

```

1 """
2 Author :      Thomas Fujise
3 Date   :      08.04.2022
4 File   :      __init__.py
5 Version :      1.0.0
6 Brief  :      Application initialization
7 """
8
9 from flask import Flask
10 from flask_login import LoginManager
11 from flask_sqlalchemy import SQLAlchemy
12 from importlib import import_module
13 from flask_mail import Mail
14
15
16
17 db = SQLAlchemy()
18 login_manager = LoginManager()
19 mail = Mail()
20
21 def register_extensions(app):
22     """
23     Initialize extensions
24
25     - LoginManager
26     - SQLAlchemy
27     - FlaskMail
28     """
29     db.init_app(app)
30     login_manager.init_app(app)
31     mail.init_app(app)
32
33
34 def register_blueprints(app):
35     """
36     Register all blueprints
37     """
38     for module_name in ('authentication', 'client', 'coach'):
39         module = import_module('apps.{}.routes'.format(module_name))
40         app.register_blueprint(module.blueprint)
41
42 def config_database(app):
43     """
44     Configure database
45     """
46     @app.before_first_request
47     def initialize_database():
48         db.create_all()
49
50     @app.teardown_request
51     def remove_session(exception=None):
52         db.session.remove()
53
54 def create_app(config):
55     app = Flask(__name__)
56     app.config.from_object(config)
57     register_extensions(app)
58     register_blueprints(app)
59     config_database(app)
60     return app

```

```

1 """
2 Author :      Thomas Fujise
3 Date   :      08.04.2022
4 File   :      config.py
5 Version :      1.0.0
6 Brief  :      Set the different config mode
7 """
8
9 import os
10 from decouple import config
11

```

```

12 class Config(object):
13     """
14     Config parent class
15
16     Contains all base config settings
17     """
18     basedir = os.path.abspath(os.path.dirname(__file__))
19
20     SECRET_KEY = config('SECRET_KEY', default='S3cr3t_k3y')
21
22     #Profile picture
23
24     UPLOAD_FOLDER = config('UPLOAD_FOLDER', default='apps/static/assets/img/profile')
25
26     DEFAULT_PROFILE_PIC = config('DEFAULT_PROFILE_PIC', default='default_profile_pic.png')
27
28     #SQL ALCHEMY
29
30     SQLALCHEMY_DATABASE_URI = '{}://{}:{}_{}@{}:{}_{}'.format(
31         config('DB_ENGINE', default='mysql+pymysql'),
32         config('DB_USERNAME', default='fitjourney_dba'),
33         config('DB_PASS', default='Super2012$'),
34         config('DB_HOST', default='localhost'),
35         config('DB_PORT', default='3306'),
36         config('DB_NAME', default='fitjourney')
37     )
38     SQLALCHEMY_TRACK_MODIFICATIONS = False
39
40     #FLASK MAIL
41     MAIL_SERVER = config('MAIL_SERVER', default='smtp.mailtrap.io')
42     MAIL_PORT = config('MAIL_PORT', default=2525)
43     MAIL_USERNAME = config('MAIL_USERNAME', default='5806e099aa3376')
44     MAIL_PASSWORD = config('MAIL_PASSWORD', default='c2857a008d8b60')
45     MAIL_USE_TLS = config('MAIL_USE_TLS', default=True)
46     MAIL_USE_SSL = config('MAIL_USE_SSL', default=False)
47     MAIL_DEFAULT_SENDER = config('MAIL_DEFAULT_SENDER', default='thomas.fjs@eduge.ch')
48
49 class ProductionConfig(Config):
50     """
51     Production config class
52
53     Contains all config settings for Production
54     """
55     DEBUG = False
56
57     #Security
58     SESSION_COOKIE_HTTPONLY = True
59     REMEMBER_COOKIE_HTTPONLY = True
60     REMEMBER_COOKIE_DURATION = 3600
61
62 class DebugConfig(Config):
63     """
64     Debug config class
65
66     Contains all config settings for Debug
67     """
68     DEBUG = True
69
70
71 config_dict = {
72     'Production': ProductionConfig,
73     'Debug': DebugConfig
74 }

```

```

1 """
2 Author : Thomas Fujise
3 Date : 08.04.2022
4 File : run.py
5 Version : 1.0.0
6 Brief : Load the config mode and run the flask app
7 """

```

```

8
9 from flask_migrate import Migrate
10 from sys import exit
11 from decouple import config
12
13 from apps.config import config_dict
14 from apps import create_app, db
15
16 # By default config set to debug mode
17 DEBUG = config('DEBUG', default=False, cast=bool)
18
19 #Get the config mode
20 config_mode = 'Debug' if DEBUG else 'Production'
21
22 try:
23     app_config = config_dict[config_mode.capitalize()]
24
25 except KeyError:
26     exit('Error: Invalid config mode')
27
28 app = create_app(app_config)
29 Migrate(app,db)
30
31 if DEBUG:
32     app.logger.info('DEBUG = ' + str(DEBUG))
33     app.logger.info('Environment = ' + config_mode)
34     app.logger.info('DBMS = ' + app_config.SQLALCHEMY_DATABASE_URI)
35
36 if __name__ == "__main__":
37     app.run()

```

```

1 """
2 Author : Thomas Fujise
3 Date : 08.04.2022
4 File : __init__.py
5 Version : 1.0.0
6 Brief : Authentication blueprint initialization
7 """
8
9 from flask import Blueprint
10
11
12 blueprint = Blueprint(
13     'authentication_blueprint',
14     __name__,
15     url_prefix='',
16 )

```

```

1 """
2 Author : Thomas Fujise
3 Date : 12.04.2022
4 File : forms.py
5 Version : 1.0.0
6 Brief : Authentication forms
7 """
8
9 from flask_wtf import FlaskForm
10 from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
    BooleanField, ValidationError
11 from wtforms.validators import Email, DataRequired, EqualTo, Length
12 from wtforms.fields.html5 import DateField, EmailField
13
14
15 class LoginForm(FlaskForm):
16     """
17     Create a login form
18     """
19     email = StringField('email', id='email_login', validators=[DataRequired()])
20     password = PasswordField('Password', id='pwd_login', validators=[DataRequired()])
21     login = SubmitField("login")
22
23

```

```

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=[
38         DataRequired(), EqualTo('confirm_password', message='Passwords Must Match!')])
39
40     confirm_password = PasswordField('Confirm Password', id='password2_register')
41
42     register = SubmitField("register")
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")
70
71 """
72 Author : Thomas Fujise
73 Date : 11.04.2022
74 File : models.py
75 Version : 1.0.0
76 Brief : All models needed by the application
77 """
78
79 from flask_login import UserMixin
80 from datetime import datetime
81 from apps import db, login_manager
82
83 from apps.authentication.util import hash_pass
84
85 ACCESS = {
86     'Guest': 0,
87     'Client': 1,
88     'Coach': 2
89 }
90
91 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
44             value
45             # when **kwargs is request.form, some values will be a 1-element list
46             if hasattr(value, '__iter__') and not isinstance(value, str):
47                 value = value[0]
48
49             if property == 'password':
50                 value = hash_pass(value) # we need bytes here (not plain str)
51
52             setattr(self, property, value) # set the attribute
53
54     def __repr__(self):
55         return str(self.email)
56
57     def is_authenticated(self):
58         return True
59
60     def is_active(self):
61         return True
62
63     def is_anonymous(self):
64         return False
65
66     def get_id(self):
67         return str(self.id)
68
69     def is_coach(self):
70         return self.role == ACCESS['Coach']
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)

```

```

92     body_age = db.Column(db.Integer, nullable=False)
93     date = db.Column(db.Date, nullable=False)
94     front_photo = db.Column(db.String(250), nullable=True)
95     right_side_photo = db.Column(db.String(250), nullable=True)
96     left_side_photo = db.Column(db.String(250), nullable=True)
97     back_photo = db.Column(db.String(250), nullable=True)
98
99
100 class Role(db.Model):
101     __tablename__ = 'ROLE'
102
103     id = db.Column(db.Integer, primary_key=True)
104     name = db.Column(db.String(32))
105
106 class Workout(db.Model):
107     __tablename__ = 'WORKOUT'
108
109     id = db.Column(db.Integer, primary_key=True)
110     workout_type = db.Column(db.Integer, db.ForeignKey('WORKOUT_TYPE.id'))
111     client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
112     date = db.Column(db.DateTime)
113     duration = db.Column(db.Time)
114     heart_rate_max = db.Column(db.Numeric)
115     heart_rate_min = db.Column(db.Numeric)
116     heart_rate_avg = db.Column(db.Numeric)
117     calories = db.Column(db.Numeric)
118     active_calories = db.Column(db.Numeric)
119     distance = db.Column(db.Numeric, nullable=True)
120     pace_avg = db.Column(db.Time, nullable=True)
121
122 class WorkoutType(db.Model):
123     __tablename__ = 'WORKOUT_TYPE'
124
125     id = db.Column(db.Integer, primary_key=True)
126     title = db.Column(db.String(32))
127     description = db.Column(db.String(250))
128     logo = db.Column(db.String(250))
129
130 class Program(db.Model):
131     __tablename__ = 'PROGRAM'
132
133     id = db.Column(db.Integer, primary_key=True)
134     type = db.Column(db.String(32))
135     pdf = db.Column(db.LargeBinary(length=(2**32)-1))
136     date = db.Column(db.Date)
137     client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
138     coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
139
140 class Session(db.Model):
141     __tablename__ = 'SESSION'
142
143     id = db.Column(db.Integer, primary_key=True)
144     start_time = db.Column(db.DateTime, nullable=False)
145     end_time = db.Column(db.DateTime, nullable=False)
146     duration = db.Column(db.Time, nullable=False)
147     workout_type = db.Column(db.Integer, db.ForeignKey('WORKOUT_TYPE.id'), nullable=False)
148     client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
149     coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
150
151 class Subscription(db.Model):
152     __tablename__ = 'SUBSCRIPTION'
153
154     id = db.Column(db.Integer, primary_key=True)
155     title = db.Column(db.String(64))
156     cost = db.Column(db.Integer)
157     duration = db.Column(db.Integer, unique=True)
158
159 class Purchase(db.Model):
160     __tablename__ = 'PURCHASE'
161
162     id = db.Column(db.Integer, primary_key=True)

```

```

163     client_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
164     date = db.Column(db.Date)
165     subscription_id = db.Column(db.Integer, db.ForeignKey('SUBSCRIPTION.id'))
166
167 class CoachedBy(db.Model):
168     __tablename__ = 'COACHEDBY'
169
170     client_id = db.Column(db.Integer, db.ForeignKey('USER.id'), primary_key=True)
171     coach_id = db.Column(db.Integer, db.ForeignKey('USER.id'), primary_key=True)
172     starting_date = db.Column(db.Date)
173     end_date = db.Column(db.Date, nullable=True)
174
175 class CoachingReview(db.Model):
176     __tablename__ = 'COACHING_REVIEW'
177
178     id = db.Column(db.Integer, db.ForeignKey('REVIEW.id'), primary_key=True)
179     satisfaction = db.Column(db.Integer, nullable=False)
180     support = db.Column(db.Integer, nullable=False)
181     disponibility = db.Column(db.Integer, nullable=False)
182     advice = db.Column(db.Integer, nullable=False)
183     target_id = db.Column(db.Integer, db.ForeignKey('USER.id'))
184
185 class WorkoutReview(db.Model):
186     __tablename__ = 'WORKOUT_REVIEW'
187
188     id = db.Column(db.Integer, db.ForeignKey('REVIEW.id'), primary_key=True)
189     difficulty = db.Column(db.Integer, nullable=False)
190     feel = db.Column(db.Integer, nullable=False)
191     fatigue = db.Column(db.Integer, nullable=False)
192     energy = db.Column(db.Integer, nullable=False)
193     target_id = db.Column(db.Integer, db.ForeignKey('WORKOUT.id'))
194
195 class Review(db.Model):
196     __tablename__ = 'REVIEW'
197
198     id = db.Column(db.Integer, primary_key=True)
199     comment = db.Column(db.String(250), nullable=True)
200     date = db.Column(db.DateTime)
201     type = db.Column(db.String(15), nullable=False)
202     id_client = db.Column(db.Integer, db.ForeignKey('USER.id'))
203
204 @login_manager.user_loader
205 def user_loader(id):
206     return User.query.filter_by(id=id).first()
207
208
209 @login_manager.request_loader
210 def request_loader(request):
211     email = request.form.get('email')
212     user = User.query.filter_by(email=email).first()
213     return user if user else None

```

```

1 """
2 Author : Thomas Fujise
3 Date : 11.04.2022
4 File : routes.py
5 Version : 1.0.0
6 Brief : Set all authentication routes
7 """
8
9 from flask import render_template, redirect, request, url_for, flash
10 from flask_login import (
11     current_user,
12     login_user,
13     logout_user
14 )
15 from flask_login import login_required
16
17 from apps import db, login_manager
18 from apps.authentication import blueprint
19 from apps.authentication.forms import LoginForm, RegisterForm
20 from apps.authentication.models import User

```

```

21 from apps.authentication.util import verify_pass
22
23
24
25 #Default route
26 @blueprint.route('/')
27 def route_default():
28     return redirect(url_for('authentication_blueprint.login'))
29
30
31 #Login route
32 @blueprint.route('/login', methods=['GET', 'POST'])
33 def login():
34     """
35     Display login page with the login form
36     """
37     login_form = LoginForm()
38     if login_form.validate_on_submit():
39         #read form data
40         email = request.form['email']
41         password = request.form['password']
42
43         #Locate the user
44         user = User.query.filter_by(email=email).first()
45         #Check password
46         if user and verify_pass(password, user.password):
47             login_user(user)
48             return redirect(url_for('authentication_blueprint.route_default'))
49
50         # Something is not ok (user or password)
51         flash('Wrong user or password', 'danger')
52         return redirect(url_for('authentication_blueprint.route_default', msg='Wrong
user or password'))
53
54     if not current_user.is_authenticated:
55
56         return render_template('accounts/login.html', form=login_form)
57
58     if current_user.role == 1:
59         #CLIENT
60         return redirect(url_for('client_blueprint.index'))
61     elif current_user.role == 2:
62         #COACH
63         return redirect(url_for('coach_blueprint.dashboard'))
64
65     return redirect(url_for('client_blueprint.index'))
66
67
68 #Register route
69 @blueprint.route('/register', methods=['GET', 'POST'])
70 def register():
71     """
72     Display Register page with the register form
73     """
74     register_form = RegisterForm()
75
76     if register_form.validate_on_submit():
77
78         email = request.form['email']
79         #name = request.form['name']
80
81         #Check if user already exists
82         user = User.query.filter_by(email=email).first()
83
84         if user:
85             flash("Email already registered", 'warning')
86             return render_template('accounts/register.html', msg='Email Already
registered', success=False, form=register_form)
87
88         #try to create the user
89         try:
90

```



```

91         user = User(name=request.form['name'], surname=request.form['surname'],
92 email=request.form['email'], birthdate=request.form['birthdate'], password=request.
93 form['password'], role=2)
94         db.session.add(user)
95         db.session.commit()
96         flash("User has been created please login", 'success')
97         return redirect(url_for('authentication_blueprint.login'))
98     except:
99         db.session.rollback()
100         flash("Error while registering new user", 'danger')
101         return redirect(url_for('authentication_blueprint.register'))
102
103     return render_template('accounts/register.html', msg='<a href="/login"> LOGIN</
104 a> ', success=True, form=register_form)
105
106     else:
107         return render_template('accounts/register.html', form=register_form)
108
109 @blueprint.route('/logout')
110 @login_required
111 def logout():
112
113     logout_user()
114
115     return redirect(url_for('authentication_blueprint.login'))
116
117 # Errors
118
119 @login_manager.unauthorized_handler
120 def unauthorized_handler():
121     return render_template('errors/page-403.html'), 403
122
123
124 @blueprint.errorhandler(403)
125 def access_forbidden(error):
126     return render_template('errors/page-403.html'), 403
127
128
129 @blueprint.errorhandler(404)
130 def not_found_error(error):
131     return render_template('errors/page-404.html'), 404
132
133
134 @blueprint.errorhandler(500)
135 def internal_error(error):
136     return render_template('errors/page-500.html'), 500

```

```

1 """
2 Author :      Thomas Fujise
3 Date   :      12.04.2022
4 File   :      util.py
5 Version :      1.0.0
6 Brief   :      Util functions file for authentication
7 """
8
9 import os
10 import hashlib
11 import binascii
12
13 def hash_pass(password):
14     """
15     Function that hash a password for storing
16     Encodes a provided password in a way that is safe to store on a database
17
18     Parameter(s) :
19     NAME         | TYPE   | DESC
20     password    | STRING | The password to hash
21
22     Return :

```

```

23 | BYTES | The password hashed with salt
24
25 """
26 salt = hashlib.sha256(os.urandom(60)).hexdigest().encode('ascii')
27 pdhash = hashlib.pbkdf2_hmac('sha512', password.encode('utf-8'), salt, 100000)
28 pdhash = binascii.hexlify(pdhash)
29
30 return (salt + pdhash) # return bytes
31
32
33 def verify_pass(provided_password, stored_password):
34     """
35     Function that verify a stored password against one provided by the user
36     Given an encoded password and a plain text one which is provided by the user, it
37     verifies whether the provided password matches the encoded one.
38
39     Parameter(s) :
40     NAME          | TYPE   | DESC
41     provided_password | STRING | The password provided by the user
42     stored_password  | BYTES  | The hashed with salt password stored in db
43
44     Return :
45     | BOOLEAN | True if passwords are the same, else False
46
47     """
48     stored_password = stored_password.decode('ascii')
49     salt = stored_password[:64]
50     stored_password = stored_password[64:]
51     pdhash = hashlib.pbkdf2_hmac('sha512', provided_password.encode('utf-8'), salt.
52     encode('ascii'), 100000)
53     pdhash = binascii.hexlify(pdhash).decode('ascii')
54
55     return pdhash == stored_password

```

```

1 """
2 Author :      Thomas Fujise
3 Date   :      08.04.2022
4 File   :      __init__.py
5 Version :     1.0.0
6 Brief  :      Coach blueprint initialization
7 """
8
9 from flask import Blueprint
10
11 blueprint = Blueprint(
12     'coach_blueprint',
13     __name__,
14     url_prefix='',
15 )

```

```

1 """
2 Author :      Thomas Fujise
3 Date   :      24.05.2022
4 File   :      forms.py
5 Version :     1.0.0
6 Brief  :      Coach forms
7 """
8
9 from flask_wtf import FlaskForm
10 from flask_wtf.file import FileField, FileAllowed, FileRequired
11 from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
12 BooleanField, ValidationError, SelectField, TextAreaField, HiddenField,
13 DecimalField
14 from wtforms.validators import Email, DataRequired, EqualTo, Length, Regexp
15 from wtforms.fields.html5 import DateField, EmailField, IntegerRangeField, TimeField
16 from wtforms.ext.dateutil.fields import DateTimeField
17 from datetime import datetime
18
19 class SessionForm(FlaskForm):
20     """

```

```

19 Create an adding session form
20 """
21
22 client = SelectField('Clients', id='clients_list', coerce=int)
23
24 date = DateField('Date', id='datepick', validators=[DataRequired()])
25
26 start_time = TimeField('Start Time', id="start_time", validators=[DataRequired()])
27
28 end_time = TimeField('End Time', id="end_time", validators=[DataRequired()])
29
30 duration = SelectField('Duration', id="duration", coerce=int, validators=[
DataRequired()])
31
32 type = SelectField('Type', id="type_picker", coerce=int)
33
34 add = SubmitField("Add")
35
36 class AddClientForm(FlaskForm):
37     """
38     Create an adding client form
39     """
40
41     name = StringField('Name', id="name", validators=[DataRequired()])
42
43     surname = StringField('Surname', id="surname", validators=[DataRequired()])
44
45     email = EmailField('Email', id="email", validators=[DataRequired()])
46
47     birthdate = DateField('Birthdate', id="birthdate", validators=[DataRequired()])
48
49     height = IntegerField('Height', id="height", validators=[DataRequired()])
50
51     start_date = DateField('Start date', id="start_date", validators=[DataRequired()])
52
53     subscription = SelectField('Subscription', id="subscription", validators=[
DataRequired()])
54
55     end_date = DateField('End date', id="end_date", validators=[DataRequired()])
56
57     add = SubmitField("Add")
58
59 class RenewSubscriptionForm(FlaskForm):
60     """
61     Create a renew subscription form
62     """
63
64     name = StringField('Name', id="name", validators=[DataRequired()])
65
66     surname = StringField('Surname', id="surname", validators=[DataRequired()])
67
68     email = EmailField('Email', id="email", validators=[DataRequired()])
69
70     start_date = DateField('Start date', id="start_date", validators=[DataRequired()])
71
72     end_date = DateField('End date', id="end_date", validators=[DataRequired()])
73
74     subscription = SelectField('Subscription', id="subscription", validators=[
DataRequired()])
75
76     submit = SubmitField("Submit")
77
78 class ClientForm(FlaskForm):
79     """
80     Create a update form
81     """
82
83     name = StringField('Name', id='name', validators=[DataRequired()])
84
85     surname = StringField('Surname', id='surname', validators=[DataRequired()])
86
87     email = EmailField('Email', id="email", validators=[DataRequired(), Email()])

```

```

88     birthdate = DateField('Birthdate', id="birthdate", validators=[DataRequired()])
89
90     profile_pic = FileField('Profile Pic', id="profile_pic")
91
92     card_id = StringField('Card ID', id='card_update')
93
94     weight = StringField('Weight', id='weight_update')
95
96     height = StringField('Height', id='height_update')
97
98     subscriptionEnd = DateField('Subscription Until', id='subscription_until')
99
100     address = StringField('Address', id='address')
101
102     country = StringField('Country', id='country')
103
104     city = StringField('City', id='city')
105
106     npa = StringField('NPA', id='npa')
107
108     register_date = DateField('Register date', id='register_date')
109
110     change = SubmitField("Change")
111
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
120                       = [("Diet", "Diet"), ("Workout", "Workout")])
121
122     file = FileField('Program File', id="program_file", validators=[FileRequired(),
123                           FileAllowed(['pdf'], 'PDF only !')])
124
125     client = HiddenField('Client', id="program_client")
126
127     add = SubmitField("Upload")
128
129 class AddCheckUpForm(FlaskForm):
130     """
131     Create the check up form
132     """
133
134     name = StringField('Name', id='name', validators=[DataRequired()])
135
136     surname = StringField('Surname', id='surname', validators=[DataRequired()])
137
138     height = IntegerField('Height', id='height', validators=[DataRequired()])
139
140     age = IntegerField('Age', id='age', validators=[DataRequired()])
141
142     weight = DecimalField('Weight', id='weight', validators=[DataRequired()])
143
144     bmi = DecimalField('BMI', id='bmi', validators=[DataRequired()])
145
146     water = DecimalField('Water %', id='water', validators=[DataRequired()])
147
148     protein = DecimalField('Protein %', id='protein', validators=[DataRequired()])
149
150     muscle_mass_percent = DecimalField('Muscle mass %', id='muscle_mass_percent',
151                                       validators=[DataRequired()])
152
153     body_fat_percent = DecimalField('Body Fat %', id='body_fat_percent', validators=[
154                                       DataRequired()])
155
156     bone_mass_percent = DecimalField('Bone Mass %', id='bone_mass_percent', validators
157                                     = [DataRequired()])
158
159     bmr = DecimalField('BMR', id='bmr', validators=[DataRequired()])

```

```

155 muscle_mass = DecimalField('Muscle mass', id='muscle_mass', validators=[
156     DataRequired()])
157
158 body_fat = DecimalField('Body fat', id='body_fat', validators=[DataRequired()])
159
160 fat_visceral = DecimalField('Fat Visceral', id='fat_visceral', validators=[
161     DataRequired()])
162
163 bone_mass = DecimalField('Bone Mass', id='bone_mass', validators=[DataRequired()])
164
165 lean_body_mass = DecimalField('Lean body mass', id='lean_body_mass', validators=[
166     DataRequired()])
167
168 body_age = DecimalField('Body age', id='body_age', validators=[DataRequired()])
169
170 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
13 , WorkoutReview, Review, Workout, WorkoutType, Session, Program, CoachedBy,
14 Purchase
15 from apps.coach.forms import SessionForm, AddClientForm, ClientForm, AddProgramForm,
16 AddCheckUpForm, RenewSubscriptionForm
17 from apps.config import Config
18
19 # FLASK
20 from flask import render_template, redirect, request, url_for, flash, send_file

```

```

18 from flask_login import login_required
19 from jinja2 import TemplateNotFound
20 from flask_mail import Message
21 from flask_login import (
22     current_user
23 )
24
25 # UTILS
26 from io import BytesIO
27 import uuid as uuid
28 import os
29 from werkzeug.utils import secure_filename
30 from datetime import date, datetime
31
32 from apps.coach.util import *
33 from apps.client.util import *
34 from apps.coach.reader import get_card_id
35
36 @blueprint.route('/dashboard')
37 @login_required
38 def dashboard():
39     #Check if user is coach
40     if not current_user.is_coach():
41         return redirect(url_for('authentication_blueprint.login'))
42
43     nextClient = get_coach_next_session(current_user.id)
44     clientLastWorkout = get_last_workout(nextClient.id) if nextClient != None else None
45
46     clients = get_clients(current_user.id)
47     return render_template('coach/dashboard.html', segment='coach_dashboard',
48                             nextClient=nextClient, clientLastWorkout=clientLastWorkout, clients=clients)
49
50 @blueprint.route('/calendar', methods=['POST', 'GET'])
51 @login_required
52 def calendar():
53     #Check if user is coach
54     if not current_user.is_coach():
55         return redirect(url_for('authentication_blueprint.login'))
56
57     form = SessionForm(request.form)
58     # Set select input choices
59     form.client.choices = [(client['id'], str(client['name'] + " " + client['surname']))
60                             for client in get_clients(current_user.id)]
61     form.duration.choices = [(i,i) for i in range(1,4)]
62     form.type.choices = [(type.id, type.title) for type in get_all_workout_types()]
63
64     sessions = get_sessions(current_user.id)
65     #print(sessions)
66
67     if request.method == 'POST':
68         #Check if client there is client in the select field
69         if 'client' not in request.form:
70             flash("You don't have client to select, please add a client before", '
71             danger')
72             return redirect( url_for('coach_blueprint.calendar') )
73
74         startDate = request.form['date']
75         startTime = request.form['start_time']
76         endTime = request.form['end_time']
77         durationSelected = request.form['duration']
78
79         # Set datetime value
80         startDateTime = datetime.strptime(startDate + " " + startTime, '%Y-%m-%d %H:%M')
81
82         endDateTime = datetime.strptime(startDate + " " + endTime, '%Y-%m-%d %H:%M:%S')
83
84         # Set duration format
85         duration = "0{durationSelected}:00:00".format(durationSelected=durationSelected)
86
87         client = get_client_details(request.form['client'])
88         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 + " - " +
154     str(subscription.cost) + "CHF") for subscription in get_all_subscriptions()]
155
156     if request.method == 'POST':
157         height = request.form['height']
158         surname = request.form['surname']
159         email = request.form['email']
160         name = request.form['name']
161         birthdate = request.form['birthdate']
162         date = request.form['start_date']
163         end_date = request.form['end_date']
164         subscription = request.form['subscription']
165
166         try:
167             #Add client
168             newClient = User(name=name, surname=surname, email=email, birthdate=
169             birthdate, profile_pic=Config.DEFAULT_PROFILE_PIC, role=1, password=name+"123")
170             db.session.add(newClient)
171             db.session.flush()
172             #Assign coach and add purchase of the subscription selected
173             newCoachAssign = CoachedBy(client_id=newClient.id, coach_id=current_user.id
174             , starting_date=date, end_date=end_date)
175             newPurchase = Purchase(client_id=newClient.id, date=date, subscription_id=
176             get_subscription_id(subscription).id)
177
178             db.session.add(newPurchase)
179             db.session.add(newCoachAssign)
180             db.session.commit()
181             flash("New client is registered !", 'success')
182             #Set and send mail
183             msg = Message('Your account is now created ', recipients =[newClient.email
184             ])
185             msg.body = "Hey " + newClient.name + ", your Fitjourney account just been
186             created. You can access to it by filling the login form with : \r\n Email : " +
187             newClient.email + " \r\n Password : " + newClient.name + "123 \r\n Thank you. \r\n
188             Fitjourney"
189             mail.send(msg)
190             return redirect( url_for('coach_blueprint.dashboard') )
191         except:
192             db.session.rollback()
193             flash("Error, while trying to register new client, please try again", '
194             danger')
195             return redirect(url_for('coach_blueprint.dashboard'))
196
197     return render_template('coach/add_client.html', segment='add_client', form=form)
198
199 @blueprint.route('/add_program', methods=['POST', 'GET'])
200 @login_required
201 def add_program():
202     #Check if user is coach
203     if not current_user.is_coach():
204         return redirect(url_for('authentication_blueprint.login'))
205
206     form=AddProgramForm()
207     #Get client id
208     client_id = request.args.get('clientId')
209
210     if request.method == "POST":
211         if form.validate_on_submit():
212
213             file_name = form.file.data
214             client = get_client_details(request.form['client'])
215             try:
216                 newProgram = Program(type=request.form['type'], pdf=file_name.read(),

```



```

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

```

```

267         db.session.commit()
268         flash("Check up added !", 'success')
269         return redirect(url_for('coach_blueprint.client', clientId=client_id))
270     except:
271         flash("Error while trying to add check up", 'danger')
272         return redirect(url_for('coach_blueprint.client', clientId=client_id))
273
274     return render_template('coach/add_checkup.html', segment='add_checkup', form=form,
275                           infos=static_infos)
276
277 @blueprint.route('/new_subscription', methods=['POST', 'GET'])
278 @login_required
279 def new_subscription():
280     #Check if user is coach
281     if not current_user.is_coach():
282         return redirect(url_for('authentication_blueprint.login'))
283
284     form=RenewSubscriptionForm(request.form)
285
286     #Set the subscription choices
287     form.subscription.choices = [(subscription.duration, subscription.title + " - " +
288     str(subscription.cost) + "CHF") for subscription in get_all_subscriptions()]
289
290     client_id = request.args.get('clientId')
291     # Get the client details
292     client = get_client_details(client_id)
293     if request.method == "POST":
294         if form.validate_on_submit():
295             try:
296                 print(get_subscription_id(request.form['subscription']).id)
297                 newCoachAssign = CoachedBy(client_id=client_id, coach_id=current_user.
298                 id, starting_date=request.form['start_date'], end_date=request.form['end_date'])
299                 newPurchase = Purchase(client_id=client_id, date=request.form['
300                 start_date'], subscription_id=get_subscription_id(request.form['subscription']).id)
301                 print("OK")
302                 db.session.add(newPurchase)
303                 db.session.add(newCoachAssign)
304                 db.session.commit()
305                 flash("New subscription is purchased !", 'success')
306                 msg = Message('New subscription purchased', recipients =[client.email])
307                 msg.body = "Hey " + client.name + ", you just purchased a new
308                 subscription for " + request.form['subscription'] + " month(s). \r\n Thank you. \r
309                 \n Fitjourney"
310                 mail.send(msg)
311                 return redirect( url_for('coach_blueprint.client', clientId=client_id)
312                 )
313             except:
314                 db.session.rollback()
315                 flash("Error, while trying to purchase new subscription", 'danger')
316                 return redirect(url_for('coach_blueprint.client', clientId=client_id))
317
318     return render_template('coach/new_subscription.html', segment='new_subscription',
319                           form=form, client=client)
320
321 @blueprint.route('/new_card', methods=['GET'])
322 @login_required
323 def new_card():
324     #Check if user is coach
325     if not current_user.is_coach():
326         return redirect(url_for('authentication_blueprint.login'))
327
328     client_id = request.args.get('clientId')
329     card = get_card_id()
330     client = get_client_details(client_id)
331     if card == None:
332         flash("No new card was detected", 'danger')
333         return redirect(url_for('coach_blueprint.client', clientId=client_id))
334
335     if update_card_id(client_id, card):
336         flash("Member card updated", 'success')
337     else:
338         flash("Error while updating the member card, please try again", 'danger')

```

```

331         return redirect(url_for('coach_blueprint.client', clientId=client_id))
332
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 (
35     date)
36     """
37     lastWorkout = db.session.query(Workout.date).filter(Workout.client_id==userId).
38     order_by(Workout.date.desc()).first()
39
40     return lastWorkout
41
42 def get_coach_next_session(coachId):
43     """
44     Get the next session of a coach
45
46     Parameter(s):
47     NAME      |  TYPE  |  DESC
48     coachId   |  INT   |  the id of the coach
49
50     Return :
51     | 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     return result
110
111
112 def get_client_details(clientId):

```

```

113     """
114     Get client details with his id
115
116     Parameter(s):
117         NAME      | TYPE  | DESC
118         clientId | INT   | the id of the client
119
120     Return :
121         | Dict | Dict, contains all properties (details) from user that we need
122     """
123
124     client = db.session.query(User.id, User.name, User.surname, User.email, User.
125         birthdate, User.profile_pic, User.created_at, User.card_id, User.address, User.city
126         , User.country, User.npa).filter(User.id==clientId).first()
127
128     return client
129
130 def get_all_workout_types():
131     """
132     Get all types of workouts
133
134     Parameter(s):
135         NAME      | TYPE  | DESC
136         /          | /      | /
137
138     Return :
139         | Array[Query()] | Array of query object with all properties selected
140     """
141
142     types = db.session.query(WorkoutType.id, WorkoutType.title).order_by(WorkoutType.
143         title.asc())
144     return types
145
146 def get_sessions(coachId):
147     """
148     Get all sessions as event and create array to be used with FullCalendar.io
149
150     Parameter(s):
151         NAME      | TYPE  | DESC
152         coachId   | INT   | the id of the coach
153
154     Return :
155         | Array[Obj()] | Array of object contains properties required to be an event with
156         FullCalendar (title, start, end)
157     """
158
159     sessions = db.session.query(User.name, User.surname, Session.start_time, Session.
160         end_time).join(User, Session.client_id==User.id).filter(Session.coach_id==coachId)
161
162     result = []
163
164     for session in sessions:
165         result.append(
166             {
167                 'title': session.name + " " + session.surname,
168                 'start' : session.start_time.strftime("%Y-%m-%dT%H:%M:%S"),
169                 'end'   : session.end_time.strftime("%Y-%m-%dT%H:%M:%S")
170             })
171
172     return result
173
174 def get_all_subscriptions():
175     """
176     Get all types of subscriptions available
177
178     Parameter(s):
179         NAME      | TYPE  | DESC
180         /          | /      | /

```

```

180     Return :
181     | Array[Query()] | Array of query object with all subscription properties selected
182     """
183
184     subscription = db.session.query(Subscription.id, Subscription.title, Subscription.
185     cost, Subscription.duration).order_by(Subscription.title.asc())
186
187     return subscription
188
189 def get_subscription_id(duration):
190     """
191     Get the id of a subscription by his duration
192
193     Parameter(s):
194     NAME      | TYPE | DESC
195     duration | STRING | the duration of the subscription
196
197     Return :
198     | Query | Array of object contains properties required to be an event with
199     FullCalendar (title, start, end)
200     """
201     id = db.session.query(Subscription.id).filter(Subscription.duration==duration).
202     first()
203
204     return id
205
206 def get_program(clientId):
207     """
208     Get the actual program of a client
209
210     Parameter(s):
211     NAME      | TYPE | DESC
212     clientId | INT  | the id of the client
213
214     Return :
215     | Array[Query()] | Array with 2 Query object 1 represent the latest workout program
216     and the other the latest diet program.
217     """
218     # Get the latest workout program uploaded for this user (Program type : 1 => Diet |
219     2 => Workout)
220     workoutProgram = db.session.query(Program.id, Program.date, User.name, User.surname
221     ).join(User, Program.coach_id==User.id).filter(Program.type=="Workout").filter(
222     Program.client_id==clientId).order_by(Program.date.desc()).first()
223     dietProgram = db.session.query(Program.id, Program.date, User.name, User.surname).
224     join(User, Program.coach_id==User.id).filter(Program.type=="Diet").filter(Program.
225     client_id==clientId).order_by(Program.date.desc()).first()
226
227     result = []
228
229     result.append(workoutProgram)
230     result.append(dietProgram)
231
232     return result
233
234 def get_program_by_id(programId):
235     """
236     Get a specific program
237
238     Parameter(s):
239     NAME      | TYPE | DESC
240     programId | INT  | the id of the program
241
242     Return :
243     | Program | Program object with the selected properties
244     """
245     program = db.session.query(Program.date, Program.type, Program.pdf).filter(Program.
246     id==programId).first()
247     return program

```

```

242 def update_card_id(clientId, cardId):
243     """
244     Update the card id of a client
245
246     Parameter(s):
247     NAME      | TYPE  | DESC
248     clientId | INT   | the id of the client
249     cardId   | INT   | the new card id
250
251     Return :
252     | Boolean | True if the update success, else False
253     """
254     try:
255         db.session.query(User).filter(User.id==clientId).update({'card_id':cardId})
256         db.session.commit()
257         return True
258     except:
259         return False
260
261 def cancel_last_subscription(clientId):
262     """
263     Cancel the last subscription purchased for a user
264
265     Parameter(s):
266     NAME      | TYPE  | DESC
267     clientId | INT   | the id of the client
268
269     Return :
270     | Boolean | True if the delete success, else False
271     """
272     try:
273         #Set the last subscription
274         last_subscription = db.session.query(Purchase).filter(Purchase.client_id==
clientId).order_by(Purchase.date.desc()).first()
275         db.session.query(Purchase).filter(Purchase.id==last_subscription.id).delete()
276
277         db.session.commit()
278         return True
279     except:
280         return False

```

```

1  """
2  Author   :      Thomas Fujise
3  Date    :      08.04.2022
4  File    :      __init__.py
5  Version :      1.0.0
6  Brief   :      Client blueprint initialization
7  """
8
9  from flask import Blueprint
10
11  blueprint = Blueprint(
12      'client_blueprint',
13      __name__,
14      url_prefix='',
15  )

```

```

1  """
2  Author   :      Thomas Fujise
3  Date    :      12.04.2022
4  File    :      forms.py
5  Version :      1.0.0
6  Brief   :      Client forms
7  """
8
9  from flask_wtf import FlaskForm
10 from wtforms import FileField, StringField, SubmitField, PasswordField, IntegerField,
BooleanField, ValidationError, SelectField, TextAreaField, HiddenField,
PasswordField
11 from wtforms.validators import Email, DataRequired, EqualTo, Length
12 from wtforms.fields.html5 import DateField, EmailField, IntegerRangeField
13

```

```

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 """

```



```

2 Author : Thomas Fujise
3 Date : 11.04.2022
4 File : routes.py
5 Version : 1.0.0
6 Brief : Set all the client routes
7 """
8
9 # APP
10 from apps.client import blueprint
11 from apps import db, login_manager
12 from apps.authentication.models import User, PhysicalInfo, Subscription, CoachingReview
13     , WorkoutReview, Review, Workout, WorkoutType, Session
14 from apps.client.forms import UpdateForm, AddReviewForm, ChangePasswordForm
15 from apps.config import Config
16 from apps.authentication.util import verify_pass, hash_pass
17
18 # FLASK
19 from flask import render_template, redirect, request, url_for, flash
20 from flask_login import login_required
21 from jinja2 import TemplateNotFound
22
23 from flask_login import (
24     current_user
25 )
26
27 # UTILS
28 import uuid as uuid
29 import os
30 from werkzeug.utils import secure_filename
31 from datetime import date, datetime
32
33 from apps.client.util import *
34 from apps.coach.util import get_program
35
36 @blueprint.route('/index')
37 @login_required
38 def index():
39     nextSessions = get_next_session(current_user.id)
40     return render_template('client/index.html', segment='index', nextSessions=
41         nextSessions)
42
43 # Create Profile Page
44 @blueprint.route('/profile', methods=['GET', 'POST'])
45 @login_required
46 def profile():
47     update_form = UpdateForm(request.form)
48     # Get the last physical information of the user
49     physicalInfo = get_physical_infos(current_user.id)
50
51     # Get the reviews posted by user
52     reviews = get_reviews(current_user.id)
53
54     # Get each workout type and how many client have made in 2 separate array to use them
55     # in js
56     wrktTypeList = get_workout_type_count(current_user.id)[0]
57
58     wrktTypeCount = get_workout_type_count(current_user.id)[1] if get_workout_type_count(
59         current_user.id)[1] != [] else 0
60
61     # Get the number of workout per month during this year
62     nbWorkoutPerMonth = get_workout_count_per_month(current_user.id)
63
64     # Get the average of heart rate during this week
65     avgHeartRate = get_average_heart_rate_last_week(current_user.id)
66
67     # Get the average of calories burned this week
68     avgCalories = get_average_calories_last_week(current_user.id)
69     # Get the total time training this week
70     totalTime = get_time_working_out_last_week(current_user.id)
71
72     # 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=
187                            client, target=target)
188
189 @blueprint.route('/checkout/', methods=['GET'])
190 @login_required
191 def checkout():
192     checkoutId = request.args.get('checkoutId')
193     #Get the check up values
194     checkUp = get_check_up(checkoutId)
195
196     return render_template('client/checkout.html', segment='checkout', checkUp=checkUp)
197
198 # Create Workouts Page
199 @blueprint.route('/workouts')
200 @login_required
201 def workouts():
202     # Get all data from workout
203     workouts = get_workouts(current_user.id)
204     return render_template('client/workouts.html', segment='workouts', workouts=workouts)
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,
27           USER.name, USER.surname
28     FROM SESSION
29     JOIN WORKOUT_TYPE ON SESSION.workout_type = WORKOUT_TYPE.id
30     JOIN USER ON USER.id = SESSION.coach_id
31     WHERE SESSION.client_id = 1 AND SESSION.start_time > curdate()
32     ORDER BY SESSION.start_time
33
34     Parameter(s) :
35     NAME      |  TYPE  |  DESC
36     userId    |  INT   |  The id of the user
37
38     Return :
39     | ARRAY[] | Array with all the next sessions planed for a user

```

```

39     """
40     sessions = db.session.query(Session.start_time, Session.end_time, Session.duration,
41                               WorkoutType.title, WorkoutType.logo, User.name, User.surname).join(WorkoutType,
42                               Session.workout_type==WorkoutType.id).join(User, Session.coach_id==User.id).filter(
43                               Session.client_id==userId).filter(Session.start_time>datetime.now()).order_by(
44                               Session.start_time)
45
46     return sessions
47
48 def get_review_author(clientId):
49     """
50     Get the client name and surname who add the review with his id
51
52     Parameter(s):
53     NAME      | TYPE  | DESC
54     clientId | INT   | the id of the client who post the review.
55
56     Return :
57     | USER() | User name and surname
58     """
59     result = db.session.query(User.name, User.surname).filter(User.id==clientId).first()
60
61     return result
62
63 def get_review_details(reviewId, reviewType):
64     """
65     Get the details of a review depends on his type
66
67     Parameter(s) :
68     NAME      | TYPE  | DESC
69     reviewId  | INT   | The id of the review
70     reviewType| STRING | The type of the review ("WORKOUT" or "COACHING")
71
72     Return :
73     | ARRAY[REVIEW(), ARRAY[]] | Array with the review details and an array that
74     contains the target details of the review
75     """
76     #Queries to get all Review from user
77     #coachingReviewQuery = db.session.query(CoachingReview.id, CoachingReview.
78     satisfaction.label("Field1"), CoachingReview.support.label("Field2"),
79     CoachingReview.disponibility.label("Field3"), CoachingReview.advice.label("Field4")
80     , CoachingReview.target_id.label("target_id"))
81     #workoutReviewQuery = db.session.query(WorkoutReview.id, WorkoutReview.difficulty,
82     WorkoutReview.feel, WorkoutReview.fatigue, WorkoutReview.energy, WorkoutReview.
83     target_id)
84
85     #UNION with the 2 queries
86     #reviewsUnion = union(coachingReviewQuery,workoutReviewQuery).alias()
87
88     # Query to get field from Review table and from the review union made before
89     #reviewsQuery = db.session.query(Review.id, Review.comment, Review.date, Review.type,
90     reviewsUnion.c.Field1, reviewsUnion.c.Field2, reviewsUnion.c.Field3, reviewsUnion.
91     c.Field4, Review.id_client, reviewsUnion.c.target_id).select_from(reviewsUnion).
92     join(Review, Review.id==reviewsUnion.c.COACHING_REVIEW_id).filter(Review.id_client==
93     id).order_by(Review.date.desc())
94
95     target = {}
96     if reviewType == "WORKOUT":
97         review = db.session.query(Review.id, Review.comment, Review.date, Review.type,
98         WorkoutReview.difficulty.label("Field1"), WorkoutReview.feel.label("Field2"),
99         WorkoutReview.fatigue.label("Field3"), WorkoutReview.energy.label("Field4"),
100         Review.id_client, WorkoutReview.target_id).join(Review, Review.id==
101         WorkoutReview.id).filter(Review.id==reviewId).first()
102
103         targetQuery = db.session.query(WorkoutType.title, Workout.date, Workout.
104         duration).join(WorkoutType, WorkoutType.id==Workout.workout_type).filter(Workout.id
105         ==review.target_id).first()
106
107         target['Type'] = targetQuery.title
108         target['Date'] = targetQuery.date
109         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

```

```

152     Parameter(s):
153     /
154
155     Return :
156     | ARRAY[STRING] | Array with the workout review fields name
157     """
158     fields = db.session.query(WorkoutReview).statement.columns.keys()
159
160     return fields
161
162 def get_coaching_review_field():
163     """
164     Get the fields name for a coaching review (the fields names are set in the database
165     )
166
167     Parameter(s):
168     /
169
170     Return :
171     | ARRAY[STRING] | Array with the coaching review fields name
172     """
173     fields = db.session.query(CoachingReview).statement.columns.keys()
174     return fields
175
176 def is_workout_reviewed(workoutId):
177     """
178     Check if a workout is already reviewed
179
180     Parameter(s):
181     NAME      | TYPE  | DESC
182     workoutId | INT   | The id of the workout
183
184     Return :
185     | BOOLEAN | True if the workout is already reviewed, else False
186     """
187     result = db.session.query(WorkoutReview.target_id).filter(WorkoutReview.target_id==
188     workoutId).first() is not None
189     return result
190
191 def get_user(userId):
192     """
193     Get user object with id
194
195     Parameter(s):
196     NAME      | TYPE  | DESC
197     userId    | INT   | The id of the user
198
199     Return :
200     | USER() | the user
201     """
202     user = User.query.filter_by(id=userId).first()
203     return user
204
205 def get_physical_infos(userId):
206     """
207     Get the last added physical infos for a user
208
209     Parameter(s):
210     NAME      | TYPE  | DESC
211     userId    | INT   | The id of the user
212
213     Return :
214     | PHYSICALINFO() | the last updated physical infos of the user
215     """
216     infos = PhysicalInfo.query.filter_by(user_id=userId).order_by(PhysicalInfo.date.
217     desc()).first()
218     return infos
219
220 def get_subscription_end_date(userId):

```



```

221     """
222     Get the last subscription purchased end date
223
224     Parameter(s):
225     NAME      | TYPE  | DESC
226     userId    | INT   | The id of the user
227
228     Request in SQL :
229     SELECT 'SUBSCRIPTION'.DURATION AS 'SUBSCRIPTION_DURATION', 'PURCHASE'.DATE AS '
230     PURCHASE_DATE'
231     FROM 'PURCHASE'
232     INNER JOIN 'SUBSCRIPTION' ON 'PURCHASE'.SUBSCRIPTION_ID = 'SUBSCRIPTION'.ID
233     WHERE 'PURCHASE'.CLIENT_ID = userId
234     ORDER BY 'PURCHASE'.DATE
235
236     Return :
237     | DATE | the end date of the last purchased subscription
238
239     """
240     subscription = db.session.query(Subscription.duration, Purchase.date).join(
241     Subscription, Purchase.subscription_id==Subscription.id).filter(Purchase.client_id
242     ==userId).order_by(Purchase.date.desc()).first()
243
244     endDate=''
245     if subscription != None:
246         #Add duration to the purchase date to get end date
247         endDate = (subscription[1] + relativedelta(months=subscription[0]))
248
249     return endDate
250
251 def get_workout_type_count(clientId):
252     """
253     Get the count of each type of workout made by user and split workout type count in
254     2 array (1 for the title and 1 for the count) to use them
255     in javascript with Chart.JS
256
257     Parameter(s) :
258     NAME      | TYPE  | DESC
259     clientId  | INT   | The id of the client
260
261     Return :
262     | ARRAY[ARRAY[STRING], ARRAY[INT]] | Array with 2 differents array inside. 1 for
263     all the titles and 1 for all the counts
264     """
265     workoutTypeCount = db.session.query(WorkoutType.title, func.count(Workout.
266     workout_type).label("count")).join(Workout, Workout.workout_type==WorkoutType.id).
267     filter(Workout.client_id==clientId).group_by(Workout.workout_type)
268
269     wrktTypeList = []
270     wrktTypeCount = []
271
272     for wtCount in workoutTypeCount:
273         wrktTypeList.append(' ' + wtCount.title + ' ')
274         wrktTypeCount.append(wtCount.count)
275
276     return [wrktTypeList, wrktTypeCount]
277
278 def get_workout_count_per_month(clientId):
279     """
280     Get the number of workout made each month during this year
281
282     Parameter(s) :
283     NAME      | TYPE  | DESC
284     clientId  | INT   | The id of the client
285
286     Return :
287     | ARRAY[INT] | Array with 12 int values represent the 12 months of a year. (Each
288     values is the number of workout made for the month)
289     """

```

```

285     count = db.session.query(func.month(Workout.date).label("month"), func.count(
286         Workout.id).label("count")).filter(Workout.client_id==clientId).filter(func.year(
287         date.today())==func.year(Workout.date)).group_by(func.month(Workout.date))
288
289     # Create a year array with 12 values (each value represent a month)
290     nbWorkoutPerMonth = [0,0,0,0,0,0,0,0,0,0,0,0]
291     for i in range(12):
292         for wtCountbyMonth in count:
293             if i == wtCountbyMonth.month-1: # If month got result from query set the
294                 value else keep 0
295                 nbWorkoutPerMonth[i] = wtCountbyMonth.count
296                 break
297
298     return nbWorkoutPerMonth
299
300 def get_actual_coach_id(clientId):
301     """
302     Get the actual coach id for a client
303
304     Parameter(s) :
305         NAME      |  TYPE  |  DESC
306         clientId  |  INT   |  The id of the client
307
308     Return :
309         | INT | The coach id
310     """
311     coachId = db.session.query(CoachedBy.coach_id).filter(CoachedBy.client_id==clientId)
312     .order_by(CoachedBy.end_date.desc()).first()
313
314     return coachId
315
316 def get_average_calories_last_week(clientId):
317     """
318     Get the average of calories burned this week
319
320     SQL :
321     SELECT AVG(calories) FROM WORKOUT WHERE WORKOUT.client_id = clientId AND WEEK(
322     WORKOUT.date) = WEEK(CURDATE()) - 1;
323
324     Parameter(s) :
325         NAME      |  TYPE  |  DESC
326         clientId  |  INT   |  The id of the client
327
328     Return :
329         | DECIMAL | the average of calories burned rounded to 1 decimal
330     """
331     calories = db.session.query(func.avg(Workout.calories).label("avg")).filter(Workout
332     .client_id==clientId).filter(func.week(Workout.date)>=func.week(date.today()) -1).
333     first()
334
335     return round(calories.avg,1) if calories.avg != None else 0.0
336
337 def get_average_heart_rate_last_week(clientId):
338     """
339     Get the average heart rate recorded this week
340
341     SQL :
342     SELECT AVG(heart_rate_avg) FROM WORKOUT WHERE WORKOUT.client_id = clientId AND
343     WEEK(WORKOUT.date) = WEEK(CURDATE()) - 1;
344
345     Parameter(s) :
346         NAME      |  TYPE  |  DESC
347         clientId  |  INT   |  The id of the client
348
349     Return :
350         | DECIMAL | the average of heart rate recorded rounded to 1 decimal
351     """
352     heart_rate = db.session.query(func.avg(Workout.heart_rate_avg).label("avg")).filter
353     (Workout.client_id==clientId).filter(func.week(Workout.date)>=func.week(date.today
354     ()) -1).first()

```

```

347     return round(heart_rate.avg,1) if heart_rate.avg != None else 0.0
348
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.
363     extract("minute",Workout.duration)*60+ func.extract("second",Workout.duration)).
364     label("time")).filter(Workout.client_id==clientId).filter(func.week(Workout.date)>
365     func.week(date.today())-1)
366     result = 0.0
367
368     for workout in subquery:
369         result += workout.time
370
371     return round(result /60,1)
372
373 def get_weight_update(clientId):
374     """
375     Get the average weight each month of a client during the whole year (months without
376     values will show 0)
377
378     Parameter(s) :
379         NAME      |  TYPE  |  DESC
380         clientId  |  INT   |  The id of the client
381
382     Return :
383     | ARRAY[INT] | Array with 12 int values represent the 12 months of a year. (Each
384     values is the average of all the weight recorded for the month in kg)
385     """
386
387     weights = db.session.query(func.month(PhysicalInfo.date).label("month"), func.avg(
388     PhysicalInfo.weight).label("avg")).filter(PhysicalInfo.user_id==clientId).filter(
389     func.year(date.today())==func.year(PhysicalInfo.date)).group_by(func.month(
390     PhysicalInfo.date))
391
392     # Create a year array with 12 values (each value represent a month)
393     weightPerMonth = [0,0,0,0,0,0,0,0,0,0,0,0]
394     for i in range(12):
395         for weight in weights:
396             if i == weight.month-1: # If month got result from query set the value else
397                 keep 0
398                 weightPerMonth[i] = float(round(weight.avg,1))
399                 break
400
401     return weightPerMonth
402
403 def get_all_check_up(clientId):
404     """
405     Get all the check ups made for a client
406
407     Parameter(s) :
408         NAME      |  TYPE  |  DESC
409         clientId  |  INT   |  The id of the client
410
411     Return :
412     | ARRAY(QUERY()) | Array contains query object with field selected as properties (
413     query object as PhysicalInfo)
414     """
415
416     checkUp = db.session.query(PhysicalInfo.id, PhysicalInfo.date, PhysicalInfo.weight)
417     .filter(PhysicalInfo.user_id==clientId).order_by(PhysicalInfo.date.desc())
418     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     def show_menu(self):
46         while self.running:
47             print("\nChoose an option:\n" +
48                 "-----\n" +
49                 "1) Get user information\n" +
50                 "2) Check available data\n" +
51                 "3) Revoke access token\n" +
52                 "4) Exit\n" +
53                 "-----")
54             self.get_menu_choice()
55
56     def get_menu_choice(self):
57

```

```

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     ],
69                                                         access_token=self.config["
70     access_token"])
71     pretty_print_json(user_info)
72
73     def check_available_data(self):
74         available_data = self.accesslink.pull_notifications.list()
75
76         if not available_data:
77             print("No new data available.")
78             return False
79
80         print("Available data:")
81         pretty_print_json(available_data)
82
83         for item in available_data["available-user-data"]:
84             if item["data-type"] == "EXERCISE":
85                 self.get_exercises()
86             elif item["data-type"] == "ACTIVITY_SUMMARY":
87                 self.get_daily_activity()
88             elif item["data-type"] == "PHYSICAL_INFORMATION":
89                 self.get_physical_info()
90
91     def revoke_access_token(self):
92         self.accesslink.users.delete(user_id=self.config["user_id"],
93                                     access_token=self.config["access_token"])
94
95         del self.config["access_token"]
96         del self.config["user_id"]
97         save_config(self.config, CONFIG_FILENAME)
98
99         print("Access token was successfully revoked.")
100
101         self.exit()
102
103     def exit(self):
104         self.running = False
105
106     def get_exercises(self, userId):
107         # Query to insert
108         insertWorkoutQuery = "INSERT INTO WORKOUT (workout_type, client_id, date,
109         duration, heart_rate_max, heart_rate_avg, calories, distance) VALUES (%s, %s, %s, %
110         s, %s, %s, %s, %s)"
111
112         #Init values
113         values = ()
114
115         workout_type = ''
116         client_id = userId
117         date = ''
118         duration = ''
119         heart_rate_max = ''
120         heart_rate_avg = ''
121         calories = ''
122         distance = 0.0
123
124         transaction = self.accesslink.training_data.create_transaction(user_id=self.
125         config["user_id"],
126                                                         access_token=
127         self.config["access_token"])
128         if not transaction:
129             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')
140             .%fZ')
141
142             #Try different format (depends on duration exemple : PT1H30M)
143             if 'H' in exercise_summary["duration"]:
144                 duration = datetime.strptime(exercise_summary["duration"], 'PT%HH%MM')
145             elif 'M' in exercise_summary["duration"]:
146                 duration = datetime.strptime(exercise_summary["duration"], 'PT%MM%S.%fS')
147             elif 'S' in exercise_summary["duration"]:
148                 duration = datetime.strptime(exercise_summary["duration"], 'PT%S.%fS')
149
150             heart_rate_max = exercise_summary["heart-rate"]["maximum"]
151             heart_rate_avg = exercise_summary["heart-rate"]["average"]
152             calories = exercise_summary["calories"]
153
154             if 'distance' in exercise_summary:
155                 distance = exercise_summary['distance']
156
157             values = (workout_type, client_id, date, duration, heart_rate_max,
158                     heart_rate_avg, calories, distance)
159
160             print(values)
161             cursor.execute(insertWorkoutQuery, values)
162             db.commit()
163             print(cursor.rowcount, "was inserted.")
164             transaction.commit()
165
166         def get_daily_activity(self):
167             transaction = self.accesslink.daily_activity.create_transaction(user_id=self.
168                                     config["user_id"],
169                                     access_token=
170                                     self.config["access_token"])
171             if not transaction:
172                 print("No new daily activity available.")
173                 return
174
175             resource_urls = transaction.list_activities()["activity-log"]
176
177             for url in resource_urls:
178                 activity_summary = transaction.get_activity_summary(url)
179
180                 print("Activity summary:")
181                 pretty_print_json(activity_summary)
182
183                 transaction.commit()
184
185         def get_physical_info(self):
186             transaction = self.accesslink.physical_info.create_transaction(user_id=self.
187                                     config["user_id"],
188                                     access_token=
189                                     self.config["access_token"])
190             if not transaction:
191                 print("No new physical information available.")
192                 return

```

```

188     resource_urls = transaction.list_physical_infos()["physical-informations"]
189
190     for url in resource_urls:
191         physical_info = transaction.get_physical_info(url)
192
193         print("Physical info:")
194         pretty_print_json(physical_info)
195
196     transaction.commit()

```

```

1  #!/usr/bin/env python
2  # encoding: utf-8
3  """
4  Author   :      Thomas Fujise
5  Date    :      02.06.2022
6  File    :      config.py
7  Version :      1.0.0
8  Brief   :      Config constants
9  """
10
11 #DATABASE
12 DB_HOST = "localhost"
13 DB_USER = "fitjourney_dba"
14 DB_PASSWORD = "Super2012$"
15 DB = "fitjourney"
16
17 #WORKOUT TYPE
18 WORKOUT_TYPE = {
19     "STRENGTH_TRAINING": 1,
20     "CYCLING" : 2,
21     "RUNNING" : 3
22 }

```

```

1  #!/usr/bin/env python
2
3  """
4  Author   :      Thomas Fujise
5  Date    :      02.06.2022
6  File    :      db.py
7  Version :      1.0.0
8  Brief   :      Database connector
9  """
10
11 import config
12 import mysql.connector
13
14 db = mysql.connector.connect(
15     host=config.DB_HOST,
16     user=config.DB_USER,
17     password=config.DB_PASSWORD,
18     database=config.DB
19 )
20
21 cursor = db.cursor()
22
23 def getUserIdByCard(cardId):
24     sql = "SELECT id FROM USER WHERE card_id = %s"
25     card = (cardId,)
26
27     cursor.execute(sql, card)
28
29     result = cursor.fetchone()
30
31     return result

```

```

1  """
2  Author   :      Thomas Fujise
3  Date    :      02.06.2022
4  File    :      main.py
5  Version :      1.0.0
6  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

```

```

7 """
8 from smartcard.CardRequest import CardRequest
9 from smartcard.Exceptions import CardRequestTimeoutException
10 from smartcard.CardType import AnyCardType
11 from smartcard import util
12 from accesslink_polar import PolarAccessLink
13 import time
14
15 from db import cursor, getUserIdByCard
16
17 if __name__ == '__main__':
18
19     #Init Polar class
20     polar = PolarAccessLink()
21
22     #Query to get the card id from user
23     selectQuery = "SELECT name, surname, card_id FROM USER"
24
25
26     cursor.execute(selectQuery)
27
28     myresult = cursor.fetchall()
29     # respond to the insertion of any type of smart card
30     card_type = AnyCardType()
31
32     # create the request. Wait for up to x seconds for a card to be attached
33     request = CardRequest(timeout=None, cardType=card_type)
34
35     entry = False
36     while True:
37         time.sleep(0.1)
38         # listen for the card
39         service = None
40         try:
41             service = request.waitforcard()
42         except CardRequestTimeoutException:
43             print("ERROR: No card detected")
44             exit(-1)
45
46         # when a card is attached, open a connection
47         conn = service.connection
48         conn.connect()
49
50         # get and print the ATR and UID of the card
51         get_uid = util.toBytes("FF CA 00 00 00")
52
53         data, sw1, sw2 = conn.transmit(get_uid)
54
55         print(data)
56         card_scanned = ' '.join(str(e) for e in data)
57         print(card_scanned)
58         card_found = False
59
60         for client in myresult:
61
62             if client[2] == card_scanned:
63                 card_found = True
64                 if entry:
65                     print("-----GOODBYE " + client[0] + " " + client
66 [1]+ "-----")
67                     if polar.check_available_data():
68                         polar.get_exercises(getUserIdByCard(client[2]))[0]
69                         entry = False
70                     else:
71                         print("-----WELCOME " + client[0] + " " + client
72 [1]+ "-----")
73                         entry = True
74
75         if not card_found:
76             print("-----UNRECOGNIZED CARD-----")

```



```

77         time.sleep(2)

1  <!--
2  =====
3  Author      :      Thomas Fujise
4  Date        :      11.04.2022
5  File        :      base.html
6  Version     :      1.0.0
7  Brief       :      HTML base for each page of the application
8  =====
9  -->
10 <!DOCTYPE html>
11 <html lang="en">
12 <head>
13     <meta charset="utf-8" />
14     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"
15     >
16     <title>
17         FitJourney - {% block title %} {% endblock %}
18     </title>
19     {% include "includes/css.html" %}
20     <!-- Specific CSS HERE -->
21     {% block stylesheets %}{% endblock stylesheets %}
22 </head>
23 <body class="">
24     {% include "includes/nav.html" %}
25     <div class="main-content" id="panel">
26         <!-- Header -->
27         <div class="header bg-gradient-purple py-7 py-lg-8">
28             <div class="container">
29                 <div class="header-body text-center mb-7">
30                     <div class="row justify-content-center">
31                         <div class="col-lg-5 col-md-6">
32                             <h1 class="text-white">
33                                 <a target="_blank" class="text-white"
34                                 href="">FitJourney</a>
35                             </h1>
36                             <p class="text-lead text-light">
37                                 The application to manage your coaching
38                             </p>
39                         </div>
40                     </div>
41                 </div>
42             </div>
43             {% block content %}{% endblock content %}
44         </div>
45         {% include "includes/scripts.html" %}
46         <!-- Specific JS HERE -->
47         {% block javascripts %}{% endblock javascripts %}
48     </body>
49 </html>

1  <!-- Fonts -->
2  <link href="https://fonts.googleapis.com/css?family=Poppins:200,300,400,600,700,800"
3  rel="stylesheet">
4
5  <!-- Icons -->
6  <link href="/static/assets/vendor/nucleo/css/nucleo.css" rel="stylesheet">
7
8  <!-- Argon CSS -->
9  <link type="text/css" href="/static/assets/css/argon.min.css" rel="stylesheet">
10 <link rel="stylesheet" href="/static/assets/css/argon.css?v=1.2.0" type="text/css">

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

```

```

8     </div>
9     <div class="navbar-inner">
10         <!-- Collapse -->
11         <div class="collapse navbar-collapse" id="sidenav-collapse-main">
12             <!-- Nav items -->
13             <ul class="navbar-nav">
14                 {% if not current_user.is_authenticated %}
15                 <li class="nav-item">
16                     <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
17                     = "{{ url_for('authentication_blueprint.login') }}">
18                         <i class="ni ni-single-02 text-yellow"></i>
19                         <span class="nav-link-text">Login</span>
20                     </a>
21                 </li>
22                 <li class="nav-item">
23                     <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
24                     = "{{ url_for('authentication_blueprint.register') }}">
25                         <i class="ni ni-single-02 text-yellow"></i>
26                         <span class="nav-link-text">Register</span>
27                     </a>
28                 </li>
29                 {% elif current_user.is_authenticated and current_user.role == 1 %}
30                 <li class="nav-item">
31                     <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
32                     = "{{ url_for('client_blueprint.profile') }}">
33                         <i class="ni ni-single-02 text-yellow"></i>
34                         <span class="nav-link-text">Profile</span>
35                     </a>
36                 </li>
37                 <li class="nav-item">
38                     <a class="nav-link {% if 'index' in segment %} active {% endif %}" href="
39                     /">
40                         <i class="ni ni-calendar-grid-58 text-orange"></i>
41                         <span class="nav-link-text">Next Sessions</span>
42                     </a>
43                 </li>
44                 <li class="nav-item">
45                     <a class="nav-link {% if 'workouts' in segment %} active {% endif %}"
46                     href="{{ url_for('client_blueprint.workouts') }}">
47                         <i class="ni ni-planet text-orange"></i>
48                         <span class="nav-link-text">Workouts</span>
49                     </a>
50                 </li>
51                 {% elif current_user.is_authenticated and current_user.role == 2 %}
52                 <li class="nav-item">
53                     <a class="nav-link {% if 'dashboard' in segment %} active {% endif %}"
54                     href="{{ url_for('coach_blueprint.dashboard') }}">
55                         <i class="ni ni-tv-2 text-primary"></i>
56                         <span class="nav-link-text">Dashboard</span>
57                     </a>
58                 </li>
59                 <li class="nav-item">
60                     <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
61                     = "{{ url_for('client_blueprint.profile') }}">
62                         <i class="ni ni-single-02 text-yellow"></i>
63                         <span class="nav-link-text">Profile</span>
64                     </a>
65                 </li>
66                 <li class="nav-item">
67                     <a class="nav-link {% if 'calendar' in segment %} active {% endif %}"
68                     href="{{ url_for('coach_blueprint.calendar') }}">
69                         <i class="ni ni-calendar-grid-58 text-orange"></i>
70                         <span class="nav-link-text">Calendar</span>
71                     </a>
72                 </li>
73             {% endif %}
74         </ul>
75         <!-- Divider -->
76         <hr class="my-3">
77         {% if current_user.is_authenticated %}
78         <!-- Heading -->
79         <ul class="navbar-nav">

```

```

72         <li class="nav-item">
73             <a class="nav-link" href="{ url_for('authentication_blueprint.logout') }}"
74             >
75                 <i class="ni ni-user-run text-red"></i><i class="fa fa-sign-out" aria-
76                 hidden="true"></i>
77                 <span class="nav-link-text">Logout</span>
78             </a>
79         </li>
80     </ul>
81     {% endif %}
82 </div>
83 </div>
84 </nav>

```

```

1 <!-- Core -->
2 <script src="https://code.jquery.com/jquery-3.6.0.slim.js" integrity="sha256-
3 HwWONEZrpuoh951cQD1ov2HUK5zA5DwJ1DNUXaM6FsY=" crossorigin="anonymous"></script>
4 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
5 integrity="sha384-IQsoLX15PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
6 crossorigin="anonymous"></script>
7 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
8 integrity="sha384-cVKIPhGWIC2A14u+LWgxfKTRICfu0JTxR+EQDz/bgl0Eyl14H0zUF0QKbrJ0EcQF"
9 crossorigin="anonymous"></script>
10 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.1/chart.min.js"
11 integrity="sha512-QSkVNOCYLjt73J4hbmVo0V6KVZuMluZlioC+
12 trLpewV8qMjsWqlIQvkn1KGX2StWvPMdWGBqim1xlC8krl1EKQ==" crossorigin="anonymous"
13 refererPolicy="no-referrer"></script>
14
15
16
17 <!-- Theme JS -->
18 <script src="/static/assets/js/argon.js?v=1.2.0"></script>

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Add check up {% endblock title %}
4
5 {% 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=
15         "Close">Close</button>
16     </div>
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">Add check up </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.check_up', clientId=infos.id) }}"
37                     method="POST">
38                         {{ 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             </div>
59         </div>
60         <hr class="my-4" />
61         <div class="row justify-content-center">
62             <div class="col-lg-3">
63                 <div class="form-group">
64                     <label class="form-control-label" for="weight">Weight (kg)</label
>
65                     {{form.weight(placeholder="Weight",class="form-control", oninput=
"this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\.\.*)"\/g, '$1');")}}
66                 </div>
67             </div>
68             <div class="col-lg-3">
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 class="col-lg-3">
75                 <div class="form-group">
76                     <label class="form-control-label" for="water">Water %</label>
77                     {{form.water(placeholder="Water %",class="form-control", oninput=
"this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\.\.*)"\/g, '$1');")}}
78                 </div>
79             </div>
80             <div class="col-lg-3">
81                 <div class="form-group">
82                     <label class="form-control-label" for="protein">Protein %</label>
83                     {{form.protein(placeholder="Protein %",class="form-control",
oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\.\.*)"\/g, '$1'
);")}}
84                 </div>
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">

```

```

96         <div class="form-group">
97             <label class="form-control-label" for="body_fat_percent">Body
fat %</label>
98             {{form.body_fat_percent(placeholder="Body fat %",class="form-
control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\..*)
\./g, '$1');")}}
99         </div>
100     </div>
101     <div class="col-lg-3">
102         <div class="form-group">
103             <label class="form-control-label" for="bone_mass_percent">
Bone mass %</label>
104             {{form.bone_mass_percent(placeholder="Bone mass %",class="
form-control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace
(/(\..*)\./g, '$1');")}}
105         </div>
106     </div>
107     <div class="col-lg-3">
108         <div class="form-group">
109             <label class="form-control-label" for="bmr">BMR</label>
110             {{form.bmr(placeholder="BMR",class="form-control", oninput="
this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\..*)\./g, '$1');")}}
111         </div>
112     </div>
113 </div>
114
115     <div class="row justify-content-center">
116         <div class="col-lg-3">
117             <div class="form-group">
118                 <label class="form-control-label" for="muscle_mass">Muscle
Mass (kg)</label>
119                 {{form.muscle_mass(placeholder="Muscle mass (kg)",class="form
-control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\..*)
\./g, '$1');")}}
120             </div>
121         </div>
122         <div class="col-lg-3">
123             <div class="form-group">
124                 <label class="form-control-label" for="body_fat_percent">Body
fat (kg)</label>
125                 {{form.body_fat(placeholder="Body fat (kg)",class="form-
control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\..*)
\./g, '$1');")}}
126             </div>
127         </div>
128         <div class="col-lg-3">
129             <div class="form-group">
130                 <label class="form-control-label" for="bone_mass_percent">
Visceral fat (kg)</label>
131                 {{form.fat_visceral(placeholder="Visceral fat (kg)",class="
form-control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace
(/(\..*)\./g, '$1');")}}
132             </div>
133         </div>
134         <div class="col-lg-3">
135             <div class="form-group">
136                 <label class="form-control-label" for="bmr">Bone mass (kg)</
label>
137                 {{form.bone_mass(placeholder="Bone mass (kg)",class="form-
control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\..*)
\./g, '$1');")}}
138             </div>
139         </div>
140     </div>
141     <div class="row justify-content-center">
142         <div class="col-lg-3">
143             <div class="form-group">
144                 <label class="form-control-label" for="lean_body_mass">Lean
body mass (kg)</label>
145                 {{form.lean_body_mass(placeholder="Lean body mass (kg)",class
="form-control", oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace
(/(\..*)\./g, '$1');")}}

```

```

146         </div>
147     </div>
148     <div class="col-lg-3">
149         <div class="form-group">
150             <label class="form-control-label" for="lean_body_mass">Body
age</label>
151             {{form.body_age(placeholder="Body age",class="form-control",
oninput="this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\.\.*)"
;"))}}
152         </div>
153     </div>
154     <div class="col-lg-3">
155         <div class="form-group">
156             <label class="form-control-label" for="bmi">BMI</label>
157             {{form.bmi(placeholder="BMI",class="form-control", oninput="
this.value = this.value.replace(/[~0-9.]/g, '').replace(/(\.\.*)"
;"))}}
158         </div>
159     </div>
160 </div>
161 <div class="row">
162     <div class="col-md-12">
163         <div class="form-group">
164             {{form.add(class="form-control btn btn-primary")}}
165         </div>
166     </div>
167 </div>
168 </div>
169 </form>
170 </div>
171 </div>
172 </div>
173
174
175
176 {% endblock content %}
177
178 <!-- Specific JS goes HERE -->
179 {% block javascripts %}
180
181 {% endblock javascripts %}

```

```

1  {% extends 'layout/base.html' %}
2
3  {% block title %} Add client {% endblock title %}
4
5  {% 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">Add new client </h3>
29                          </div>
30                          <div class="col-4 text-right">

```

```

31         </div>
32     </div>
33 </div>
34 <div class="card-body">
35     <form action="/add_client" method="POST">
36         {{ form.hidden_tag() }}
37         <div class="p1-lg-4">
38             <div class="row justify-content-center">
39                 <div class="col-lg-6">
40                     <div class="form-group">
41                         <label class="form-control-label" for="name">Name</label>
42                         {{form.name(placeholder="Name", class="form-control", type="
43 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</
49 label>
50                         {{form.surname(placeholder="Surname", class="form-control",
51 type="text")}}
52                     </div>
53                 </div>
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="
60 email")}}
61                     </div>
62                 </div>
63                 <div class="col-lg-4">
64                     <div class="form-group">
65                         <label class="form-control-label" for="birthdate">Date of
66 birth</label>
67                         {{form.birthdate(placeholder="Date of birth", class="form-
68 control")}}
69                     </div>
70                 </div>
71                 <div class="col-lg-4">
72                     <div class="form-group">
73                         <label class="form-control-label" for="height">Height (cm)</
74 label>
75                         {{form.height(placeholder="Height", class="form-control",
76 oninput="this.value = this.value.replace(/[^0-9.]/g, '').replace(/(\.\.)*\./g, '$1'
77 );")}}
78                     </div>
79                 </div>
80             </div>
81             <div class="row justify-content-center">
82                 <div class="col-lg-4">
83                     <div class="form-group">
84                         <label class="form-control-label" for="start_date">
85 Starting date</label>
86                         {{form.start_date(class="form-control", onchange="
87 updateEndTime()")}}
88                     </div>
89                 </div>
90                 <div class="col-lg-4">
91                     <div class="form-group">
92                         <label class="form-control-label" for="subscription">
93 Subscription type</label>
94                         {{form.subscription(class="form-control", onchange="
95 updateEndTime()")}}
96                     </div>
97                 </div>
98             </div>
99         </div>
100     </form>
101 </div>

```

```

90         <div class="form-group">
91             <label class="form-control-label" for="end_date">
Subscription until</label>
92                 {{form.end_date(class="form-control", readonly=readonly)
}}
93             </div>
94         </div>
95     </div>
96
97     <div class="row">
98         <div class="col-md-12">
99             <div class="form-group">
100                 {{form.add(class="form-control btn btn-primary")}}
101             </div>
102         </div>
103     </div>
104 </div>
105 </form>
106 </div>
107 </div>
108 </div>
109
110
111
112 {% endblock content %}
113
114 <!-- Specific JS goes HERE -->
115 {% block javascripts %}
116 <script>
117     function updateEndTime(){
118         var subscription = document.getElementById("subscription").value;
119         var start_date = document.getElementById("start_date").value;
120
121         if(subscription != "" && start_date != ""){
122             //Check if the fields aren't empty, then set a end time depends on values
selected before
123             var dt = new Date(start_date);
124             dt.setMonth(dt.getMonth() + parseInt(subscription));
125             let [end_date] = dt.toISOString().split('T');
126             document.getElementById("end_date").value = end_date
127         }
128     }
129
130 </script>
131 {% endblock javascripts %}

```

```

1  {% extends 'layout/base.html' %}
2
3  {% block title %} Add new program {% endblock title %}
4
5  {% 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">Add new program </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.add_program')}}" method="POST"
37         enctype="multipart/form-data">
38             {{ form.csrf_token() }}
39             <div class="pl-lg-4">
40                 <div class="row justify-content-center">
41                     <div class="col-lg-6">
42                         <div class="form-group">
43                             <label class="form-control-label" for="program_type">Type</
44                             label>
45                             {{form.type(placeholder="Type",class="form-control")}}
46                         </div>
47                         <div class="col-lg-6">
48                             <div class="form-group">
49                                 <label class="form-control-label" for="program_file">File</
50                                 label>
51                                 {{form.file(placeholder="Select your file",class="form-
52                                 control", accept="application/pdf")}}
53                             </div>
54                         </div>
55                     <div class="row">
56                         <div class="col-md-12">
57                             <div class="form-group">
58                                 {{form.client(class="form-control", value=clientId)}}
59                                 {{form.add(class="form-control btn btn-primary")}}
60                             </div>
61                         </div>
62                     </div>
63                 </div>
64             </form>
65         </div>
66     </div>
67
68
69
70 {% endblock content %}
71
72 <!-- Specific JS goes HERE -->
73 {% block javascripts %}
74 {% endblock javascripts %}

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Add review {% endblock title %}
4
5 {% block stylesheets %}{% endblock stylesheets %}
6
7 {% block content %}
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=
15         "Close">Close</button>
16     </div>
17
18     {% 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>

```

```

80     </div>
81   </form>
82 </div>
83 </div>
84
85 {% endblock content %}
86 {% block javascripts %}
87 <script>
88   // Update output value for each range input
89   function outputUpdate(id, val){
90     document.getElementById(id.concat("-value")).value = val;
91   }
92 </script>
93 {% endblock javascripts %}

1  div{% extends 'layout/base.html' %}
2
3  {% block title %} Calendar {% endblock title %}
4
5  {% block stylesheets %}
6  <link type="text/css" rel="stylesheet" href="{ url_for('static', filename='assets/css
   /calendar.css') }}" />
7
8  <style>
9
10   body {
11     font-family: Arial, Helvetica Neue, Helvetica, sans-serif;
12     font-size: 14px;
13   }
14
15   #calendar {
16     max-width: 1100px;
17     margin: 0 ;
18   }
19
20 </style>
21 {% endblock stylesheets %}
22
23 {% block content %}
24
25
26
27 <div class="container-fluid mt--8" style="margin: auto;">
28   {% for category, message in get_flashed_messages(with_categories=true) %}
29
30   <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
31     {{ message }}
32     <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
       "Close">Close</button>
33   </div>
34
35   {% endfor %}
36 <div id="sessions" data-sessions="{{sessions}}"></div>
37 <div class="row">
38   <div class="card card-body h-auto" style="width: 18rem; float: right;">
39     <h3 class="card-header">Calendar :</h3>
40
41     <div class="row">
42
43       <div class="col-lg-6">
44         <div class="card border-light mb-3" style="max-width: 50rem;">
45           <div class="card-header">Sessions</div>
46           <div class="card-body">
47             <div id='calendar'></div>
48           </div>
49         </div>
50       </div>
51       <div class="col-lg-6">
52         <div class="card border-light mb-3" style="max-width: 50rem;">
53           <div class="card-header">Add a session</div>
54           <div class="card-body">
55             <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 :
61
62                     {{form.client(class="form-control")}}
63                 </div>
64             </div>
65             <div class="col-lg-4">
66                 <div class="form-group">
67                     <label class="form-control-label" for="datepicker">Date :</
68                     label>
69
70                     {{ form.date(placeholder="Date", class="form-control",
71                     type="date")}}
72                 </div>
73             </div>
74
75             <div class="col-lg-4">
76                 <div class="form-group">
77                     <label class="form-control-label" for="start_time">Time for
78                     the session</label>
79
80                     {{form.start_time(class="form-control" , type="time", min="
81                     07:00:00", max="20:00:00", onchange="updateEndTime()")}}
82                 </div>
83             </div>
84
85             <div class="row">
86                 <div class="col-lg-4">
87                     <div class="form-group">
88                         <label class="form-control-label" for="type_picker">Type </
89                         label>
90
91                         {{form.type(class="form-control")}}
92                     </div>
93                 </div>
94                 <div class="col-lg-4">
95                     <div class="form-group">
96                         <label class="form-control-label" for="duration">Duration (h)
97
98                         {{form.duration(class="form-control", onchange="updateEndTime
99                         ()")}}
100                     </div>
101                 </div>
102             </div>
103
104             <div class="col-lg-4">
105                 <div class="form-group">
106                     <label class="form-control-label" for="end_time">End time :</
107                     label>
108
109                     {{form.end_time(class="form-control" , type="time")}}
110                 </div>
111             </div>
112
113             <div class="row">
114                 <div class="col-lg-12">
115                     <div class="form-group">
116                         {{form.add(class="form-control btn btn-primary")}}
117                     </div>
118                 </div>
119             </div>
120         </div>
121     </div>
122 </div>
123 {% endblock content %}
124

```

```

119 <!-- Specific JS goes HERE -->
120 {% block javascripts %}
121 <script type="text/javascript" src="{% url_for('static', filename='assets/js/calendar.
    js') %}"></script>
122 <script>
123     function updateEndTime(){
124         var duration = document.getElementById("duration").value;
125         var start = document.getElementById("start_time").value;
126         var date = document.getElementById("datepicker").value;
127
128         if(duration != "" && start != "" && date != ""){
129             //Check if the fields aren't empty, then set a end time depends on values
            selected before
130             var dt = new Date(date+"T"+start);
131             dt.setHours(dt.getHours() + parseInt(duration));
132             document.getElementById("end_time").value = dt.toTimeString().slice(0, 8);
133         }
134     }
135 }
136
137 document.addEventListener('DOMContentLoaded', function() {
138     var calendarEl = document.getElementById('calendar');
139
140     var calendar = new FullCalendar.Calendar(calendarEl, {
141         initialDate: new Date(Date.now()),
142         initialView: 'timeGridWeek',
143         headerToolbar: {
144             left: 'prev,next today',
145             center: 'title',
146             right: 'dayGridMonth,timeGridWeek,timeGridDay,listWeek'
147         },
148         height: "600px",
149         navLinks: true, // can click day/week names to navigate views
150         editable: false,
151         selectable: true,
152         selectMirror: true,
153         nowIndicator: true,
154         events: [{ sessions | safe }]
155     });
156
157     calendar.render();
158 });
159
160 </script>
161 {% endblock javascripts %}

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Profile {% endblock title %}
4
5 {% 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">Change password </h3>
29         </div>
30         <div class="col-4 text-right">
31             <a href="#" class="btn btn-sm btn-warning">Password</a>
32         </div>
33     </div>
34 </div>
35 <div class="card-body">
36     <form action="/change_password" method="POST">
37         {{ form.hidden_tag() }}
38         <div class="pl-lg-4">
39             <div class="row justify-content-center">
40
41                 <div class="col-lg-4">
42                     <div class="form-group">
43                         <label class="form-control-label" for="oldPassword">Old
password</label>
44                         {{ form.oldPassword(placeholder="Old password", class="form-
control") }}
45                     </div>
46                 </div>
47             </div>
48             <div class="row justify-content-center">
49                 <div class="col-lg-4">
50                     <div class="form-group">
51                         <label class="form-control-label" for="newPassowrd">New
password</label>
52                         {{ form.newPassword(placeholder="New password", class="form-
control") }}
53                     </div>
54                 </div>
55             </div>
56             <div class="row justify-content-center">
57                 <div class="col-lg-4">
58                     <div class="form-group">
59                         <label class="form-control-label" for="confirmPassword">
Confirm new password</label>
60                         {{ form.confirmPassword(placeholder="Confirm new password
", class="form-control") }}
61                     </div>
62                 </div>
63             </div>
64
65             <div class="row">
66                 <div class="col-md-12">
67                     <div class="form-group">
68                         {{ form.change(class="btn btn-primary btn-block") }}
69                     </div>
70                 </div>
71             </div>
72         </div>
73     </form>
74 </div>
75 </div>
76 </div>
77
78
79 {% endblock content %}
80
81 <!-- Specific JS goes HERE -->
82 {% block javascripts %}
83
84
85 {% endblock javascripts %}

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Add check up {% endblock title %}
4
5 {% 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=
15             "Close">Close</button>
16     </div>
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")
29                             }}</h3>
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.
63                                         water_percentage}}</h5>
64                                     </div>
65                                 </div>
66                             </div>
67                             <div class="col-lg-3">
68                                 <div class="form-group">
69                                     <div class="card-header text-center">Protein (%)</div>
70                                     <div class="card-body">
71                                         <h5 class="card-title text-center"> {{checkUp.
72                                         protein_percentage}}</h5>
73                                     </div>
74                                 </div>
75                             </div>
76                         </div>
77                     </div>
78                 </div>
79             </div>
80         </div>
81     </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">

```



```

139         <h5 class="card-title text-center"> {{checkUp.bone_mass}}
140     </h5>
141     </div>
142 </div>
143 </div>
144 <div class="row justify-content-center">
145     <div class="col-lg-3">
146         <div class="form-group">
147             <div class="card-header text-center">Lean body mass (kg)</
148         div>
149             <div class="card-body">
150                 <h5 class="card-title text-center"> {{checkUp.
151                 leanbody_mass}}</h5>
152             </div>
153         </div>
154     </div>
155     <div class="col-lg-3">
156         <div class="form-group">
157             <div class="card-header text-center">Body age</div>
158             <div class="card-body">
159                 <h5 class="card-title text-center"> {{checkUp.body_age}}<
160             /h5>
161         </div>
162     </div>
163     <div class="col-lg-3">
164         <div class="form-group">
165             <div class="card-header text-center">BMI </div>
166             <div class="card-body">
167                 <h5 class="card-title text-center"> {{checkUp.bmi}}</h5>
168             </div>
169         </div>
170     </div>
171 </div>
172 </div>
173 </div>
174 </div>
175 </div>
176
177
178
179 {% endblock content %}
180
181 <!-- Specific JS goes HERE -->
182 {% block javascripts %}
183
184 {% endblock javascripts %}

```

```

1  {% extends 'layout/base.html' %}
2
3  {% block title %} Client {% endblock title %}
4
5  {% 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="
15              "Close">Close</button>
16          </div>
17      {% endfor %}
18
19
20

```

```

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                         </div>
54                         <div class="row">
55                             <div class="col-lg-4">
56                                 <div class="form-group">
57                                     <label class="form-control-label" for="birthdate"> Birthdate</
label>
58                                     {{ form.birthdate(placeholder="Birthdate", class="form-control",
type="date", value=client.birthdate)}}
59                                 </div>
60                             </div>
61                             <div class="col-lg-4">
62                                 <div class="form-group">
63                                     <label class="form-control-label" for="email">Email address</
label>
64                                     {{ form.email(placeholder="Email", class="form-control", type="
email", value=client.email)}}
65                                 </div>
66                             </div>
67                             <div class="col-lg-4">
68                                 <div class="form-group">
69                                     <div class="card-profile-image">
70                                         <a href="#">
71                                             {% if client.profile_pic %}
72                                                 
73                                             {% endif %}
74                                         </a>
75                                     </div>
76                                 </div>
77                             </div>
78                         </div>
79                     </div>
80                 </div>
81             <div class="row">
82                 <div class="col-lg-4">

```

```

83         <div class="form-group">
84             <label class="form-control-label" for="weight">Weight (kg)<
/label>
85             {{ form.weight(placeholder="Weight", class="form-control",
type="number", value=physicalInfo.weight, disabled=true)}}
86         </div>
87     </div>
88     <div class="col-lg-4">
89         <div class="form-group">
90             <label class="form-control-label" for="height">Height (cm)<
/label>
91             {{ form.height(placeholder="Height", class="form-control",
type="number", value=physicalInfo.height, disabled=true)}}
92         </div>
93     </div>
94     <div class="col-lg-4">
95         <div class="form-group">
96             <label class="form-control-label" for="register_date">Date of
register </label>
97             {{ form.register_date(placeholder="Register Date", class="
form-control", type="date", value=client.created_at.strftime("%Y-%m-%d"), readonly=
readonly)}}
98         </div>
99     </div>
100 </div>
101
102     <div class="row">
103         <div class="col-lg-6">
104             <div class="form-group">
105                 <label class="form-control-label" for="card_update">Card ID #
</label>
106                 {{ form.card_id(placeholder="Card ID", class="form-control",
type="text", value=client.card_id if client.card_id != None, disabled=true)}}
107                 <a href="{{ url_for('coach_blueprint.new_card', clientId=
client.id) }}" class="btn btn-sm btn-primary">Change</a>
108             </div>
109         </div>
110     </div>
111     <div class="col-lg-6">
112         <div class="form-group">
113             <label class="form-control-label" for="subscription_until">
Subscription Until</label>
114             {{ form.subscription_end(placeholder="Subscription Until",
class="form-control", type="date", value=subscriptionUntil, disabled=true)}}
115             <a href="{{ url_for('coach_blueprint.new_subscription',
clientId=client.id) }}" class="btn btn-sm btn-success">Renew subscription</a>
116             <a href="{{ url_for('coach_blueprint.cancel_subscription',
clientId=client.id) }}" class="btn btn-sm btn-danger">Cancel</a>
117         </div>
118     </div>
119 </div>
120 </div>
121 <hr class="my-4" />
122 <!-- Address -->
123 <h6 class="heading-small text-muted mb-4">Contact information</h6>
124 <div class="pl-lg-4">
125     <div class="row">
126         <div class="col-md-12">
127             <div class="form-group">
128                 <label class="form-control-label" for="address_update">Address</
label>
129                 {{ form.address(placeholder="Address", class="form-control", type
="text", value=client.address if client.address != None)}}
130             </div>
131         </div>
132     </div>
133     <div class="row">
134         <div class="col-lg-4">
135             <div class="form-group">
136                 <label class="form-control-label" for="city_update">City</label>
137                 {{ form.city(placeholder="City", class="form-control", type="text
", value=client.city if client.address != None)}}

```

```

138         </div>
139     </div>
140     <div class="col-lg-4">
141         <div class="form-group">
142             <label class="form-control-label" for="country_update">Country</
label>
143             {{ form.country(placeholder="Country", class="form-control", type
="text", value=client.country if client.country != None)}}
144         </div>
145     </div>
146     <div class="col-lg-4">
147         <div class="form-group">
148             <label class="form-control-label" for="input-country">Postal code
</label>
149             {{ form.npa(placeholder="Postal code", class="form-control", type
="number", value=client.npa if client.npa != None)}}
150         </div>
151     </div>
152 </div>
153 </div>
154 <hr class="my-4" />
155 <div class="row">
156     <div class="col-lg-12">
157         <div class="form-group">
158             {{ form.change(class="btn btn-primary btn-block") }}
159         </div>
160     </div>
161 </div>
162 </form>
163 </div>
164 </div>
165 </div>
166 <div class="card card-body h-auto" style="width: 18rem; float: right;">
167     <div class="card-header">
168         <div class="row align-items-center">
169             <div class="col-8">
170                 <h3 class="mb-0">Programs </h3>
171             </div>
172             <div class="col-4 text-right">
173                 <a href="{{ url_for('coach_blueprint.add_program', clientId=client.id) }}"
class="btn btn-sm btn-primary">Upload new program</a>
174             </div>
175         </div>
176     </div>
177     <div class="row">
178         <div class="col-lg-12">
179             <div class="form-group">
180                 {% if workoutProgram != None %}
181                 <label class="form-control-label">Workout Program added the {{
workoutProgram.date.strftime('%d/%m/%Y')}}: </label>
182                 </div>
183                 <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>
184                 {% else %}
185                 <h4>No workout plan available yet</h4>
186                 {% endif %}
187             </div>
188         </div>
189     </div>
190     <div class="row">
191         <div class="col-lg-12">
192             <div class="form-group">
193                 {% if dietProgram != None %}
194                 <label class="form-control-label">Diet Program added the {{dietProgram.
date.strftime('%d/%m/%Y')}}: </label>
195                 </div>
196                 <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>
197                 {% else %}
198                 <h4>No diet plan available yet</h4>

```

```

199         {% endif %}
200     </div>
201 </div>
202 </div>
203
204 <hr class="my-4" />
205 <div class="row">
206     <div class="col-8">
207         <h3 class="mb-0">Check up :</h3>
208     </div>
209
210 </div>
211 {% if checkUps.count() > 0 %}
212 <table class="table">
213     <thead>
214         <tr>
215             <th scope="col">Type</th>
216             <th scope="col">Date</th>
217             <th scope="col">Details</th>
218         </tr>
219     </thead>
220
221     {% for checkUp in checkUps %}
222     <tr>
223         <td>
224             {{ checkUp.date.strftime("%d/%m/%Y, %H:%M:%S") }}
225         </td>
226         <td>
227             {{ checkUp.weight }}KG
228         </td>
229         <td>
230             <a class="btn btn-sm btn-primary" href="{% url_for('client_blueprint.checkup', checkUpId=checkUp.id) %}">Details</a>
231         </td>
232     </tr>
233     {% endfor %}
234 </table>
235 {% else %}
236 <h4>No check up done yet</h4>
237 {% endif %}
238 </div>
239 </div>
240
241 <div class="row">
242     <div class="col-xl-8">
243         <div class="card">
244             <div class="card-header">
245                 <div class="row align-items-center">
246                     <div class="col-8">
247                         <h3 class="mb-0">Analytics {% if wrktTypeCount == 0 %} - No workout
done yet {% endif %}</h3>
248                     </div>
249                     <div class="col-4 text-right">
250                         <a href="#" class="btn btn-sm btn-warning">Statistic</a>
251                     </div>
252                 </div>
253             </div>
254             <div class="card-body">
255                 <div class="row">
256                     <div class="col-lg-4">
257                         <div class="chart-container" style="max-width: 25rem; position: relative;">
258
259                             <canvas id="workoutTypes"></canvas>
260
261                         </div>
262                     </div>
263                     <div class="col-lg-4">
264                         <div class="chart-container" style="max-width: 40rem; position: relative;">
265
266                             <canvas id="workoutCount"></canvas>
267
268                         </div>
269                     </div>
270                 </div>
271             </div>
272         </div>
273     </div>
274 </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 16zm7-8A7 7 0 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                 }

```

```

397     }
398   }
399   });
400   const workoutCountChart = new Chart(ctx2, {
401     type: 'line',
402     data: {
403       labels: [
404         'Jan',
405         'Feb',
406         'Mar',
407         'Apr',
408         'Mai',
409         'Jun',
410         'Jul',
411         'Aug',
412         'Sept',
413         'Oct',
414         'Nov',
415         'Dec'
416       ],
417       datasets: [{
418         label: 'Number of workouts',
419         data: {{nbWorkoutPerMonth}},
420         fill: false,
421         borderColor: 'rgb(128, 0, 128)',
422         tension: 0.1
423       }]
424     },
425     options: {
426       r: {
427         suggestedMin: 0
428       },
429       elements: {
430         line: {
431           borderWidth: 3
432         }
433       }
434     }
435   });
436   const workoutTypeChart = new Chart(ctx, {
437     type: 'doughnut',
438     data: {
439       labels: workoutList,
440       datasets: [{
441         label: 'Stats',
442         data: {{ wrktTypeCount }},
443         backgroundColor: [
444           'rgb(255, 99, 132)',
445           'rgb(54, 162, 235)',
446           'rgb(255, 205, 86)'
447         ],
448         hoverOffset: 4
449       }]
450     }
451   });
452   });
453
454   </script>
455   {% endblock javascripts %}

```

```

1   {% extends 'layout/base.html' %}
2
3   {% block title %} Payment {% endblock title %}
4
5   {% 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=
15         "Close">Close</button>
16 </div>
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             </div>
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                         
43
44                         {{nextClient.name}} {{nextClient.surname}}
45
46                     </h5>
47                 </div>
48             </div>
49
50             <div class="col-lg-4">
51
52                 <div class="card border-light mb-3" style="max-width: 36rem;">
53                     <div class="card-header">Date</div>
54                     <div class="card-body">
55
56                         {{nextClient.start_time.strftime("%d %B, %Y")}}
57
58                     </div>
59                 </div>
60
61                 <div class="col-lg-2">
62
63                     <div class="card border-light mb-3" style="max-width: 18rem;">
64                         <div class="card-header">Card ID</div>
65                         <div class="card-body">
66                             <h5 class="card-title">{{nextClient.card_id}}</h5>
67                         </div>
68                     </div>
69                 </div>
70             </div>
71
72             <div class="row justify-content-center">
73                 <div class="col-lg-2">
74                     <div class="card border-light mb-3" style="max-width: 18rem;">
75                         <div class="card-header">Session time</div>
76                         <div class="card-body">
77                             <h5 class="card-title">
78
79                                 {{nextClient.start_time.strftime("%H:%M:%S")}}
80
81                             </h5>
82                         </div>
83                     </div>
84                 </div>
85             </div>
86         </div>
87     </div>
88
89     <div class="col-lg-2">
90
91         <div class="card border-light mb-3" style="max-width: 18rem;">
92             <div class="card-header">Card ID</div>
93             <div class="card-body">
94                 <h5 class="card-title">{{nextClient.card_id}}</h5>
95             </div>
96         </div>
97
98         <div class="col-lg-4">
99
100             <div class="card border-light mb-3" style="max-width: 36rem;">
101                 <div class="card-header">Date</div>
102                 <div class="card-body">
103
104                     {{nextClient.start_time.strftime("%d %B, %Y")}}
105
106                 </div>
107             </div>
108
109             <div class="col-lg-2">
110
111                 <div class="card border-light mb-3" style="max-width: 18rem;">
112                     <div class="card-header">Card ID</div>
113                     <div class="card-body">
114                         <h5 class="card-title">{{nextClient.card_id}}</h5>
115                     </div>
116                 </div>
117             </div>
118
119             <div class="row justify-content-center">
120                 <div class="col-lg-2">
121                     <div class="card border-light mb-3" style="max-width: 18rem;">
122                         <div class="card-header">Session time</div>
123                         <div class="card-body">
124                             <h5 class="card-title">
125
126                                 {{nextClient.start_time.strftime("%H:%M:%S")}}
127
128                             </h5>
129                         </div>
130                     </div>
131                 </div>
132             </div>
133         </div>
134     </div>
135
136     <div class="col-lg-2">
137
138         <div class="card border-light mb-3" style="max-width: 18rem;">
139             <div class="card-header">Card ID</div>
140             <div class="card-body">
141                 <h5 class="card-title">{{nextClient.card_id}}</h5>
142             </div>
143         </div>
144
145         <div class="col-lg-4">
146
147             <div class="card border-light mb-3" style="max-width: 36rem;">
148                 <div class="card-header">Date</div>
149                 <div class="card-body">
150
151                     {{nextClient.start_time.strftime("%d %B, %Y")}}
152
153                 </div>
154             </div>
155
156             <div class="col-lg-2">
157
158                 <div class="card border-light mb-3" style="max-width: 18rem;">
159                     <div class="card-header">Card ID</div>
160                     <div class="card-body">
161                         <h5 class="card-title">{{nextClient.card_id}}</h5>
162                     </div>
163                 </div>
164             </div>
165
166             <div class="row justify-content-center">
167                 <div class="col-lg-2">
168                     <div class="card border-light mb-3" style="max-width: 18rem;">
169                         <div class="card-header">Session time</div>
170                         <div class="card-body">
171                             <h5 class="card-title">
172
173                                 {{nextClient.start_time.strftime("%H:%M:%S")}}
174
175                             </h5>
176                         </div>
177                     </div>
178                 </div>
179             </div>
180         </div>
181     </div>

```

```

82         <div class="card border-light mb-3" style="max-width: 18rem;">
83             <div class="card-header">Duration</div>
84             <div class="card-body">
85                 <h5 class="card-title">
86
87                     {{nextClient.duration}}
88
89                 </h5>
90             </div>
91         </div>
92     </div>
93     <div class="col-lg-2">
94         <div class="card border-light mb-3" style="max-width: 18rem;">
95             <div class="card-header">Type</div>
96             <div class="card-body">
97                 <h5 class="card-title">
98
99                     
100
101                 </h5>
102             </div>
103         </div>
104     </div>
105     <div class="col-lg-2">
106         <div class="card border-light mb-3" style="max-width: 18rem;">
107             <div class="card-header">Last Workout </div>
108             <div class="card-body">
109                 <h5 class="card-title"> {{clientLastWorkout.date.strftime("%d/%
m/%Y, %H:%M:%S")} if clientLastWorkout != None else "No workout yet"}</h5>
110             </div>
111         </div>
112     </div>
113 </div>
114 {% else %}
115 <h2 align="center">No session registered</h2>
116 {% endif %}
117 </div>
118 </div>
119 </div>
120
121 <div class="row" style="margin:auto;">
122     <div class="col-xl-12" >
123         <div class="card" style="max-width: 114rem;">
124             <div class="card-header">
125                 <div class="row align-items-center">
126                     <div class="col-8">
127                         <h3 class="mb-0">Clients</h3>
128                     </div>
129                     <div class="col-4 text-right">
130                         <a href="{{ url_for('coach_blueprint.add_client') }}" class="btn btn-sm
btn-primary">Add new client</a>
131                     </div>
132                 </div>
133             </div>
134
135             <table class="table">
136                 <thead>
137                     <tr>
138                         <th scope="col"></th>
139                         <th scope="col">Name</th>
140                         <th scope="col">Subscription end date</th>
141                         <th scope="col">Last Workout </th>
142                         <th scope="col">Card ID</th>
143                         <th scope="col">Active (Last 24h)</th>
144                     </tr>
145                 </thead>
146                 <tbody>
147                     {% for client in clients %}
148
149                     <tr>

```

```

150         <td>
151             
154         </td>
155         <td>
156             <a href="{{ url_for('coach_blueprint.client', clientId=client.id) }}"
157                 class="btn btn-sm btn-primary"> {{client.name}} {{client.surname}}</a>
158         </td>
159         <td>
160             {{client.subscriptionEnd.strftime("%d/%m/%Y")}}
161         </td>
162         <td>
163             {{client.lastWorkout.date.strftime("%d/%m/%Y, %H:%M:%S") if client.
164                 lastWorkout != None else "No workout yet"}}
165         </td>
166         <td>
167             {{client.card_id}}
168         </td>
169         <td>
170             {% if client.isActive %}
171             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="
172                 currentColor" class="bi bi-check" viewBox="0 0 16 16">
173                 <path d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0
174                 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.093 3.473-4.425a.267.267 0 0 1
175                 .02-.022z"/>
176             </svg>
177             {% endif %}
178         </td>
179     </tr>
180     {% endfor %}
181 </table>
182 </div>
183 </div>
184 {% endblock content %}
185 <!-- Specific JS goes HERE -->
186 {% block javascripts %}
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

25         <th scope="col">Date</th>
26         <th scope="col">Duration</th>
27         <th scope="col">Hour</th>
28         <th scope="col">Notif</th>
29     </tr>
30 </thead>
31 {% for session in nextSessions %}
32 <br>
33 <tr>
34     <td>
35         {% if session.logo %}
36         
38         {% endif %}
39     </td>
40     <td>
41         {{ session.title }}
42     </td>
43     <td>
44         {{ session.start_time.strftime('%d/%m/%Y') }}
45     </td>
46     <td>
47         {{ session.duration }}
48     </td>
49     <td>
50         {{ session.start_time.strftime('%X') }}
51     </td>
52     <td>
53         <button type="button" class="btn btn-primary">
54             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="
55             currentColor" class="bi bi-bell" viewBox="0 0 16 16">
56                 <path d="M8 16a2 2 0 0 0 2-2H6a2 2 0 0 0 2 2zM8 1.918l-.797 1.61A4.002
57                 4.002 0 0 0 4 6c0 .628-.134 2.197-.459 3.742-.16767-.376 1.566-.663 2.258h10.244c
58                 -.287-.692-.502-1.49-.663-2.258C12.134 8.197 12 6.628 12 6a4.002 4.002 0 0
59                 0-3.203-3.92L8 1.917zM14.22 12c.223.447.481.801.78 1H1c.299-.199.557-.553.78-1C2.68
60                 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
61                 6c0 .88.32 4.2 1.22 6z"></path>
62             </svg>
63         </button>
64     </td>
65 </tr>
66 {% endfor %}
67 </table>
68 </div>
69 </div>
70 {% endblock content %}
71 {% block javascripts %}
72 {% endblock javascripts %}
73
74 1 {% extends 'layout/base.html' %}
75 2
76 3 {% block title %} Profile {% endblock title %}
77 4
78 5 {% block content %}
79 6
80 7
81 8
82 9 <div class="container-fluid mt--8" style="margin: auto;">
83 10     {% for category, message in get_flashed_messages(with_categories=true) %}
84 11
85 12     <div class="alert alert-{{category}} alert-dismissible fade show" role="alert">
86 13         {{ message }}
87 14         <button type="button" class="btn btn-secondary" data-bs-dismiss="alert" aria-label=
88         "Close">Close</button>
89 15     </div>
90 16
91 17 {% endfor %}
92 18
93 19
94 20

```

```

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<
44                                     /label>
45                                     {{ form.name(placeholder="Name", class="form-control", type="text", value= current_user.name) }}
46                                 </div>
47                             </div>
48                             <div class="col-lg-4">
49                                 <div class="form-group">
50                                     <label class="form-control-label" for="surname_update">Last
51                                     name</label>
52                                     {{ form.surname(placeholder="Surname", class="form-control", type="text", value=current_user.surname)}}
53                                 </div>
54                             </div>
55                             <div class="col-lg-4">
56                                 <div class="form-group">
57                                     <div class="card-profile-image">
58                                         <a href="#">
59                                             {% if current_user.profile_pic %}
60
61                                             
62
63                                             {% else %}
64
65                                             
66
67                                             {% endif %}
68                                         </a>
69                                     </div>
70                                 </div>
71                             </div>
72                         </div>
73                         <div class="row">
74                             <div class="col-lg-4">
75                                 <div class="form-group">
76                                     <label class="form-control-label" for="birthdate_update">
77                                     Birthdate</label>
78                                     {{ form.birthdate(placeholder="Birthdate", class="form-control", type="date", value=current_user.birthdate)}}
79                                 </div>
80                             </div>
81                             <div class="col-lg-4">
82                                 <div class="form-group">
83                                     <label class="form-control-label" for="email_update">Email
84                                     address</label>
85                                     {{ form.email(placeholder="Email", class="form-control", type="text", value=current_user.email)}}
86                                 </div>
87                             </div>
88                         </div>
89                     </div>
90                 </form>
91             </div>
92         </div>
93     </div>
94 </div>

```

```

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
88                 ="currentColor" class="bi bi-file-earmark-person" viewBox="0 0 16 16">
89                     <path d="M11 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>
90                     <path d="M14 14V4.5L9.5 0H4a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h8a2 2
91                     0 0 0 2-2zM9.5 3A1.5 1.5 0 0 0 11 4.5h2v9.25S12 12 8 12s-5 1.755-5 1.755V2a1 1 0 0
92                     1 1-1h5.5v2z"/>
93                 </svg>
94             </label>
95             {{ form.profile_pic(class="form-control", type="file") }}
96         </div>
97     </div>
98     <div>
99         {% if current_user.role == 1%}
100         <div class="row">
101             <div class="col-lg-4">
102                 <div class="form-group">
103                     <label class="form-control-label" for="weight_update">
104                         Weight (kg)</label>
105                         {{ form.weight(placeholder="Weight", class="form-control",
106                         type="number", value=current_user.physicalInfo.weight, disabled=true)}}
107                     </div>
108                 </div>
109             <div class="col-lg-4">
110                 <div class="form-group">
111                     <label class="form-control-label" for="height_update">
112                         Height (cm)</label>
113                         {{ form.height(placeholder="Height", class="form-control",
114                         type="number", value=current_user.physicalInfo.height, disabled=true)}}
115                     </div>
116                 </div>
117             </div>
118             <div class="row">
119                 <div class="col-lg-6">
120                     <div class="form-group">
121                         <label class="form-control-label" for="card_update">Card ID
122                         #</label>
123                         {{ form.card_id(placeholder="Card ID", class="form-control"
124                         , type="text", value=current_user.card_id if current_user.card_id != None, disabled
125                         =true)}}
126                     </div>
127                 </div>
128                 <div class="col-lg-6">
129                     <div class="form-group">
130                         <label class="form-control-label" for="subscription_until">
131                         Subscription Until</label>
132                         {{ form.subscriptionEnd(placeholder="Subscription Until",
133                         class="form-control", value=current_user.subscriptionEnd if current_user.
134                         subscriptionEnd != None, disabled=true)}}
135                     </div>
136                 </div>
137             </div>
138             <div class="row">
139                 <div class="col-md-12">
140                     <div class="form-group">
141                         <label class="form-control-label" for="address_update">Address</
142                         label>
143                         {{ form.address(placeholder="Address", class="form-control", type
144                         ="text", value=current_user.address if current_user.address != None)}}
145                     </div>

```

```

138         </div>
139     </div>
140     <div class="row">
141         <div class="col-lg-4">
142             <div class="form-group">
143                 <label class="form-control-label" for="city_update">City</label>
144                 {{ form.city(placeholder="City", class="form-control", type="text", value=current_user.city if current_user.city != None)}}
145             </div>
146         </div>
147         <div class="col-lg-4">
148             <div class="form-group">
149                 <label class="form-control-label" for="country_update">Country</label>
150                 {{ form.country(placeholder="Country", class="form-control", type="text", value=current_user.country if current_user.country != None)}}
151             </div>
152         </div>
153         <div class="col-lg-4">
154             <div class="form-group">
155                 <label class="form-control-label" for="input-country">Postal code
156                 </label>
157                 {{ form.npa(placeholder="Postal code", class="form-control", type="number", value=current_user.npa if current_user.npa != None)}}
158             </div>
159         </div>
160     </div>
161     <hr class="my-4" />
162     <div class="pl-lg-4">
163         <div class="row">
164             <div class="col-md-12">
165                 <div class="form-group">
166                     {{ form.update(class="btn btn-primary btn-block") }}
167                     <a class="btn btn-sm btn-warning" href="{{url_for('client_blueprint.change_password')}}">Change password</a>
168                 </div>
169             </div>
170         </div>
171     </div>
172 </form>
173 </div>
174 </div>
175 </div>
176
177 {% if current_user.role == 1%}
178 <div class="card card-body h-auto" style="width: 18rem; float: right;">
179     <div class="card-header">
180         <div class="row align-items-center">
181             <div class="col-8">
182                 <h3 class="mb-0">Programs </h3>
183             </div>
184             <div class="col-4 text-right">
185             </div>
186         </div>
187     </div>
188     <div class="row">
189         <div class="col-lg-12">
190             <div class="form-group">
191                 {% if current_user.workoutProgram != None %}
192                 <label class="form-control-label">Workout Program added the {{
193                 current_user.workoutProgram.date.strftime('%d/%m/%Y')}}: </label>
194                 <br>
195                 <a href="{{ url_for('coach_blueprint.program', programId=current_user.workoutProgram.id, programType="workout") }}" class="btn btn-sm btn-warning">
196                 Workout Program</a> added by <strong>{{current_user.workoutProgram.name}} {{
197                 current_user.workoutProgram.surname}}</strong>
198                 {% else %}
199                 <h4>No workout plan available yet</h4>
200                 {% endif %}
201             </div>
202         </div>
203     </div>
204 </div>

```

```

200     </div>
201     <div class="row">
202         <div class="col-lg-12">
203             <div class="form-group">
204                 {% if current_user.dietProgram != None %}
205                 <label class="form-control-label">Diet Program added the {{current_user
206                 .dietProgram.date.strftime('%d/%m/%Y')}}: </label>
207                 </br>
208                 <a href="{ url_for('coach_blueprint.program', programId=current_user.
209                 dietProgram.id, programType="diet") }}" class="btn btn-sm btn-success">Diet Program
210                 </a> added by <strong>{{current_user.dietProgram.name}} {{current_user.dietProgram.
211                 surname}}</strong>
212                 {% else %}
213                 <h4>No diet plan available yet</h4>
214                 {% endif %}
215             </div>
216         </div>
217     </div>
218     <hr class="my-4" />
219     <div class="row">
220         <div class="col-8">
221             <h3 class="mb-0">Check up :</h3>
222         </div>
223     </div>
224     {% if current_user.checkUps.count() >0 %}
225     <table class="table">
226         <thead>
227             <tr>
228                 <th scope="col">Type</th>
229                 <th scope="col">Date</th>
230                 <th scope="col">Details</th>
231             </tr>
232         </thead>
233         <tbody>
234             {% for checkUp in current_user.checkUps %}
235             <tr>
236                 <td>
237                     {{ checkUp.date.strftime("%d/%m/%Y, %H:%M:%S") }}
238                 </td>
239                 <td>
240                     {{ checkUp.weight }}KG
241                 </td>
242                 <td>
243                     <a class="btn btn-sm btn-primary" href="{ url_for('client_blueprint.
244                     checkup', checkUpId=checkUp.id) }}">Details</a>
245                 </td>
246             </tr>
247             {% endfor %}
248         </tbody>
249     </table>
250     {%else%}
251     <h4>No check up done yet</h4>
252     {%endif%}
253 </div>
254 </div>
255 <div class="row">
256     <div class="col-xl-8">
257         <div class="card">
258             <div class="card-header">
259                 <div class="row align-items-center">
260                     <div class="col-8">
261                         <h3 class="mb-0">Analytics {% if current_user.workoutTypeCount == 0 %}
262                         - No workout done yet {%endif%}</h3>
263                     </div>
264                     <div class="col-4 text-right">
265                         <a href="#" class="btn btn-sm btn-warning">Statistic</a>
266                     </div>

```



```

266         </div>
267     </div>
268     <div class="card-body">
269         <div class="row">
270             <div class="col-lg-4">
271                 <div class="chart-container" style="max-width: 25rem; position: relative;">
272                     <canvas id="workoutTypes" ></canvas>
273                 </div>
274             </div>
275             <div class="col-lg-4">
276                 <div class="chart-container" style="max-width: 40rem; position: relative;
277             ">
278                 <canvas id="workoutCount"></canvas>
279             </div>
280             <div class="col-lg-4">
281                 <div class="card" style="max-width: 12rem;">
282                     <div class="card-header">Calories Burn
283                     <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 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>
284
285
286                     </div>
287                     <div class="card-body">
288                         <h5 class="card-title">{{ current_user.avgCalories }}</h5>
289                     </div>
290                 </div>
291                 <div class="card" style="max-width: 10rem;">
292                     <div class="card-header">Heart rate avg
293                     <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">
294                         <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"/>
295                     </svg>
296
297                     </div>
298                     <div class="card-body">
299                         <h5 class="card-title">{{ current_user.avgHeartRate }}</h5>
300                     </div>
301                 </div>
302                 <div class="card" style="max-width: 10rem;">
303                     <div class="card-header">Time working out (min)
304                     <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill
="currentColor" class="bi bi-clock" viewBox="0 0 16 16">
305                         <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 7.71V3.5z"/>
306                         <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 16zm7-8A7 7 0 1 1 8a7
7 0 0 1 14 0z"/>
307                     </svg>
308
309                     </div>
310                     <div class="card-body">
311                         <h5 class="card-title">{{ current_user.totalTime }}</h5>
312                     </div>
313                 </div>
314             </div>
315         </div>
316     <div class="row">
317         <div class="col-lg-12">
318             <div class="chart-container" style="max-width: 40rem; position: relative;
319             ">
320                 <canvas id="weightUpdate"></canvas>
321             </div>
322         </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],
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',

```

```

391         'Jul',
392         'Aug',
393         'Sept',
394         'Oct',
395         'Nov',
396         'Dec'
397     ],
398     datasets: [{
399         label: 'Weight update',
400         data: {{current_user.weightUpdate}},
401         fill: false,
402         borderColor: 'rgb(0, 128, 128)',
403         tension: 0.1
404     }]
405 },
406 options: {
407     scales: {
408         y: {
409             min: 40,
410             suggestedMin: 50,
411             suggestedMax: 100
412         }
413     },
414     elements: {
415         line: {
416             borderWidth: 3
417         }
418     }
419 }
420 });
421 const myChart2 = new Chart(ctx2, {
422     type: 'line',
423     data: {
424         labels: [
425             'Jan',
426             'Feb',
427             'Mar',
428             'Apr',
429             'Mai',
430             'Jun',
431             'Jul',
432             'Aug',
433             'Sept',
434             'Oct',
435             'Nov',
436             'Dec'
437         ],
438         datasets: [{
439             label: 'Number of workouts',
440             data: {{current_user.nbWorkoutPerMonth}},
441             fill: false,
442             borderColor: 'rgb(128, 0, 128)',
443             tension: 0.1
444         }]
445     },
446     options: {
447         r: {
448             suggestedMin: 0
449         },
450         elements: {
451             line: {
452                 borderWidth: 3
453             }
454         }
455     }
456 });
457 const myChart = new Chart(ctx, {
458     type: 'doughnut',
459     data: {
460         labels: workoutList,
461         datasets: [{
462             label: 'Stats',

```

```

463         data: {{ current_user.workoutTypeCount }},
464         backgroundColor: [
465             'rgb(255, 99, 132)',
466             'rgb(54, 162, 235)',
467             'rgb(255, 205, 86)',
468             'rgb(128, 0, 128)',
469             'rgb(0,128,128)',
470             'rgb(128,128,0)'
471         ],
472         hoverOffset: 4
473     ]}
474 }
475 });
476 {%endif%}
477 </script>
478 {%endif%}
479 {% endblock javascripts %}

1  {% extends 'layout/base.html' %}
2
3  {% block title %} Login {% endblock title %}
4
5  <!-- Specific CSS HERE -->
6  {% block stylesheets %}{% endblock stylesheets %}
7
8  {% block content %}
9
10
11  <!-- Page content -->
12  <div class="container mt--8 pb-5">
13      <div class="row justify-content-center">
14          <div class="col-lg-5 col-md-7">
15              <div class="card bg-secondary shadow border-0">
16                  <div class="card-header bg-transparent pb-5">
17
18                      <div class="text-muted text-center mt-2 mb-3">
19                          LOGIN
20                      <br />
21                      {% if msg %}
22                          <span class="text-danger">{{ msg | safe }}</span>
23                      {% else %}
24                          <span>
25                              Add your credentials
26                          </span>
27                      {% endif %}
28                  </div>
29
30                  </div>
31                  {% for category, message in get_flashed_messages(with_categories=true) %}
32
33                      <div class="alert alert-{{category}} alert-dismissible fade show" role="alert
34                          ">
35                          {{ message }}
36                          <button type="button" class="btn btn-secondary" data-bs-dismiss="alert"
37                          aria-label="Close">Close</button>
38                      </div>
39
40                      {% endfor %}
41                      <div class="card-body px-lg-5 py-lg-5">
42
43                          <form role="form" method="post" action="">
44
45                              {{ form.hidden_tag() }}
46
47                              <div class="form-group mb-3">
48                                  <div class="input-group input-group-alternative">
49                                      <div class="input-group-prepend">
50                                          <span class="input-group-text"><i class="ni ni-single-02"></i></
span>
51                                  </div>
52                                  {{ form.email(placeholder="Email",class="form-control") }}

```

```

51         </div>
52     </div>
53     <div class="form-group">
54         <div class="input-group input-group-alternative">
55             <div class="input-group-prepend">
56                 <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
57             </div>
58             {{ form.password(placeholder="Password",class="form-control",type="
password") }}
59         </div>
60     </div>
61
62     <div class="text-center">
63         {{ form.login(class="btn btn-primary my-4") }}
64     </div>
65 </form>
66 </div>
67 </div>
68 <div class="row mt-3 text-center">
69     <div class="col-12">
70         <a href="{ url_for('authentication_blueprint.register') }}" class="text-
light"><small>Create new account </small></a>
71     </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 {% endblock content %}
77
78 <!-- Specific JS goes HERE -->
79 {% block javascripts %}{% endblock javascripts %}

1 <nav class="sidenav navbar navbar-vertical fixed-left navbar-expand-xs navbar-light
bg-white" id="sidenav-main">
2     <div class="scrollbar-inner">
3         <!-- Brand -->
4         <div class="sidenav-header align-items-center">
5             <a class="navbar-brand" href="/">
6                 <span class="nav-link-text">FitJourney</span>
7             </a>
8         </div>
9         <div class="navbar-inner">
10            <!-- Collapse -->
11            <div class="collapse navbar-collapse" id="sidenav-collapse-main">
12                <!-- Nav items -->
13                <ul class="navbar-nav">
14                    {% if not current_user.is_authenticated %}
15                    <li class="nav-item">
16                        <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
="{ url_for('authentication_blueprint.login') }}">
17                            <i class="ni ni-single-02 text-yellow"></i>
18                            <span class="nav-link-text">Login</span>
19                        </a>
20                    </li>
21                    <li class="nav-item">
22                        <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
="{ url_for('authentication_blueprint.register') }}">
23                            <i class="ni ni-single-02 text-yellow"></i>
24                            <span class="nav-link-text">Register</span>
25                        </a>
26                    </li>
27                    {% elif current_user.is_authenticated and current_user.role == 1 %}
28                    <li class="nav-item">
29                        <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
="{ url_for('client_blueprint.profile') }}">
30                            <i class="ni ni-single-02 text-yellow"></i>
31                            <span class="nav-link-text">Profile</span>
32                        </a>
33                    </li>
34                    <li class="nav-item">
35                        <a class="nav-link {% if 'index' in segment %} active {% endif %}" href="

```

```

36     <i class="ni ni-calendar-grid-58 text-orange"></i>
37     <span class="nav-link-text">Next Sessions</span>
38   </a>
39 </li>
40 <li class="nav-item">
41   <a class="nav-link {% if 'workouts' in segment %} active {% endif %}"
42   href="{% url_for('client_blueprint.workouts') %}">
43     <i class="ni ni-planet text-orange"></i>
44     <span class="nav-link-text">Workouts</span>
45   </a>
46   {% elif current_user.is_authenticated and current_user.role == 2 %}
47   <li class="nav-item">
48     <a class="nav-link {% if 'dashboard' in segment %} active {% endif %}"
49     href="{% url_for('coach_blueprint.dashboard') %}">
50       <i class="ni ni-tv-2 text-primary"></i>
51       <span class="nav-link-text">Dashboard</span>
52     </a>
53   </li>
54   <li class="nav-item">
55     <a class="nav-link {% if 'profile' in segment %} active {% endif %}" href
56     ="{% url_for('client_blueprint.profile') %}">
57       <i class="ni ni-single-02 text-yellow"></i>
58       <span class="nav-link-text">Profile</span>
59     </a>
60   </li>
61   <li class="nav-item">
62     <a class="nav-link {% if 'calendar' in segment %} active {% endif %}"
63     href="{% url_for('coach_blueprint.calendar') %}">
64       <i class="ni ni-calendar-grid-58 text-orange"></i>
65       <span class="nav-link-text">Calendar</span>
66     </a>
67   </li>
68   {% endif %}
69 </ul>
70 <!-- Divider -->
71 <hr class="my-3">
72 {% if current_user.is_authenticated %}
73 <!-- Heading -->
74 <ul class="navbar-nav">
75 <li class="nav-item">
76   <a class="nav-link" href="{% url_for('authentication_blueprint.logout') %}"
77   >
78     <i class="ni ni-user-run text-red"></i><i class="fa fa-sign-out" aria-
79     hidden="true"></i>
80     <span class="nav-link-text">Logout</span>
81   </a>
82 </li>
83 </ul>
84 {% endif %}
85 </div>
86 </div>
87 </nav>

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Renew subscription {% endblock title %}
4
5 {% 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="
15   Close">Close</button>
16 </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
37 .id)}}" method="POST">
38                     {{ form.hidden_tag() }}
39                     <div class="pl-lg-4">
40                         <div class="row justify-content-center">
41                             <div class="col-lg-4">
42                                 <div class="form-group">
43                                     <label class="form-control-label" for="name">Name</label>
44                                     {{form.name(placeholder="Name", class="form-control", type="
45 text", value=client.name, readonly=readonly)}}
46                                 </div>
47                                 </div>
48                                 <div class="col-lg-4">
49                                     <div class="form-group">
50                                         <label class="form-control-label" for="surname">Surname</
51 label>
52                                         {{form.surname(placeholder="Surname", class="form-control",
53 type="text", value=client.surname, readonly=readonly)}}
54                                     </div>
55                                     </div>
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=
60 "email", value=client.email, readonly=readonly)}}
61                                         </div>
62                                         </div>
63                                     <div class="col-lg-4">
64                                         <div class="form-group">
65                                             <label class="form-control-label" for="start_date">
66 Starting date</label>
67                                             {{form.start_date(class="form-control", onchange="
68 updateEndTime()")}}
69                                         </div>
70                                         </div>
71                                     <div class="col-lg-4">
72                                         <div class="form-group">
73                                             <label class="form-control-label" for="subscription">
74 Subscription type</label>
75                                             {{form.subscription(class="form-control", onchange="
76 updateEndTime()")}}
77                                         </div>
78                                         </div>
79                                     <div class="col-lg-4">
80                                         <div class="form-group">
81                                             <label class="form-control-label" for="end_date">
82 Subscription until</label>
83                                             {{form.end_date(class="form-control", readonly=readonly)

```

```

78     }}
79         </div>
80     </div>
81
82     <div class="row">
83         <div class="col-md-12">
84             <div class="form-group">
85                 {{form.submit(class="form-control btn btn-primary")}}
86             </div>
87         </div>
88     </div>
89 </div>
90 </form>
91 </div>
92 </div>
93 </div>
94
95
96
97 {% endblock content %}
98
99 <!-- Specific JS goes HERE -->
100 {% block javascripts %}
101 <script>
102     function updateEndTime(){
103         var subscription = document.getElementById("subscription").value;
104         var start_date = document.getElementById("start_date").value;
105
106         if(subscription != "" && start_date != ""){
107             //Check if the fields aren't empty, then set a end time depends on values
108             selected before
109             var dt = new Date(start_date);
110             dt.setMonth(dt.getMonth() + parseInt(subscription));
111             let [end_date] = dt.toISOString().split('T');
112             document.getElementById("end_date").value = end_date
113         }
114     }
115 </script>
116 {% endblock javascripts %}

```

```

1  {% extends 'layout/base.html' %}
2
3  {% block title %} Error 403 {% endblock title %}
4
5  <!-- Specific CSS goes HERE -->
6  {% block stylesheets %}{% endblock stylesheets %}
7
8  {% block content %}
9
10     <!-- Page content -->
11     <div class="container mt--8 pb-5">
12         <div class="row justify-content-center">
13             <div class="col-lg-5 col-md-7">
14                 <div class="card bg-secondary shadow border-0">
15                     <div class="card-header bg-transparent pb-5">
16
17                         <div class="text-muted text-center mt-2 mb-3">
18                             Oops! Error 403
19                         </div>
20
21                         <div class="text-center mb-4">
22                             Access denied - Please authenticate <a href="{{ url_for('
authentication_blueprint.login' ) }}">Login</a>
23                             USER : {{ current_user }}
24                         </div>
25
26                     </div>
27                 </div>
28             </div>
29         </div>

```



```

30     </div>
31
32 {% endblock content %}
33
34 <!-- Specific JS goes HERE -->
35 {% block javascripts %}{% endblock javascripts %}

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Error 404 {% endblock title %}
4
5 <!-- Specific CSS goes HERE -->
6 {% block stylesheets %}{% endblock stylesheets %}
7
8 {% block content %}
9
10     <!-- Page content -->
11     <div class="container mt--8 pb-5">
12         <div class="row justify-content-center">
13             <div class="col-lg-5 col-md-7">
14                 <div class="card bg-secondary shadow border-0">
15                     <div class="card-header bg-transparent pb-5">
16
17                         <div class="text-muted text-center mt-2 mb-3">
18                             Oops! Error 404
19                         </div>
20
21                         <div class="text-center mb-4">
22                             Page not found - <a href="/">Home</a>
23                         </div>
24
25                     </div>
26                 </div>
27             </div>
28         </div>
29     </div>
30
31 {% endblock content %}
32
33 <!-- Specific JS goes HERE -->
34 {% block javascripts %}{% endblock javascripts %}

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Error 404 {% endblock title %}
4
5 <!-- Specific CSS goes HERE -->
6 {% block stylesheets %}{% endblock stylesheets %}
7
8 {% block content %}
9
10     <!-- Page content -->
11     <div class="container mt--8 pb-5">
12         <div class="row justify-content-center">
13             <div class="col-lg-5 col-md-7">
14                 <div class="card bg-secondary shadow border-0">
15                     <div class="card-header bg-transparent pb-5">
16
17                         <div class="text-muted text-center mt-2 mb-3">
18                             Oops! Error 500
19                         </div>
20
21                         <div class="text-center mb-4">
22                             Server error, please contact support - <a href="/">Home</a>.
23                         </div>
24
25                     </div>
26                 </div>
27             </div>
28         </div>
29     </div>
30

```

```

31 {% endblock content %}
32
33 <!-- Specific JS goes HERE -->
34 {% block javascripts %}{% endblock javascripts %}

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Register {% endblock title %}
4
5 <!-- Specific CSS HERE -->
6 {% block stylesheets %}{% endblock stylesheets %}
7
8 {% block content %}
9
10
11 <!-- Page content -->
12 <div class="container mt--8 pb-5">
13     <div class="row justify-content-center">
14         <div class="col-lg-5 col-md-7">
15             <div class="card bg-secondary shadow border-0">
16                 <div class="card-header bg-transparent pb-5">
17
18                     <div class="text-muted text-center mt-2 mb-3">
19                         REGISTER
20                     <br />
21                     {% if msg and success%}
22                     <span class="text-success">{{ msg | safe }}</span>
23                     {% else %}
24                         Add your credentials
25                     {% endif %}
26
27                     {% for error in form.password.errors %} <p class="text-danger">{{ error |
safe }}</p> {% endfor%}
28                 </div>
29
30                 </div>
31                 {% for category, message in get_flashed_messages(with_categories=true) %}
32
33                 <div class="alert alert-{{category}} alert-dismissible fade show" role="alert
">
34                     {{ message }}
35                     <button type="button" class="btn btn-secondary" data-bs-dismiss="alert "
aria-label="Close">Close</button>
36                 </div>
37
38                 {% endfor %}
39                 <div class="card-body px-lg-5 py-lg-5">
40
41                     {% if not success %}
42
43                     <form role="form" method="post" action="">
44
45                         {{ form.hidden_tag() }}
46
47                         <div class="form-group mb-3">
48                             <div class="input-group input-group-alternative">
49                                 <div class="input-group-prepend">
50                                     <span class="input-group-text"><i class="ni ni-single-02"></i></
span>
51
52                                     </div>
53                                     {{ form.name(placeholder="Name",class="form-control") }}
54                                 </div>
55                             </div>
56                             <div class="form-group mb-3">
57                                 <div class="input-group input-group-alternative">
58                                     <div class="input-group-prepend">
59                                         <span class="input-group-text"><i class="ni ni-single-02"></i></
span>
60
61                                         </div>
62                                         {{ form.surname(placeholder="Surname",class="form-control") }}
63                                     </div>

```

```

63         <div class="form-group mb-3">
64             <div class="input-group input-group-alternative">
65                 <div class="input-group-prepend">
66                     <span class="input-group-text"><i class="ni ni-email-83"></i></span>
67                 </div>
68                 {{ form.email(placeholder="Email",class="form-control") }}
69             </div>
70         </div>
71         <div class="form-group mb-3">
72             <div class="input-group input-group-alternative">
73                 <div class="input-group-prepend">
74                     <span class="input-group-text"><i class="ni ni-single-02"></i></span>
75                 </div>
76                 {{ form.birthdate(placeholder="Date of Birth",class="form-control") }}
77             </div>
78         </div>
79         <div class="form-group">
80             <div class="input-group input-group-alternative">
81                 <div class="input-group-prepend">
82                     <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
83                 </div>
84                 {{ form.password(placeholder="Password",class="form-control") }}
85             </div>
86         </div>
87         <div class="form-group">
88             <div class="input-group input-group-alternative">
89                 <div class="input-group-prepend">
90                     <span class="input-group-text"><i class="ni ni-lock-circle-open"></i></span>
91                 </div>
92                 {{ form.confirm_password(placeholder="Confirm Password",class="form-control") }}
93             </div>
94         </div>
95     </div>
96     <div class="text-center">
97         {{ form.register(class="btn btn-primary my-4") }}
98     </div>
99 </form>
100 {% endif %}
101
102 </div>
103 </div>
104 <div class="row mt-3 text-center">
105     <div class="col-12">
106         <a href="{% url_for('authentication_blueprint.login') %}" class="text-light">
107             <small>Use existing account</small></a>
108     </div>
109 </div>
110 </div>
111 </div>
112 </div>
113 {% endblock content %}
114
115 <!-- Specific JS goes HERE -->
116 {% block javascripts %}{% endblock javascripts %}

```

```

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Review {% endblock title %}
4
5 {% 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=
15         "Close">Close</button>
16 </div>
17
18 {% endfor %}
19 <div class="row">
20     <div class="card card-body h-auto" style="width: 18rem; float: right;">
21         <h5 class="card-header">Review added by {{client.name}} {{client.surname}}</h5>
22         <div class="row justify-content-center">
23
24             <div class="col-lg-2">
25                 <div class="card border-light mb-3" style="max-width: 18rem;">
26                     <div class="card-header">Date</div>
27                     <div class="card-body">
28                         <h5 class="card-title">{{review.date}}</h5>
29                     </div>
30                 </div>
31             </div>
32             <div class="col-lg-4">
33
34                 <div class="card border-light mb-3" style="max-width: 36rem;">
35                     <div class="card-header">Review on</div>
36                     <div class="card-body">
37                         {% if review.type == 'WORKOUT' %}
38                         <h5 class="card-title">{{target['Type']} + " - " + target['Date']}.
39                         strftime("%m/%d/%Y, %H:%M:%S") + " - " + target['Duration'].strftime("%X")}</h5>
40                         {% elif review.type == 'COACHING' %}
41                         <h5 class="card-title">{{target['Name']} + " " + target['Surname']}</h5>
42                         {% endif %}
43                     </div>
44                 </div>
45             </div>
46             <div class="col-lg-2">
47
48                 <div class="card border-light mb-3" style="max-width: 18rem;">
49                     <div class="card-header">Type</div>
50                     <div class="card-body">
51                         <h5 class="card-title">{{review.type}}</h5>
52                     </div>
53                 </div>
54             </div>
55         </div>
56         <div class="row justify-content-center">
57             <div class="col-lg-2">
58                 <div class="card border-light mb-3" style="max-width: 18rem;">
59                     <div class="card-header">{{"Satisfaction" if review['Type'] == '
60                     COACHING' else "Difficulty"}}</div>
61                     <div class="card-body">
62                         <h5 class="card-title">{{review.Field1}}/10</h5>
63                     </div>
64                 </div>
65             </div>
66             <div class="col-lg-2">
67                 <div class="card border-light mb-3" style="max-width: 18rem;">
68                     <div class="card-header">{{"Support" if review['Type'] == 'COACHING
69                     ' else "Feel"}}</div>
70                     <div class="card-body">
71                         <h5 class="card-title">{{review.Field2}}/10</h5>
72                     </div>
73                 </div>
74             </div>
75             <div class="col-lg-2">
76                 <div class="card border-light mb-3" style="max-width: 18rem;">
77                     <div class="card-header">{{"Disponibility" if review['Type'] == '
78                     COACHING' else "Fatigue (at the end)"}</div>
79                     <div class="card-body">

```

```

76         <h5 class="card-title">{{review.Field3}}/10</h5>
77     </div>
78 </div>
79 </div>
80 <div class="col-lg-2">
81     <div class="card border-light mb-3" style="max-width: 18rem;">
82         <div class="card-header">{{"Advice quality" if review['Type'] == '
COACHING' else "Energy (at the start)"}}</div>
83         <div class="card-body">
84             <h5 class="card-title">{{review.Field4}}/10</h5>
85         </div>
86     </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92
93
94
95 {% endblock content %}
96
97 <!-- Specific JS goes HERE -->
98 {% block javascripts %}
99
100 {% endblock javascripts %}

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Workout {% endblock title %}
4
5 {% 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 <div class="row">
20     <div class="card card-body h-auto justify-content-center" style="width: 18rem;
float: right;">
21         <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
>
22
23     <div class="row justify-content-center">
24
25         <div class="col-lg-2">
26             <div class="card border-light mb-3" style="max-width: 18rem;">
27                 <div class="card-header">Type</div>
28                 <div class="card-body">
29                     <h5 class="card-title">{{ workoutDetails.title }}</h5>
30                 </div>
31             </div>
32         </div>
33         <div class="col-lg-4">
34
35             <div class="card border-light mb-3" style="max-width: 36rem;">
36                 <div class="card-header text-center">Workout</div>
37                 <div class="card-body">
38
39

```

```

40         </div>
41     </div>
42     <div class="col-lg-2">
43
44         <div class="card border-light mb-3" style="max-width: 18rem;">
45             <div class="card-header"></div>
46             <div class="card-body">
47                 <h5 class="card-title">  </h5>
48             </div>
49         </div>
50     </div>
51 </div>
52 <div class="row justify-content-center">
53     <div class="col-lg-2">
54         <div class="card border-light mb-3" style="max-width: 18rem;">
55             <div class="card-header">Heart rate average</div>
56             <div class="card-body">
57                 <h5 class="card-title">{{workoutDetails.heart_rate_avg}}
58                 <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">
59
                    <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 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>
60                 </svg>
61             </h5>
62         </div>
63     </div>
64 </div>
65 <div class="col-lg-2">
66     <div class="card border-light mb-3" style="max-width: 18rem;">
67         <div class="card-header">Calories burned</div>
68         <div class="card-body">
69             <h5 class="card-title">{{ workoutDetails.calories }}
70             <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>
71             </h5>
72         </div>
73     </div>
74 </div>
75 <div class="col-lg-2">
76     <div class="card border-light mb-3" style="max-width: 18rem;">
77         <div class="card-header">Duration</div>
78         <div class="card-body">
79             <h5 class="card-title">{{ workoutDetails.duration }}
80             <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">
81                 <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"/>
82                 <path d="M8 16A8 8 0 1 0 8 0a8 8 0 0 0 16zm7-8A7 7 0 1 1
8a7 7 0 0 1 14 0z"/>
83             </svg>
84             </h5>
85         </div>
86     </div>
87 </div>
88 <div class="col-lg-2">
89     <div class="card border-light mb-3" style="max-width: 18rem;">
90         <div class="card-header">Date</div>
91         <div class="card-body">

```

```

92         <h5 class="card-title">{{ workoutDetails.date.strftime("%d/%m/%Y,
           %H:%M:%S") }}
93         <i class="ni ni-calendar-grid-58 text-orange" style="float:
           right;"></i>
94         </h5>
95     </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </div>
102
103
104
105 {% endblock content %}
106
107 <!-- Specific JS goes HERE -->
108 {% block javascripts %}
109
110 {% endblock javascripts %}

1 {% extends 'layout/base.html' %}
2
3 {% block title %} Workouts {% endblock title %}
4
5 {% 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     <div class="row">
19         <div class="card card-body h-auto" style="width: 18rem; float: right;">
20             <h5 class="card-header">Workouts</h5>
21             <table class="table">
22                 <thead>
23                     <tr>
24                         <th scope="col">Type</th>
25                         <th scope="col">Date</th>
26                         <th scope="col">Duration</th>
27                         <th scope="col">Calories Burned</th>
28                         <th scope="col">Avg Heart Rate</th>
29                         <th scope="col">Details</th>
30                     </tr>
31                 </thead>
32                 {% for workout in workouts %}
33                 <tr>
34                     <td>
35                         {{ workout.title }}
36                     </td>
37                     <td>
38                         {{ workout.date.strftime("%d/%m/%Y, %H:%M:%S") }}
39                     </td>
40                     <td>
41                         {{ workout.duration }}
42                     </td>
43                     <td>
44                         {{ workout.calories }}
45                     </td>
46                     <td>
47                         {{ workout.heart_rate_avg }}
48                     </td>
49                     <td>

```

```

50         <a class="btn btn-sm btn-primary" href="{ url_for('
client_blueprint.workout', Id=workout.id) }}">Details</a>
51     </td>
52 </tr>
53 {% endfor %}
54 </table>
55 </div>
56 </div>
57
58
59
60 {% endblock content %}
61
62 <!-- Specific JS goes HERE -->
63 {% block javascripts %}{% endblock javascripts %}

```