}}/10</h5>
                    </div>
                  </div>
            </div>
        </div>
        </div>
      </div>
    </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Workout {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
  {% for category, message in get_flashed_messages(with_categories=true) %}

  <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
    {{ message }}
    <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
    "Close">Close</button>
  </div>

  {% endfor %}

<div class="row">
      <div class="card card-body h-auto justify-content-center" style="width: 18rem;
    float: right;">
        <h5 class="card-header">Review<a href="{{ url_for('client_blueprint.add_review
    ', field1=workoutReview[1], field2=workoutReview[2], field3=workoutReview[3],
    field4=workoutReview[4] , type="WORKOUT", target=workoutDetails.id) }}"><button
    type="button" class="btn btn-dark" style="float:right;">Add review</button></a></h5
    >

        <div class="row justify-content-center">

            <div class="col-lg-2">
                <div class="card border-light mb-3" style="max-width: 18rem;">
                    <div class="card-header">Type</div>
                    <div class="card-body">
                      <h5 class="card-title">{{ workoutDetails.title }}</h5>
                    </div>
                  </div>
          </div>
          <div class="col-lg-4">

            <div class="card border-light mb-3" style="max-width: 36rem;">
                <div class="card-header text-center">Workout</div>
                <div class="card-body">

                </div>
```

```
            </div>
        </div>
        <div class="col-lg-2">

          <div class="card border-light mb-3" style="max-width: 18rem;">
              <div class="card-header"></div>
              <div class="card-body">
                <h5 class="card-title"> <img src="{{ url_for('static', filename='/assets/img/sports/' + workoutDetails.logo)}}" alt="{{workoutDetails.logo}}" class="rounded mx-auto d-block"> </h5>
              </div>
            </div>
        </div>
      </div>
      <div class="row justify-content-center">
          <div class="col-lg-2">
              <div class="card border-light mb-3" style="max-width: 18rem;">
                  <div class="card-header">Heart rate average</div>
                  <div class="card-body">
                    <h5 class="card-title">{{workoutDetails.heart_rate_avg}}
                      <svg xmlns="http://www.w3.org/2000/svg"  style="float:right; color:red;" width="16" height="16" fill="currentColor" class="bi bi-heart-pulse " viewBox="0 0 16 16">
                        <path fill-rule="evenodd" d="m8 2.748-.717-.737C5.6.281 2.514.878 1.4 3.053.918 3.995.78 5.323 1.508 7H.43c-2.128-5.697 4.165-8.83 7.394-5.857.06.055.119.112.176.171a3.12 3.12 0 0 1 .176-.17c3.23-2.974 9.522.159 7.394 5.856h-1.078c.728-1.677.59-3.005.108-3.947C13.486.878 10.4.28 8.717 2.01L8 2.748ZM2.212 10h1.315C4.593 11.183 6.05 12.458 8 13.795c1.949-1.337 3.407-2.612 4.473-3.795h1.315c-1.265 1.566-3.14 3.25-5.788 5-2.648-1.75-4.523-3.434-5.788-5Zm8.252-6.686a.5.5 0 0 0-.945.049L7.921 8.956 6.464 5.314a.5.5 0 0 0-.88-.091L3.732 8H.5a.5.5 0 0 0 0 1H4a.5.5 0 0 0 .416-.223l1.473-2.209 1.647 4.118a.5.5 0 0 0 .945-.049l1.598-5.593 1.457 3.642A.5.5 0 0 0 12 9h3.5a.5.5 0 0 0 0-1h-3.162l-1.874-4.686Z"></path>
                      </svg>
                    </h5>
                  </div>
                </div>
          </div>
          <div class="col-lg-2">
              <div class="card border-light mb-3" style="max-width: 18rem;">
                  <div class="card-header">Calories burned</div>
                  <div class="card-body">
                    <h5 class="card-title">{{ workoutDetails.calories }}
                      <svg xmlns="http://www.w3.org/2000/svg" style="float:right;" width="16" height="16" viewBox="0 0 24 24" fill="currentColor" class="text-orange">
                       <g> <path fill="none" d="M0 0h24v24H0z"/> <path d="M12 23a7.5 7.5 0 0 1-5.138-12.963C8.204 8.774 11.5 6.5 11 1.5c6 4 9 8 3 14 1 0 2.5 0 5-2.47.27.773.5 1.604.5 2.47A7.5 7.5 0 0 1 12 23z"/> </g> </svg>
                    </h5>
                  </div>
                </div>
          </div>
          <div class="col-lg-2">
              <div class="card border-light mb-3" style="max-width: 18rem;">
                  <div class="card-header">Duration</div>
                  <div class="card-body">
                    <h5 class="card-title">{{ workoutDetails.duration }}
                      <svg xmlns="http://www.w3.org/2000/svg" style="float:right;" width="16" height="16" fill="currentColor" class="bi bi-clock" viewBox="0 0 16 16">
                        <path d="M8 3.5a.5.5 0 0 0-1 0V9a.5.5 0 0 0 .252.434l3.5 2a.5.5 0 0 0 .496-.868L8 8.71V3.5z"/>
                        <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 0 16zm7-8A7 7 0 1 1 1 8a7 7 0 0 1 14 0z"/>
                      </svg>
                    </h5>
                  </div>
                </div>
          </div>
          <div class="col-lg-2">
              <div class="card border-light mb-3" style="max-width: 18rem;">
                  <div class="card-header">Date</div>
                  <div class="card-body">
```

```
                        <h5 class="card-title">{{ workoutDetails.date.strftime("%d/%m/%Y, %H:%M:%S") }}
                          <i class="ni ni-calendar-grid-58 text-orange" style="float: right;"></i>
                        </h5>
                      </div>
                    </div>
                </div>
            </div>
        </div>
      </div>
    </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

{% endblock javascripts %}
```

```
{% extends 'layout/base.html' %}

{% block title %} Workouts {% endblock title %}

{% block content %}



<div class="container-fluid mt--8" style="margin: auto;">
    {% for category, message in get_flashed_messages(with_categories=true) %}

    <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
      {{ message }}
      <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label="Close">Close</button>
    </div>

    {% endfor %}
    <div class="row">
      <div class="card card-body h-auto" style="width: 18rem; float: right;">
        <h5 class="card-header">Workouts</h5>
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Type</th>
                    <th scope="col">Date</th>
                    <th scope="col">Duration</th>
                    <th scope="col">Calories Burned</th>
                    <th scope="col">Avg Heart Rate</th>
                    <th scope="col">Details</th>
                </tr>
            </thead>
            {% for workout in workouts %}
            <tr>
                <td>
                    {{ workout.title }}
                </td>
                <td>
                    {{ workout.date.strftime("%d/%m/%Y, %H:%M:%S") }}
                </td>
                <td>
                    {{ workout.duration }}
                </td>
                <td>
                    {{ workout.calories }}
                </td>
                <td>
                    {{ workout.heart_rate_avg }}
                </td>
                <td>
```

```
                    <a class="btn btn-sm btn-primary" href="{{ url_for('
    client_blueprint.workout', Id=workout.id) }}">Details</a>
                </td>
            </tr>
            {% endfor %}
            </table>
      </div>
    </div>



{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}{% endblock javascripts %}
```