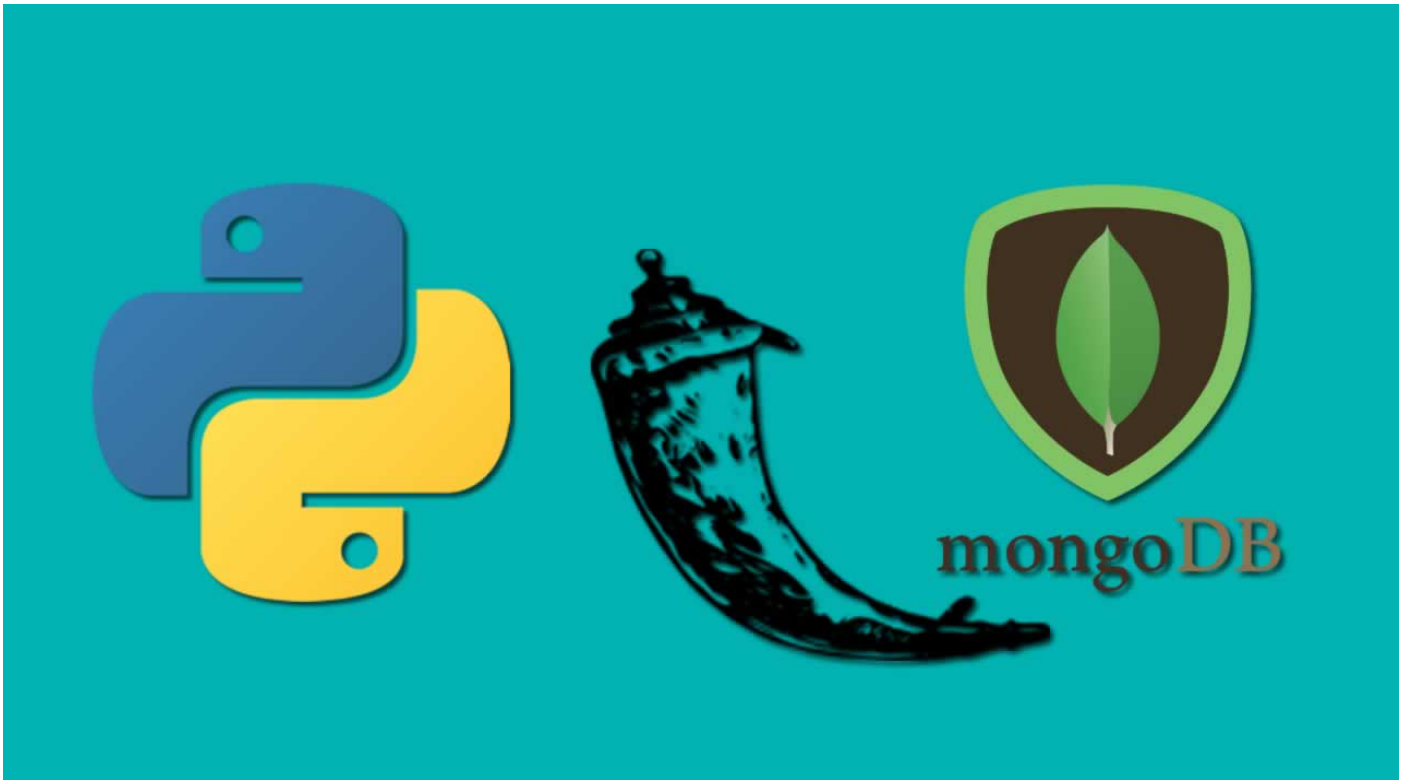
Hemanta ADHIKARI: hemanta.adhikari86@gmail.com

NoSQL Operation with MongoDb and Flask

CRM (Customer Relationship Management) System



Written By
Hemanta ADHIKARI

INTRODUCTION

This project takes track on order of customer. From the order to delivery it will shows the status of order or consignment or the project. It deals for the manipulation or the operations on the order of command. The model which can be added in future will be accounting system, leads management, and payroll management with the same system.

TECHNICAL BLOCKS

The application is developed based on the following software's,

SOFTWARE	DESCRIPTION
MongoDB	MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with the schema. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).
Flask	Flask is python based microframework as it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, from validation, upload handling various open authentication technologies and several common framework related tools.
PyMongo	Python needs a MongoDB driver to access the MongoDB database. Many other languages has their different types as driver to access database. So PyMongo is the data base driver for python to MongoDb.

HTML	Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
Bootstrap CSS	Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

INSTALLATION GUIDE

Step 1: Install **Flask** using this link https://pypi.org/project/Flask/ **or pip install -U Flask** based on your system

```
(webmongo) Hemantas-MacBook-Pro:todo hemantaadhikari$ pip install flask
Requirement already satisfied: flask in /Users/hemantaadhikari/webmongo/lib/pyth
on3.8/site-packages (1.1.2)
Requirement already satisfied: click>=5.1 in /Users/hemantaadhikari/webmongo/lib
/python3.8/site-packages (from flask) (7.1.2)
Requirement already satisfied: Werkzeug>=0.15 in /Users/hemantaadhikari/webmongo
```

Step 2: Install the MongoDb. Here I am showing the mongoDb installation for Mac OS

Tap the MongoDB Homebrew Tap

Use the below command in your terminal

brew tap mongodb/brew

Now use the below command to install MongoDB Community Edition using the brew package manager

Brew install mongodb-community@4.2

As the mondodb creates it binary files mongod.conf in usr/local/etc directory

The log directory path will be /usr/local/log/mongdb

The data directory path will be /usr/local/var/ mongodb

So this file should be access by system or to create on it to run mongoDb. Otherwise the problem may arises and couldn't work. So make sure of the directory and applied permission on it.

MongoDb needs to start the services the services before accessing it so make sure to start mongoDb

For this I show you two option.

1 To start service during the computer start

With onetime command.

brew services start mongodb-community@4.2

2. Or make manual run the services from terminal using

mongod

```
hemantaadhikari — mongod — 204×55
Hemantas-MacBook-Pro:~ hemantaadhikari$ mongod
2020-07-13T15:44:59.010+0200 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-07-13T15:44:59.017+0200 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] MongoDB starting : pid=1004 port=27017 dbpath=/data/db 64-bit host=Hemantas-MacBook-Pro.local
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] db version v4.2.8
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] git version: 43d25964249164d76d5e04dd6cf38f6111e21f5f
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] allocator: system
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] modules: none
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] build environment:
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten]     distarch: x86_64
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten]     target_arch: x86_64
2020-07-13T15:44:59.022+0200 I  CONTROL  [initandlisten] options: {}
2020-07-13T15:44:59.023+0200 I  STORAGE  [initandlisten] Detected data files in /data/db created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2020-07-13T15:44:59.023+0200 I  STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7680M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=f
alse,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=
[recovery_progress,checkpoint_progress],
```

To stop the services use below command

Brew services stop mongodb-community@4.2

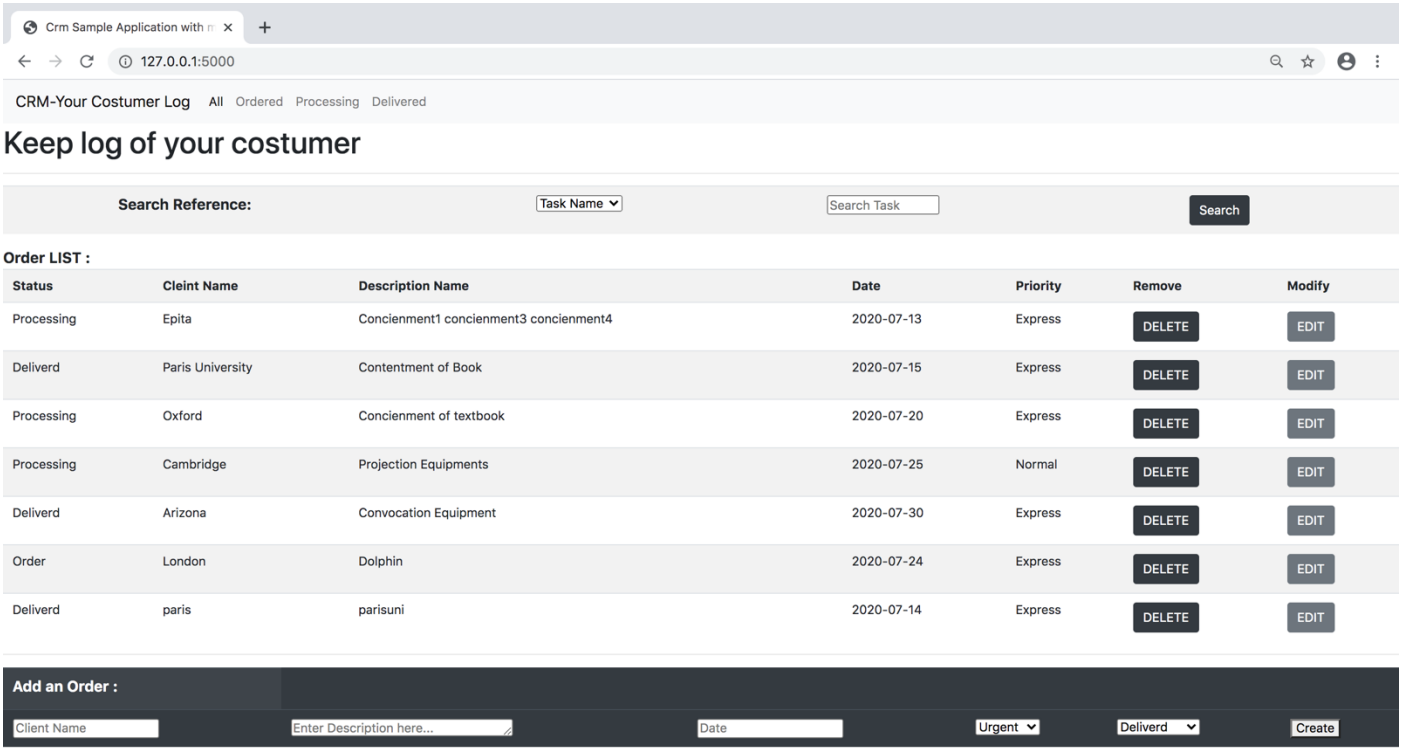
Download the project here https://github.com/hemdai/crm_mongo and chose your favourite IDE to access it to make changes or point your terminal for the same folder and run with python

Hemanta ADHIKARI: hemanta.adhikari86@gmail.com

Python app.py

The system will start and can be access in localhost with the below details

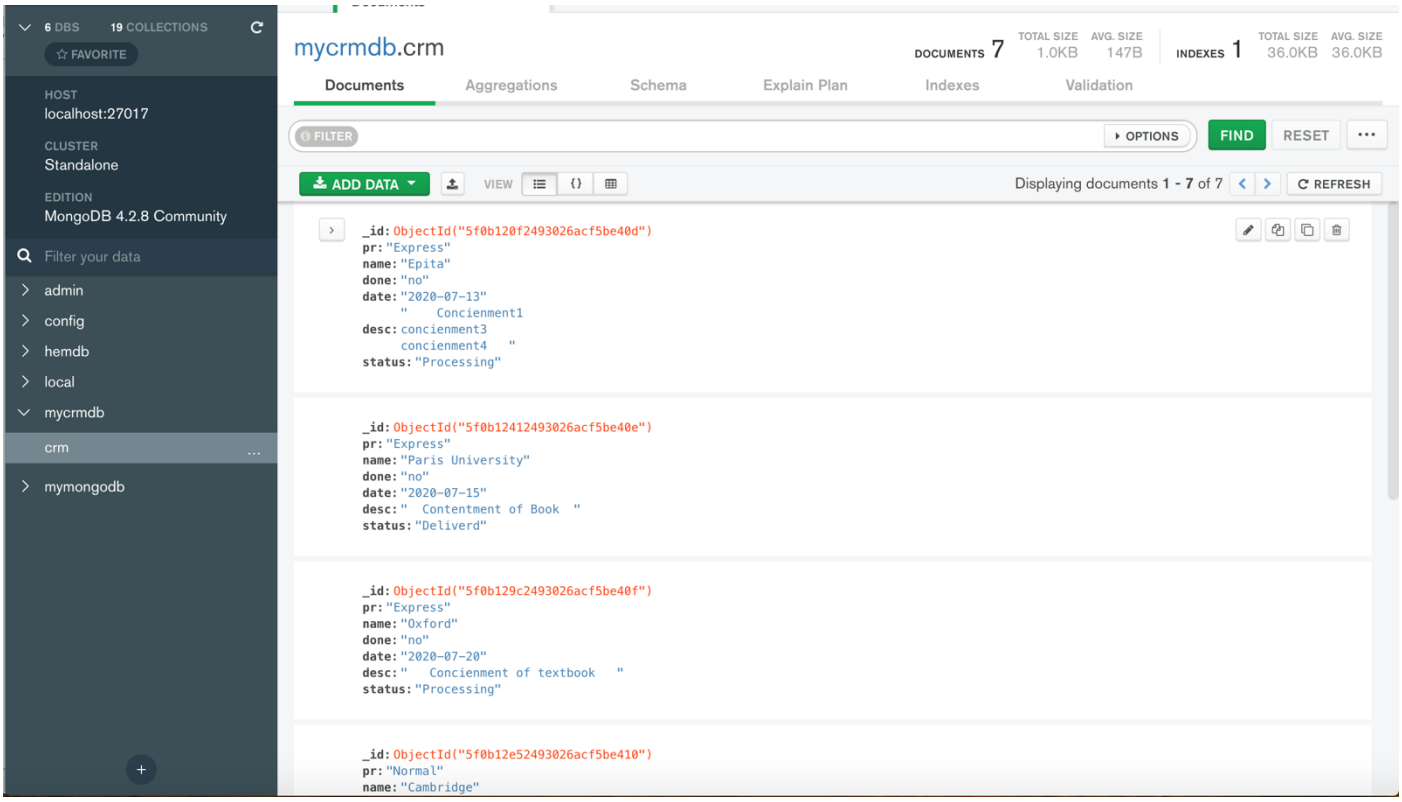
http:127.0.0.1:5000



Hemanta ADHIKARI: hemanta.adhikari86@gmail.com

GUI | USER MANUAL & CODE SNIPPETS

For GUI visual of the database you can choose tools like ROBO 3T or MongoDb Compass. Here I am using MongoDb Compass for the this visualization purpose



Hemanta ADHIKARI: **hemanta.adhikari86@gmail.com**

CRUD OPERATIONS

Step 1: CREATE

A new Order of Book is made is from Sorbonne University to the table.

CODE SNIPPET

In MongoDb

```
_id: ObjectId("5f0c76feb644255695ea63a3")
status: "Order"
name: "Sorbonne University"
desc: " Book for Data Science "
date: "2020-07-13"
pr: "Urgent"
done: "no"
```

For the HTML Handler

```
<form action="/action" method="POST">
<table class="table table-striped table-dark">
  <tr>
    <td><b><big><label>Add an Order : </label></big></b></td>
  </tr>
  <tr>
  <td><input type="text" name="name" placeholder="Client Name" /></td>
  <td><textarea name="desc" rows="1" cols="30" placeholder="Enter Description here..." required></textarea></td>
  <td><input type="text" name="date" placeholder="Date" /></td>

  <td><select name= "pr" method="GET" action="/">
  {% for value1 in priority %}
  <option value= "{{ value1 }}" SELECTED>{{ value1 }}</option>"
  {% endfor %}
  </select>
  </td>



  <td><select name= "status" method="GET" action="/">
  {% for value1 in status %}
  <option value= "{{ value1 }}" SELECTED>{{ value1 }}</option>"
  {% endfor %}
  </select>
  </td>
  <td><button type="submit"> Create </button></td>
  </tr>
</form>
```

Now this section is handled by Flask Session

```
@app.route("/action", methods=['POST'])
def action ():
    #Adding a Task
    name=request.values.get("name")
    desc=request.values.get("desc")
    date=request.values.get("date")
    pr=request.values.get("pr")
    status=request.values.get("status")
    crmd.insert({ "status":status,"name":name, "desc":desc, "date":date, "pr":pr, "done":"no"})
    return redirect("/list")
```

Hemanta ADHIKARI: hemanta.adhikari86@gmail.com

Step 2: READ

All the **order** in the table are **displayed** by default however we can make view conditional example here,

CRM-Your Costumer Log All Ordered Processing Delivered

Keep log of your costumer

Search Reference: Priority Express Search

Order LIST :

So the respective key associated with vale can be shown like ways

Its like reading document with Filter.

Result of the Search : List

Status	Task Name	Task Description	Date	Project	Delete	Modify
Processing	Epita	Concienment1 concienment3 concienment4	2020-07-13	Express	DELETE	EDIT
Deliverd	Paris University	Contentment of Book	2020-07-15	Express	DELETE	EDIT
Processing	Oxford	Concienment of textbook	2020-07-20	Express	DELETE	EDIT
Deliverd	Arizona	Convocation Equipment	2020-07-30	Express	DELETE	EDIT
Order	London	Dolphin	2020-07-24	Express	DELETE	EDIT
Deliverd	paris	parisuni	2020-07-14	Express	DELETE	EDIT

CODE SNIPPET

```
@app.route("/search", methods=['GET'])
def search():
    #Searching a Task with various references

    key=request.args.get("key")
    refer=request.args.get("refer")
    crmd_l = crmd.find({refer:key})
    a3 = "Details on "+ key


    return render_template('searchlist.html',a3=a3,crmd=crmd_l,t=title,h=heading)
```

Hemanta ADHIKARI: hemanta.adhikari86@gmail.com

Step 3: UPDATE

Now lets update the status of Sorbonne University consignment as delivered

Before:

Keep log of your costumer

Result of the Search : List

Status	Task Name	Task Description	Date	Project	Delete	Modify
Order	Sorbonne University	Book for Data Science	2020-07-13	Urgent	DELETE	EDIT

Assesment:

Keep log of your costumer

Update tasks with a reference

Unique Object ID : 5f0c76feb644255695ea63a3

Deliverd

Task Name	:	Sorbonne University
Description	:	Book for Data Science
Date	:	2020-07-13

Urgent

Update Task

Return to TaskList

Result

Crm Sample Application with

127.0.0.1:5000/search?refer=name&key=Sorbonne+University

Keep log of your costumer

Result of the Search : List

Status	Task Name	Task Description	Date	Project	Delete	Modify
Deliverd	Sorbonne University	Book for Data Science	2020-07-13	Urgent	DELETE	EDIT

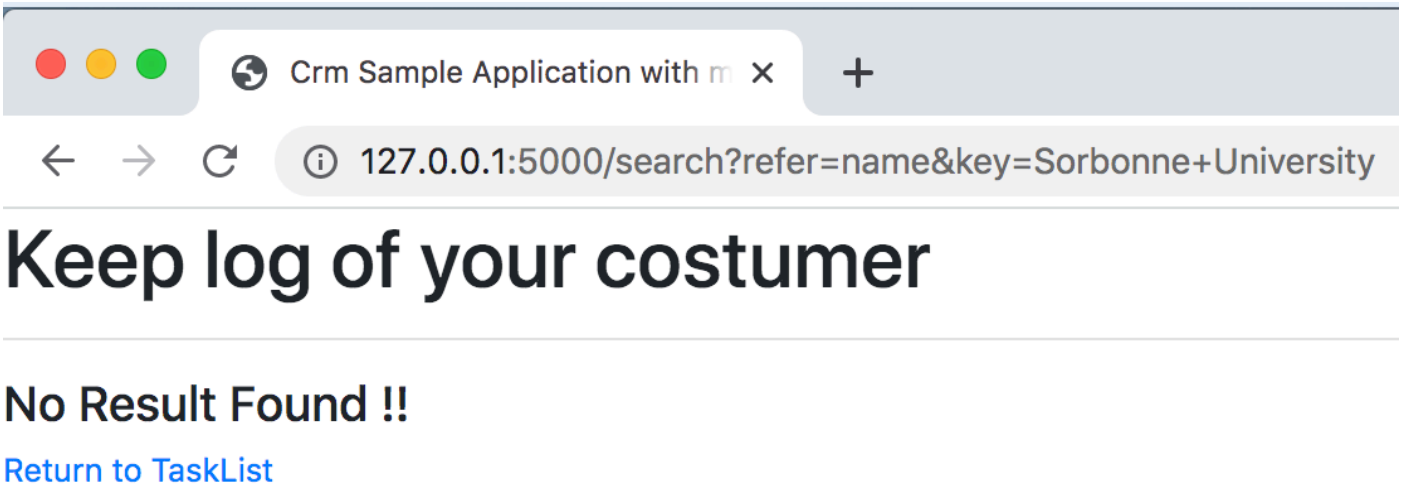
Return to TaskList

CODE SNIPPET

```
@app.route("/update")
def update ():
    id=request.values.get("_id")
    task=crmd.find({"_id":ObjectId(id)})
    return render_template('update.html',priority=priority,status=status,tasks=task,h=heading,t=title)
```

Step 4: DELETE

Now lets delete the same order



CODE SNIPPET

```
@app.route("/remove")
def remove ():
    #Deleting a Task with various references
    key=request.values.get("_id")
    crmd.remove({"_id":ObjectId(key)})
    return redirect("/list")
```

Conclusion

So like this ways we can do the CRUD operation of NoSQL in our project.

Thank you very much for your time and please feel free to contact me in case of any error or assessment in this project.

2020-07-13

Vitry

France