SMART PARKING

PHASE 4:

Developing the mobile app using Python

CODE:

```
import json
import pymongo
from flask import Flask, jsonify
from flask import request
import os
import pandas as pd
import json
import datetime


app = Flask(__name__)


usr = 'user1'
pas = 'pass1'


client = pymongo.MongoClient(
    "mongodb+srv://" + usr + ":" + pas +
    "@cluster0-dex8f.azure.mongodb.net/test?retryWrites=true&w=majority")


db = client['ParkingService']
collection = db['User']



@app.route('/create_profile', methods=['POST'])
```

```
def create_profile():
    req_data = request.get_json()
    db.User.insert_one(req_data).inserted_id
    return ('Profile created', 200)


@app.route('/list_profiles')
def list_profiles():
    documents = collection.find()
    response = []
    for document in documents:
        document['_id'] = str(document['_id'])
        response.append(document)
    return json.dumps(response)


@app.route('/login', methods=['POST'])
def login():
    req_data = request.get_json()
    user, password = req_data['name'], req_data['password']

    documents = collection.find()
    data = {document['name']: document['password'] for
            document in documents}
    if user in data.keys():
        if password == data[user]:
            return (user, 200)
    return ('Login failed', 403)


@app.route('/configure_profile', methods=['POST'])
```

```
def configure_profile():
    req_data = request.get_json()
    update_data = {key: item for key, item in req_data['update_data'].items()}

    db.User.update_one({'name': req_data['name']}, {"$set": update_data},
                       upsert=True)
    return ('Configuration finished', 200)



# %%
def getParkingLotInfo():
    files = os.listdir("mock_data")
    parkingData = {}

    for file in files:
        json_file = os.path.join("mock_data", file)
        with open(json_file, 'r') as f:
            distros_dict = json.load(f)
        parkingData[file.strip('.json')] = distros_dict
        # print(distros_dict)
    return parkingData



def getValidParkingSpots(parking_lot_data, search_parameter, tStart, tEnd):
    # Combine data
    combined_data = {}

    for key, value in parking_lot_data.items():
        for item in value:
            new_key = key + '_' + item['name']
            combined_data[new_key] = {
```

```
            'lot': key.strip('parking'),
            'space': item['name'],
            # 'location': item['location'],
            'free': item['free'],
            **item['type'],
            'bookings': [{
               key: datetime.datetime.strptime(val, '%Y-%m-%d %H:%M:%S')
               for key, val in booking.items()} for booking in item['bookings']],
            "cost": round((tEnd-tStart).seconds/3600*item["cost"], 2),
            'location': {'longitude': item['location'][0],
                     'latitude': item['location'][1]}
            }
# Convert to dataframe
df = pd.DataFrame(combined_data).transpose()
df_free = df[df['free']]
#  Check occupation
free = [True]*len(df_free)
for i, bookings in enumerate(df_free.bookings):
   if bookings != []:
      for booking in bookings:
         delta = min(booking["to"], tEnd)-max(booking["from"], tStart)
         #
         # Check if delta is negative
         #
         if delta >= datetime.timedelta(0):
            print(booking["from"], booking["to"])
            free[i] = False
df_no_booking = df_free[free]
#  Check for all values
parking_searched = df_no_booking
for key, value in search_parameter.items():
```

```
        if value:
            parking_searched = parking_searched[parking_searched[key]]
    df_output = parking_searched.drop(['bookings', 'free'], axis=1)
    return df_output



def getBookings():
    documents = db.Bookings.find()
    for document in documents:
        pass
    bookings = []
    return bookings



@app.route('/search', methods=['POST'])
def searchParkingLots():
    # return "A"
    req_data = request.get_json()
    user_name = req_data['name']
    # documents = db.User.find({'name': user_name}, {})
    documents = db.User.find()
    # print([doc['type'] for doc in documents])
    search_parameter = {document['name']: document['type'] for
                        document in documents}[user_name]
    # keyword = req_data['keyword']
    tStart = datetime.datetime.strptime(req_data['from'], "%d:%m:%Y %H:%M")
    tEnd = datetime.datetime.strptime(req_data['to'], "%d:%m:%Y %H:%M")

    parking_lot_data = getParkingLotInfo()
    valid_parking_spots = getValidParkingSpots(parking_lot_data,
                                       search_parameter,
```

```
                                    tStart,
                                    tEnd)

    dictdata = valid_parking_spots.to_dict("index")
    datalist = [value for value in dictdata.values()]

    return jsonify(datalist)

# %%



@app.route('/book', methods=["POST"])
def book():
    req_data = request.get_json()
    user_name = req_data["name"]
    parking_lot = req_data["parking_slot"]
    tStart = req_data['from']
    tEnd = req_data['to']

    db.Bookings.insert_one({'parking_spot': parking_lot,
                            "name": user_name,
                            "tStart": tStart,
                            "tEnd": tEnd})
    return ("Booking successfull. Parking: {}, User: {}, from {} to {}".format(
        parking_lot, user_name, tStart, tEnd), 200)



@app.route('/listBookings', methods=["POST"])
def listBookings():
    req_data = request.get_json()
    user_name = req_data["name"]
```

```
    documents = db.Bookings.find()

    relevant_bookings = []
    for doc in documents:
        if doc["name"] == user_name:
            relevant_bookings.append({"parking_place": doc["parking_spot"],
                                      "tStart": doc["tStart"],
                                      "tEnd": doc["tEnd"]})

    return jsonify(relevant_bookings)



if __name__ == '__main__':
    app.run(port=9000)
```