1. Develop a REST-like web service

The REST-like web service developer can be found in the GitHub repository with the name ‘rest.py’.

To run the script use the command:

```
$ python rest.py
```

or

```
$ python3 rest.py
```

It will run on localhost on the port 5002. To test it you can use the following URL in your browser:

```
http://127.0.0.1:5002/gene_suggest?q=brc&species=homo_sapiens&limit=10
```

Note: Make sure that you have installed the python libraries used, if not you can install it using the following commands:

```
$ pip install mysql-connector
$ pip install flask flask-jsonpify flask-sqlalchemy flask-restful
```

or

```
$ pip3 install mysql-connector
$ pip3 install flask flask-jsonpify flask-sqlalchemy flask-restful
```

Note 2: I've chosen Flask over other frameworks like Django, Pyramid, and web2py because it is very lightweight and therefore easy to understand.

2. Deployment

Describe how would you deploy your web service. How would you ensure your solution can scale to meet increased demand?

To deploy my service, if I had the necessary hardware, I would use a self-hosted server suitable for production like nginx with a WSGI middleware such as uWSGI. This is a popular option to host Python services because it is specially designed to work in a high performance context using a microservices architecture pattern.

I would use this architecture pattern because it reduces the complexity of the services and also enhances the scalability of the service improving the Y-axis scaling of the Scale Cube. Moreover I would run multiple instances of the service behind a load balancer for throughput and availability.

In case I wouldn't had the necessary hardware, I would use a cloud-based service like AWS or Heroku because they provide high performance services with auto-scale functions.

3. Testing

What strategies would you employ to test your application? How would you automate testing?

First I would run a serie of unit tests, taking the individual methods available in the API and testing each one of them in isolation. After that I would write some test cases with different inputs in order to verify if the API is returning the expected outputs. Finally I would execute performance test to verify compliance with the scalability requirements.

In order to run the unit tests I would use a Python test framework such as PyUnit, this is the standard test automation framework for unit testing. For the other two types of tests I would write some Python scripts with the cases and the performance tests needed.