Design & Approach Document

Environment:

Python = 3.7.4 (version)

MongoDB = 5.0 (version)

OS = Window 10

Packages:

Flask==2.0.2

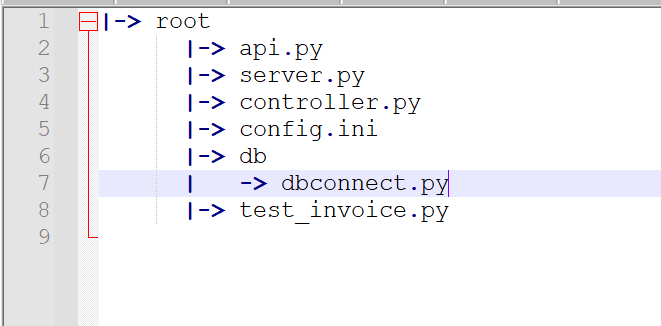
fuzzywuzzy==0.18.0

pymongo==4.0.1

Installation:

MongoDB installation on Windows is done through executable (.msi)

Folder Structure:



APIs:

Data Ingestion:

1. **Create Invoice**

Method: POST

Endpoint: {{host}}/api/invoice

1. **Update Contact**

Method: PUT

Endpoint:{{host}}/api/contact

Data Suggestion:

1. **Suggest Invoice**

Method: GET

Endpoint: {{host}}/api/suggest/invoice

**Mongo Collections:**

DB Name: invoice

Collection Name: invoice

Index: (Index is maintained as 30:1 ratio for organization vs contact)

db.invoice.createIndex(

{

contact: "text",

\_id: "text"

},

{

weights: {

contact: 1,

\_id: 30

},

name: "InvoiceIndex"

}

)

**Implementation Approach:**

**Data Ingestion**

*Invoice Create:* Data is getting as it is from the request document and stored in the collection. There is no transformation done as per the scope of the task.

*Contact Update:* The document is searched with the “contact id” from the input request and search in the collection and updated all the attributes of the contact object. (Updating the first matching document)

Note: We can use “update\_many” to update all the matching document with a small change in the code.

**Data Suggestion**

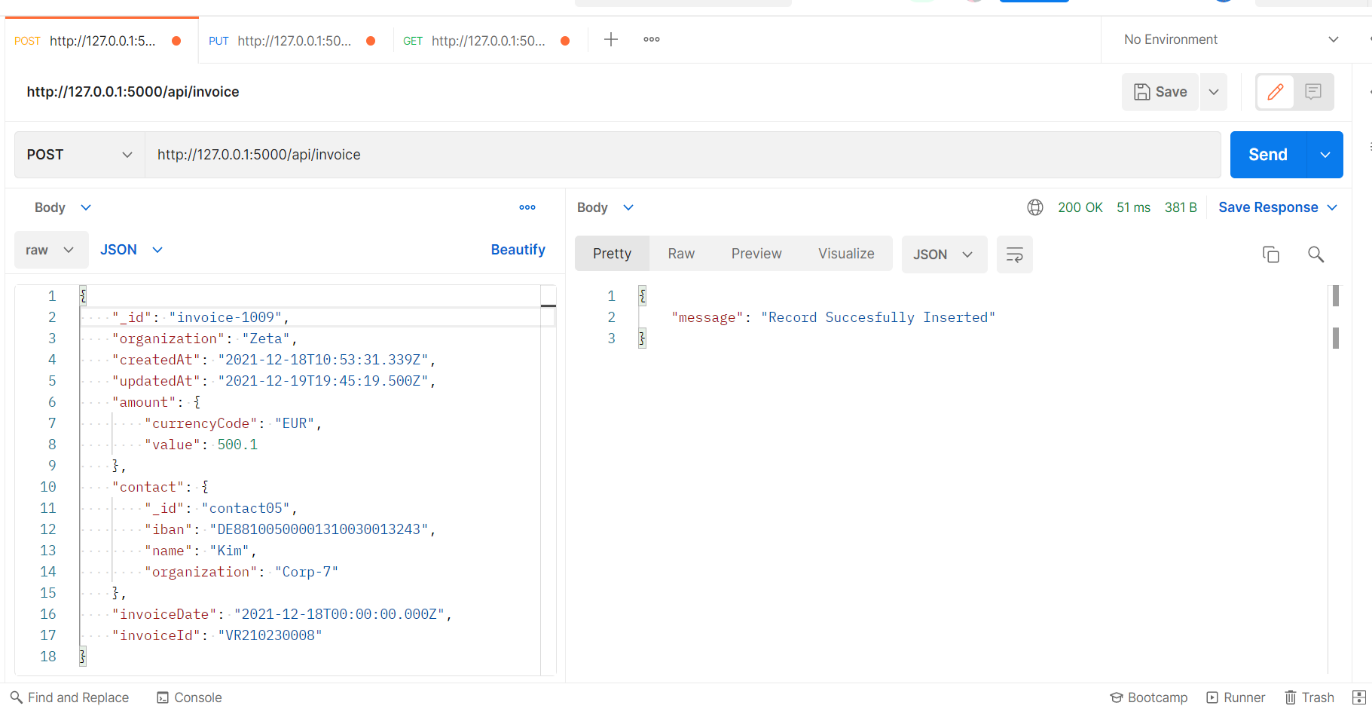
*Invoice Suggestion based on previous invoices:* The organization id and contact name are given as input. The logic written is first, taking “contact\_name” from all the documents matching with given “organization\_id”. The **set of** these contact names are taken to search against the given contact\_name.

The fuzzy matching algorithm (LVH distance) is used which can search both partial as well as full text. In this case, set of contact names are provided to the function along with given contact\_name which return the matching percentage as well as the matching string.

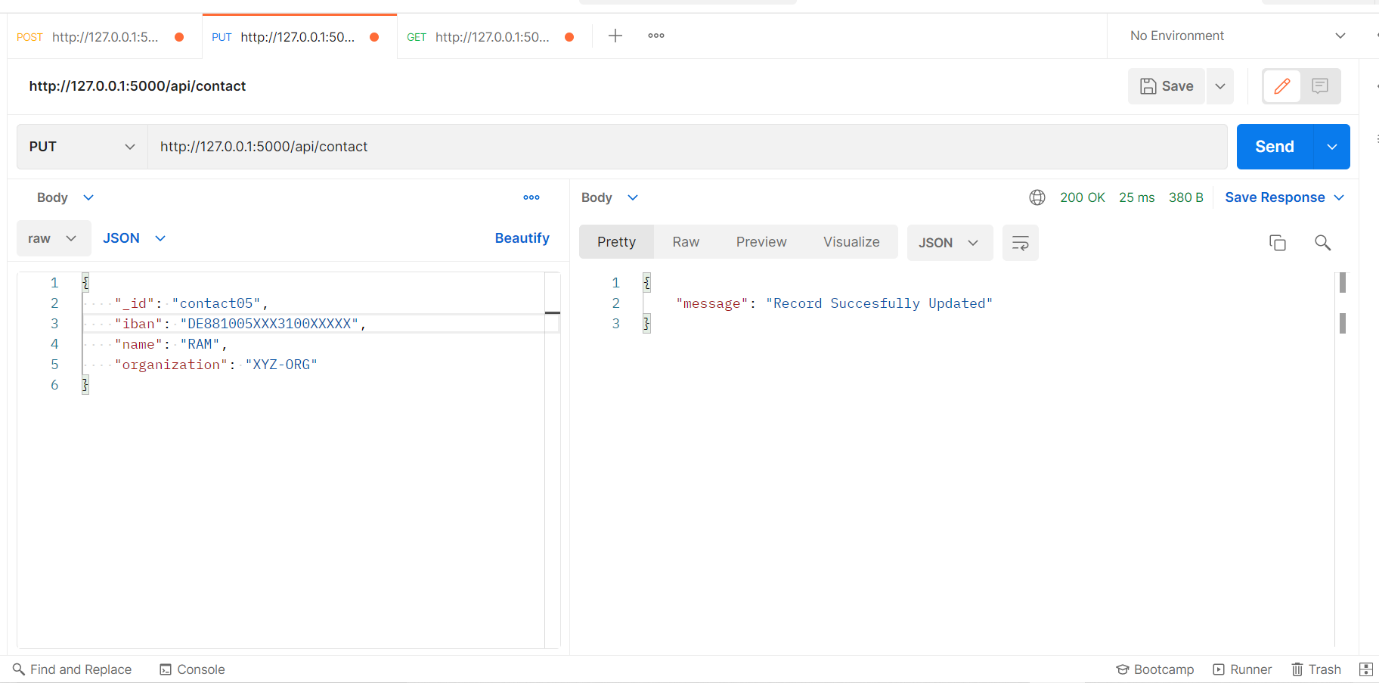
Note: We can use other string-matching algorithms like sliding window for optimization which can search in O(kn) time or can adopt interactive learning machine learning model which eventually can outperform this invoice suggestion approach.

API Screen Shots

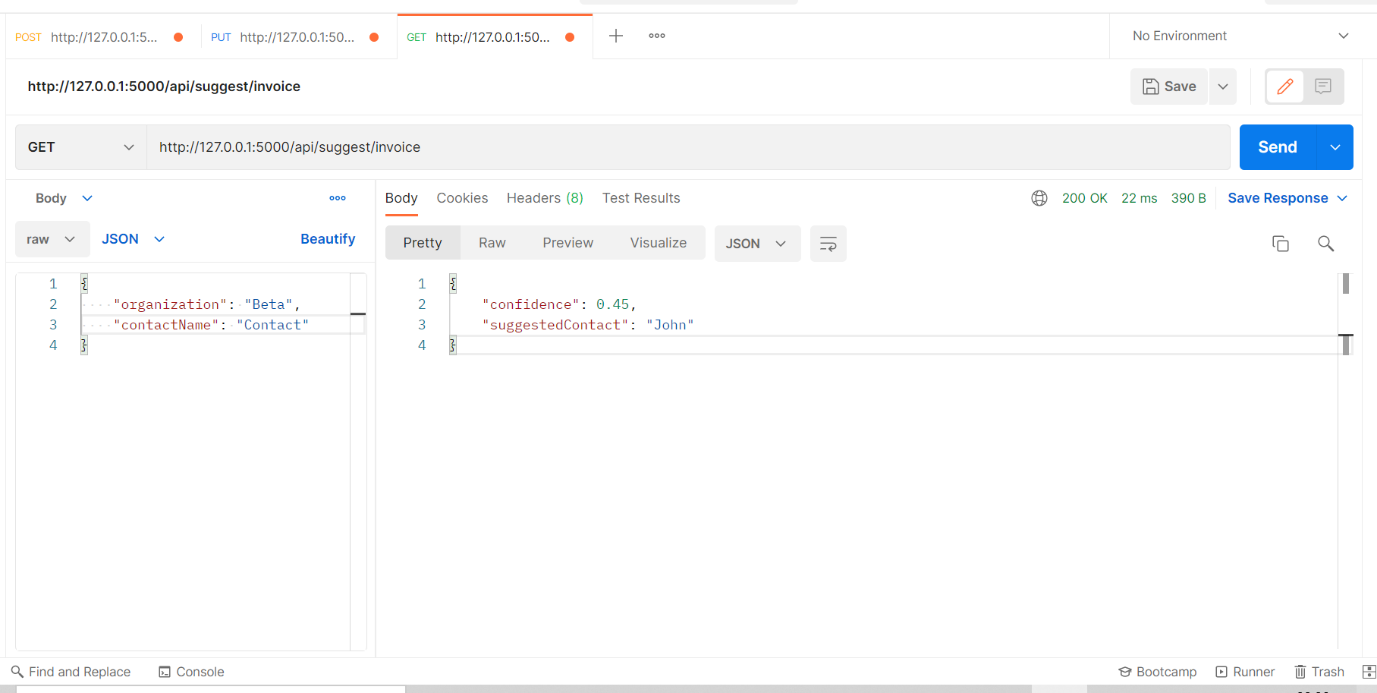
1. Create Invoice



1. Update Contact



1. Suggest Invoice Contact



1. Test Cases

