**Developing an API for a Distributed Environment**

**Question 1:***Run the API.py code. Take a screenshot of the terminal output. What command did you use to compile and run the code?*

The application can be executed using the following methods:

* api.py
* flask - - app api run - -
* flask - - app api shell

A screenshot of a computer program

Description automatically generated

**Question 2:**[***Run the following command***](http://127.0.0.1:5000/user/Ann)*at the terminal prompt: w3m*[*http://127.0.0.1:5000/user/Ann*](http://127.0.0.1:5000/user/Ann)*. What happens when this command is run, and why?*

The following error is produced

A screen shot of a computer

Description automatically generated

However, when the address is entered into a web-browser the detail for the user =Ann are displayed

A screenshot of a computer

Description automatically generated

**Question 3:**[***Run the following command***](Run%20the%20following%20command)*at the terminal prompt: w3m*[*http://127.0.0.1:5000/user/Adam*](http://127.0.0.1:5000/user/Adam)*. What happens when this command is run, and why?*

Similarly, ‘Adam’ is not found in the database. The only users are James, Jason and Ann. Flask returns an ‘user not found’ response.

A screenshot of a computer

Description automatically generated

**Question 4:** *What capability is achieved by the flask library?*

The flask library allows can be used to develop an API where the database can be amended with CRUD functions via a simple API web interface, following REST protocols

**Code**

from flask import Flask

from flask\_restful import Api, Resource, reqparse

app = Flask(\_\_name\_\_)

api = Api(app)

users = [

{

"name": "James",

"age": 30,

"occupation": "Network Engineer"

},

{

"name": "Ann",

"age": 32,

"occupation": "Doctor"

},

{

"name": "Jason",

"age": 22,

"occupation": "Web Developer"

}

]

class User(Resource):

def get(self, name):

for user in users:

if(name == user["name"]):

return user, 200

return "User not found", 404

def post(self, name):

parser = reqparse.RequestParser()

parser.add\_argument("age")

parser.add\_argument("occupation")

args = parser.parse\_args()

for user in users:

if(name == user["name"]):

return "User with name {} already exists".format(name), 400

user = {

"name": name,

"age": args["age"],

"occupation": args["occupation"]

}

users.append(user)

return user, 201

def put(self, name):

parser = reqparse.RequestParser()

parser.add\_argument("age")

parser.add\_argument("occupation")

args = parser.parse\_args()

for user in users:

if(name == user["name"]):

user["age"] = args["age"]

user["occupation"] = args["occupation"]

return user, 200

user = {

"name": name,

"age": args["age"],

"occupation": args["occupation"]

}

users.append(user)

return user, 201

def delete(self, name):

global users

users = [user for user in users if user["name"] != name]

return "{} is deleted.".format(name), 200

api.add\_resource(User, "/user/<string:name>")

app.run(debug=True)