**GETTING STARTED**

This repository contains a basic implementation of CRUD operations for managing a resource with a "name" attribute. The primary identifier for each resource is the "id" field.

To use this CRUD application, follow the instructions below:

**Prerequisites**

Before getting started, make sure you have the following prerequisites:

Node.js and npm installed on your machine.

**INSTALLATION**

* Clone this repository to your local machine.

**Git clone <repository-url>**

* Navigate to the project directory.

**cd <project-directory>**

* Install the dependencies.

**npm init -y**

**npm install**

* Start the application.

**npm start**

**DOCUMENTATION - CRUD OPERATION**

The project aims to develop the backend of a real-world application in Node.js, Express.js and MongoDB. With the CRUD OPERATION, you can do the following:

1. Add a new user using just "name" as an attribute

2. List down all names present in the db

3. Find a particular name using "ID" in the db

4. Update a particular name using "ID" in the db

5. Delete a name

I am required to develop WEB APIs using Node.js which will send and receive data in the form of JSON to/from in the CRUD APPLICATION.

The application expose endpints required to do the CRUD operation with the database maintained in MongoDB.

The application will use several nodoe packageds includeing Express and Mongoose

CRUD Schema

The CRUD schema will consist just this field:

name: STRING

HTTP Status

I have leaned the different HTTP response status codes and how to implement them when creatign the API.

Output and return the corresponding HTTP status codes based on the functionality or message

**The response codes I make use in this project are as follows:**

**HTTP Status OK: 201**: This will be used if the request made by the user results in a successful response and the user gets what he wants

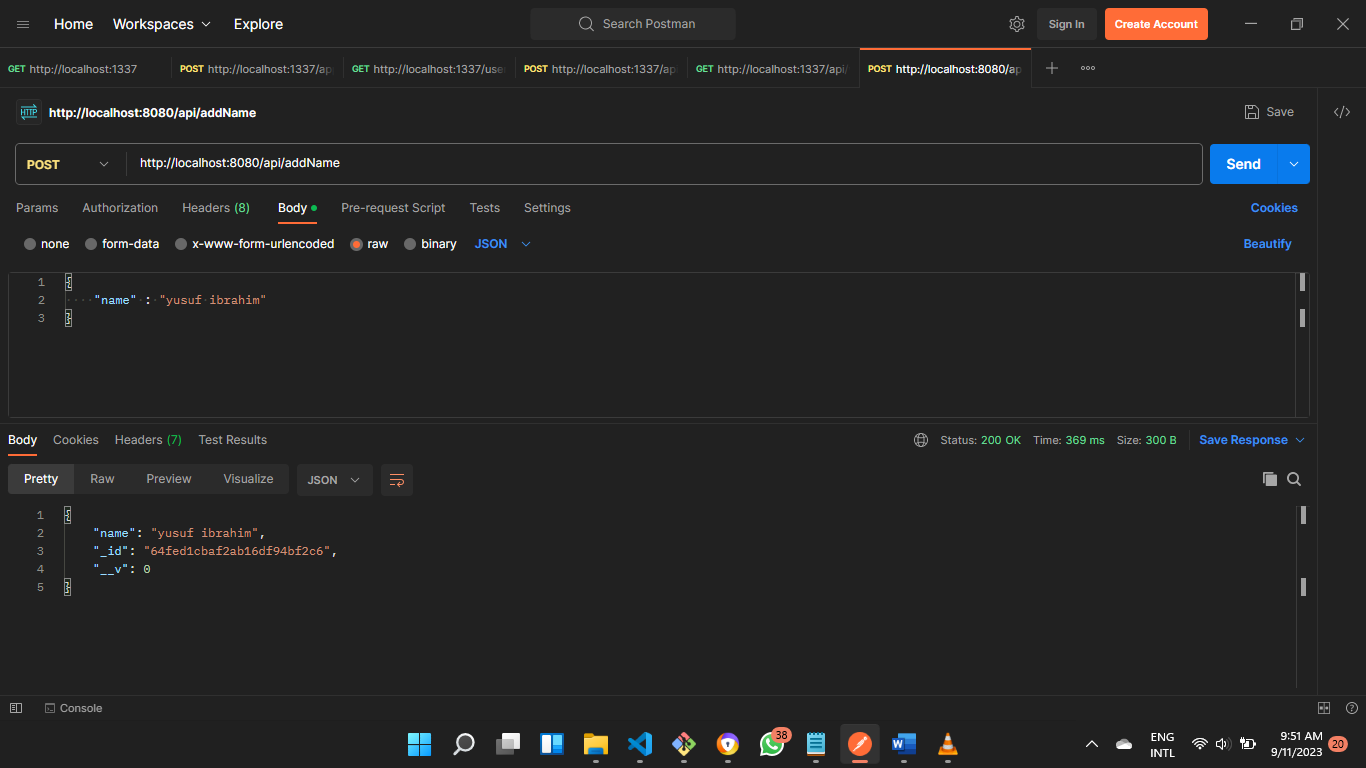
**HTTP Status 500: BAD REQUEST:** Thsi will be used if the server does not understand the request due to invalid syntax or an internal error

**RESTAPI ENDPOINTS = ADD**

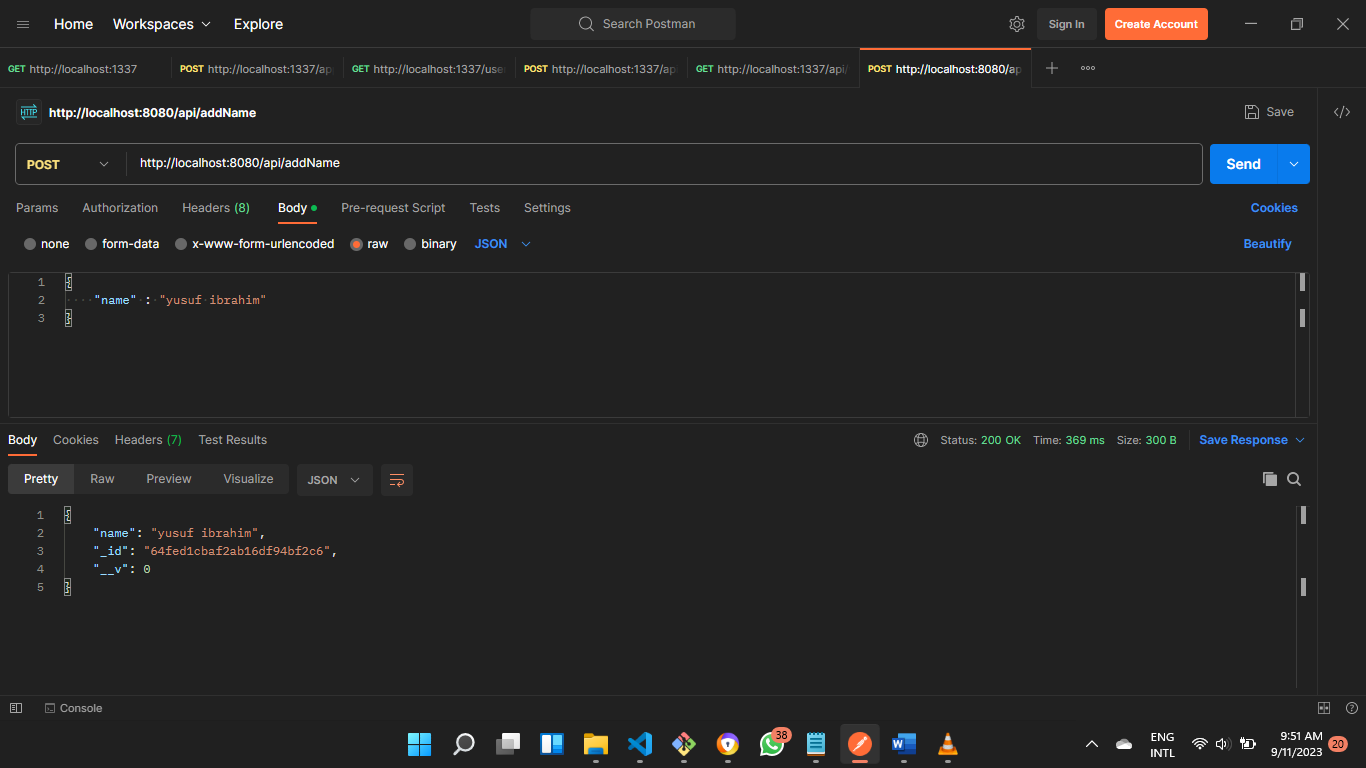
**POST /api/addName**

This API will collect new name and add it to the database.

**RESPONSE BODY**



**EXPECTED OUTPUT**

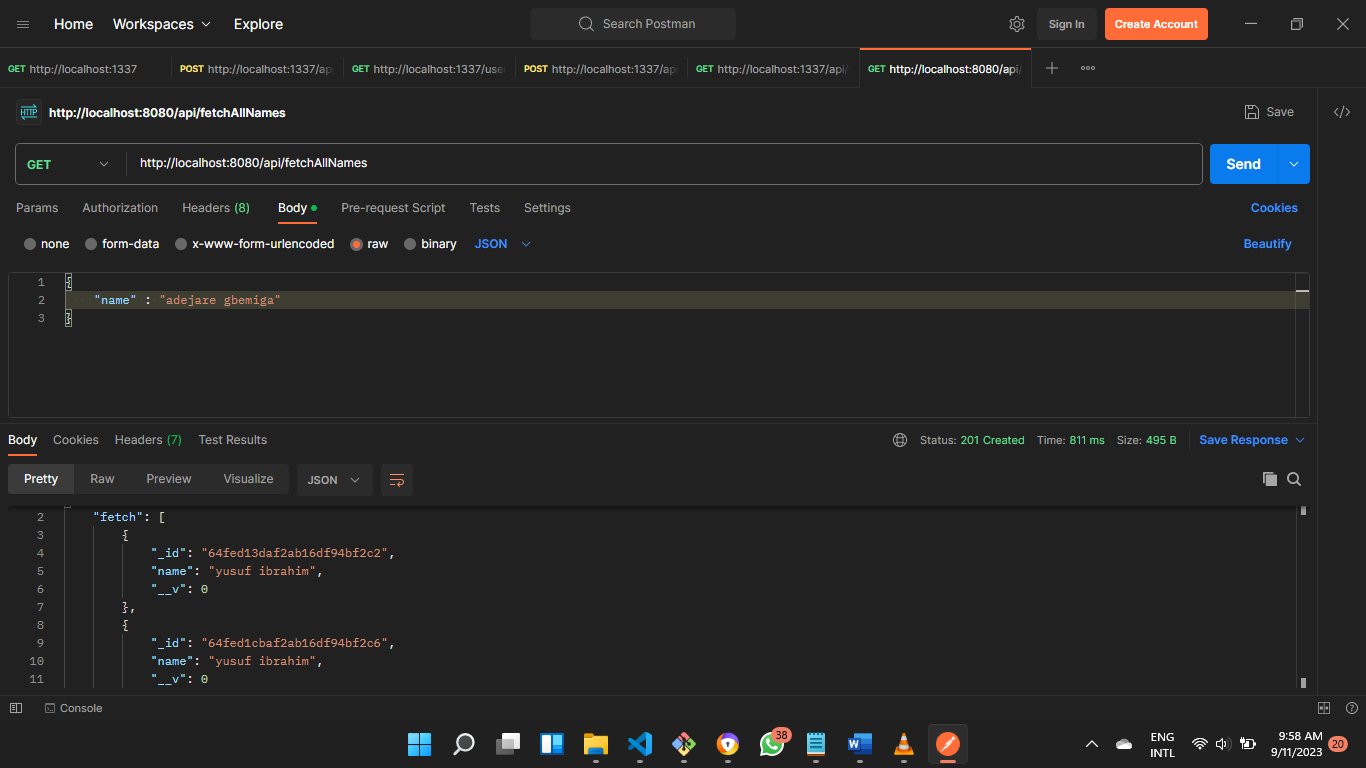


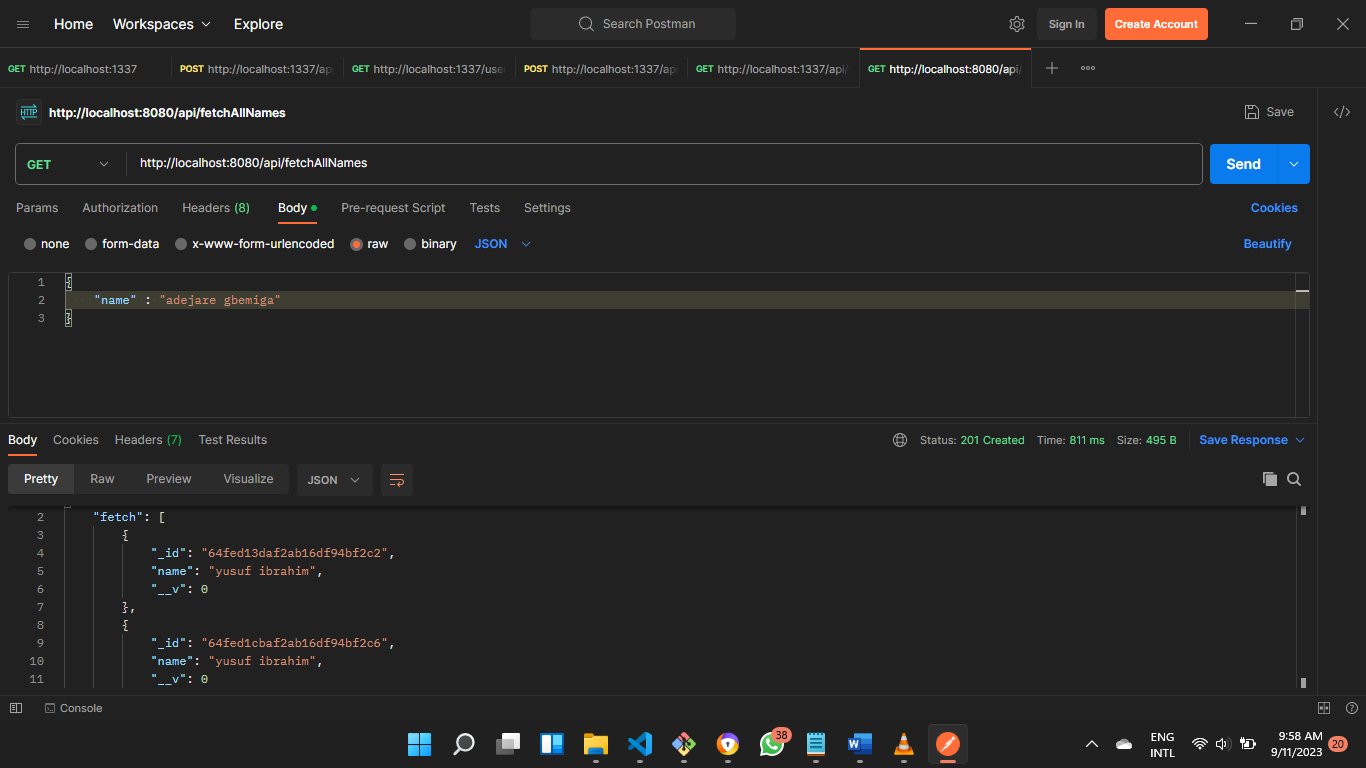
On successful creation of the name, it should return status code 201

**RESTAPI ENDPOINTS = GET NAMES**

**GET /api/fetchAllNames**

This API will return all the names present in the database.

**RESPONSE BODY**

**EXPECTED OUTPUT**

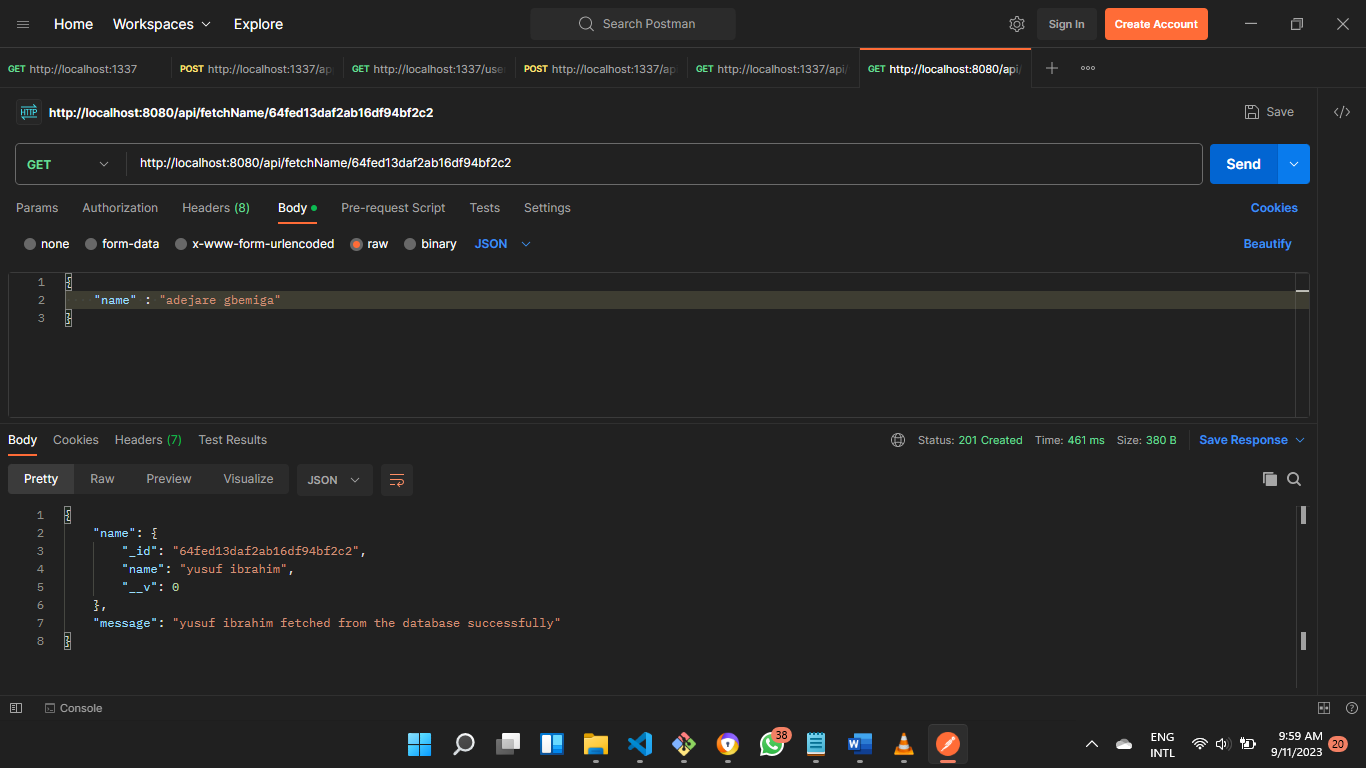
On successful creation of the name, it should return status code 201.

If there are no names, it should return [].

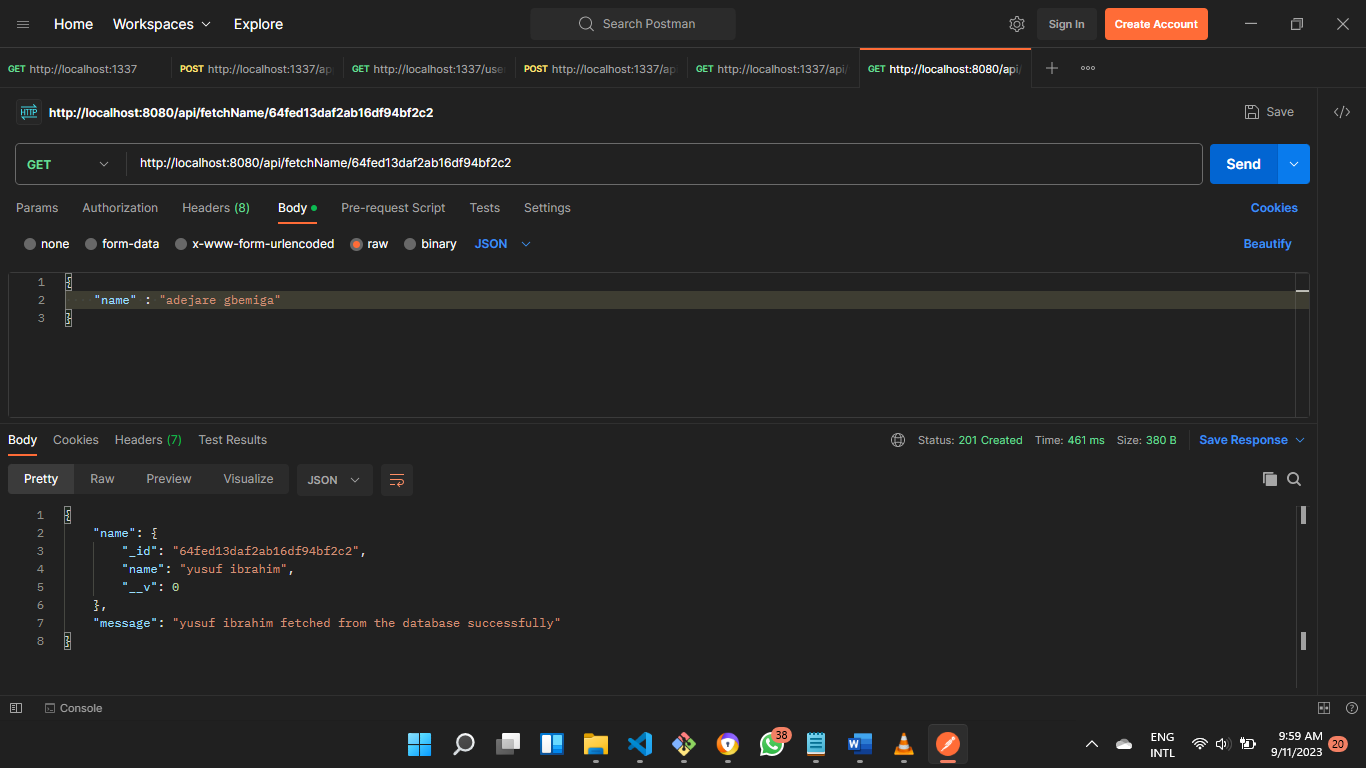
**RESTAPI ENDPOINTS = GET NAME**

**GET /api/fetchName/:id**

This API will return the name specify by the ID in the database.

**RESPONSE BODY**

**EXPECTED OUTPUT**



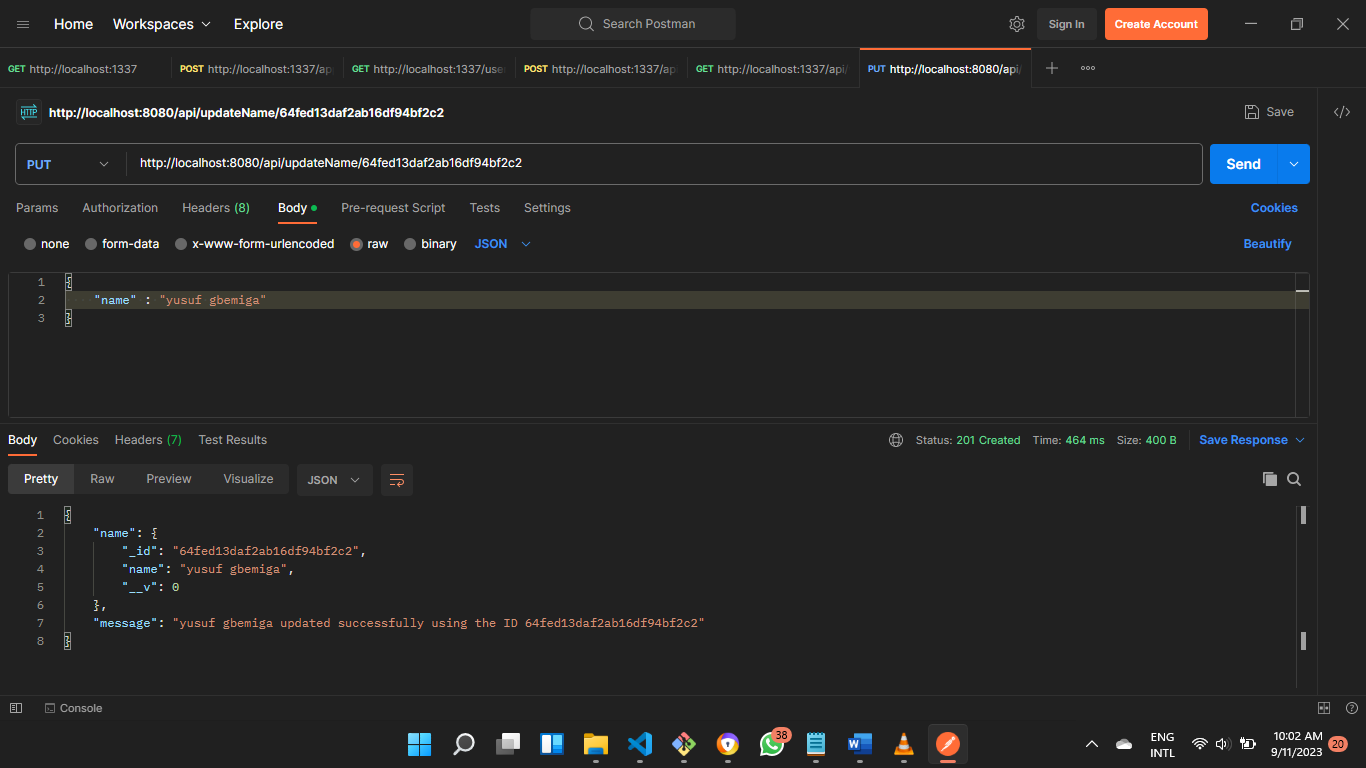
On successful creation of the name, it should return status code 201.

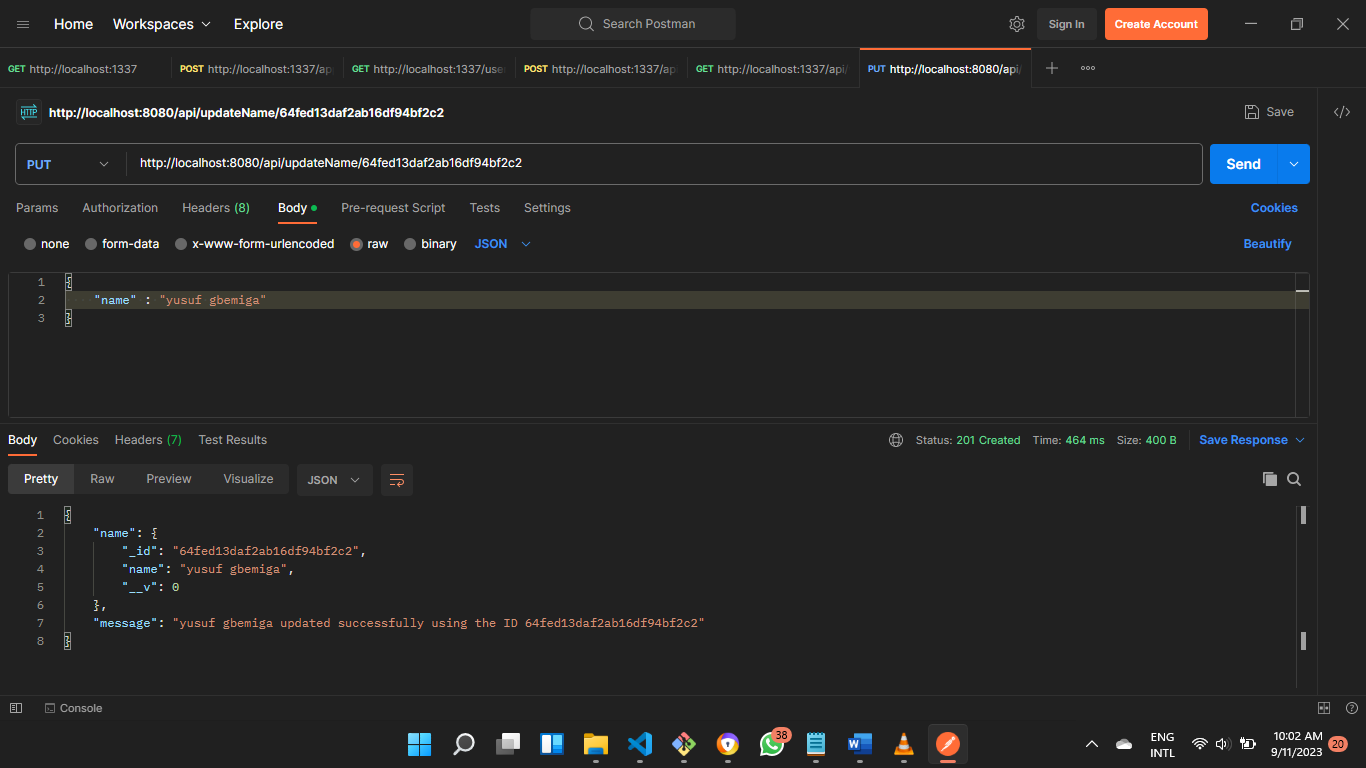
If there is no such name, it should return "NO SUCH NAME"

**RESTAPI ENDPOINTS = UPDATE NAME**

**PUT /api/updateName/:id**

This API will update the name specify by the ID in the database.

**RESPONSE BODY**

**EXPECTED OUTPUT**

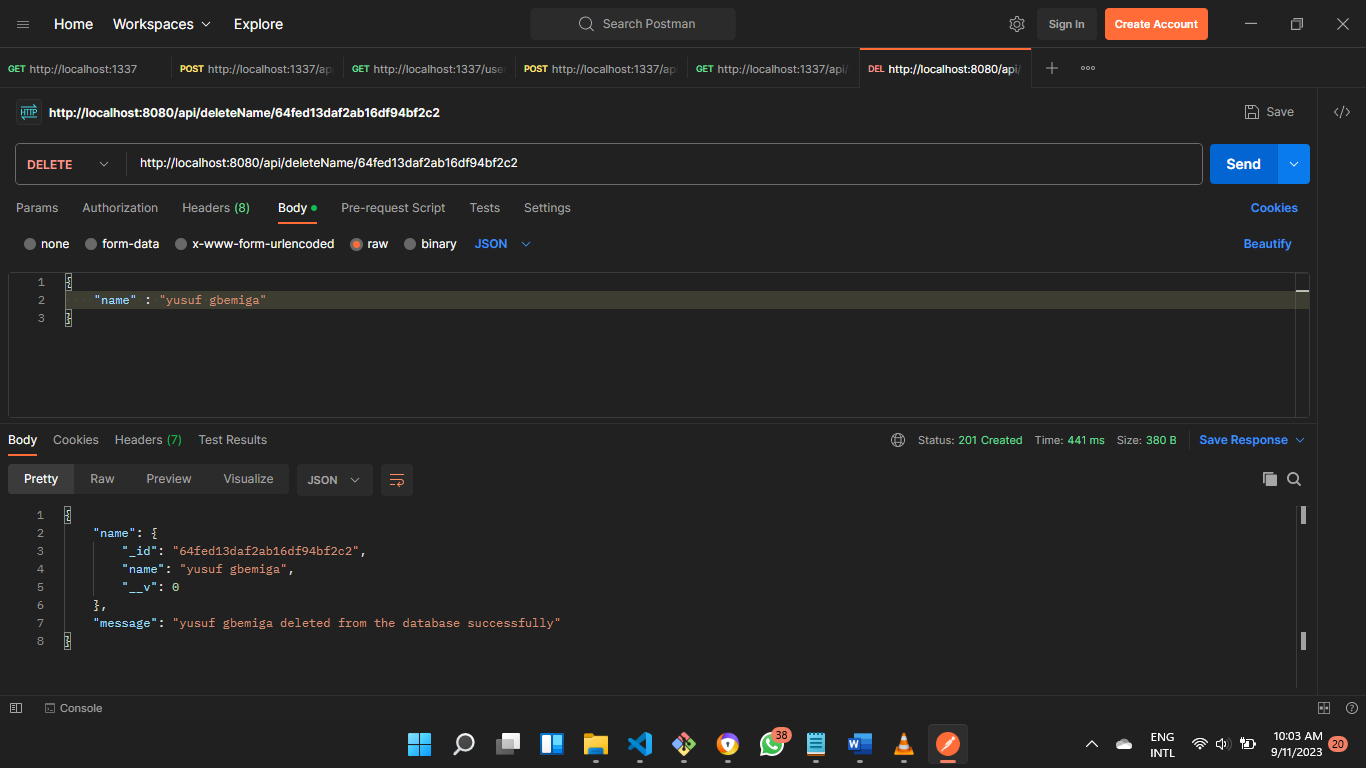
On successful creation of the name, it should return status code 201.

If there is no such name, it should return "NO SUCH NAME"

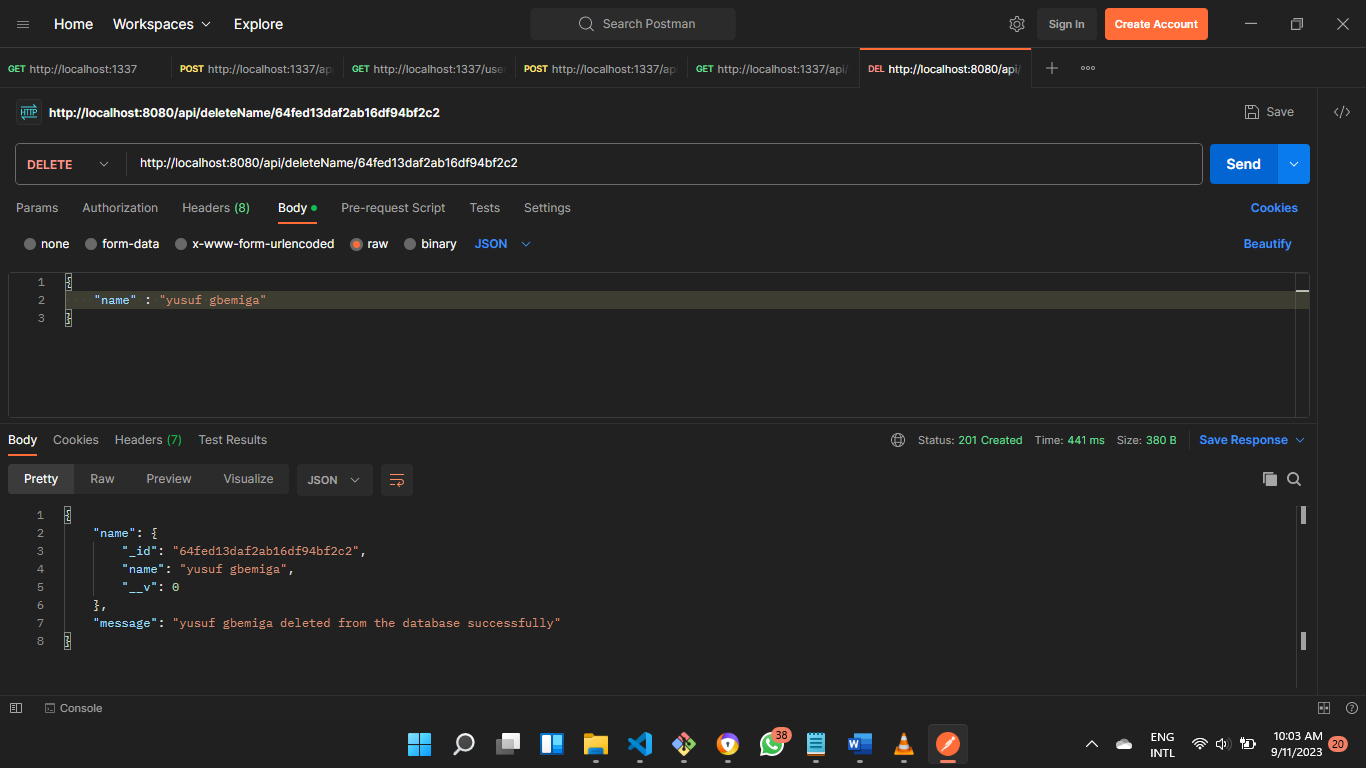
**RESTAPI ENDPOINTS = DELETE NAME**

**DELETE /api/deleteName/:id**

This API will delete the name specify by the ID in the database.

**RESPONSE BODY**

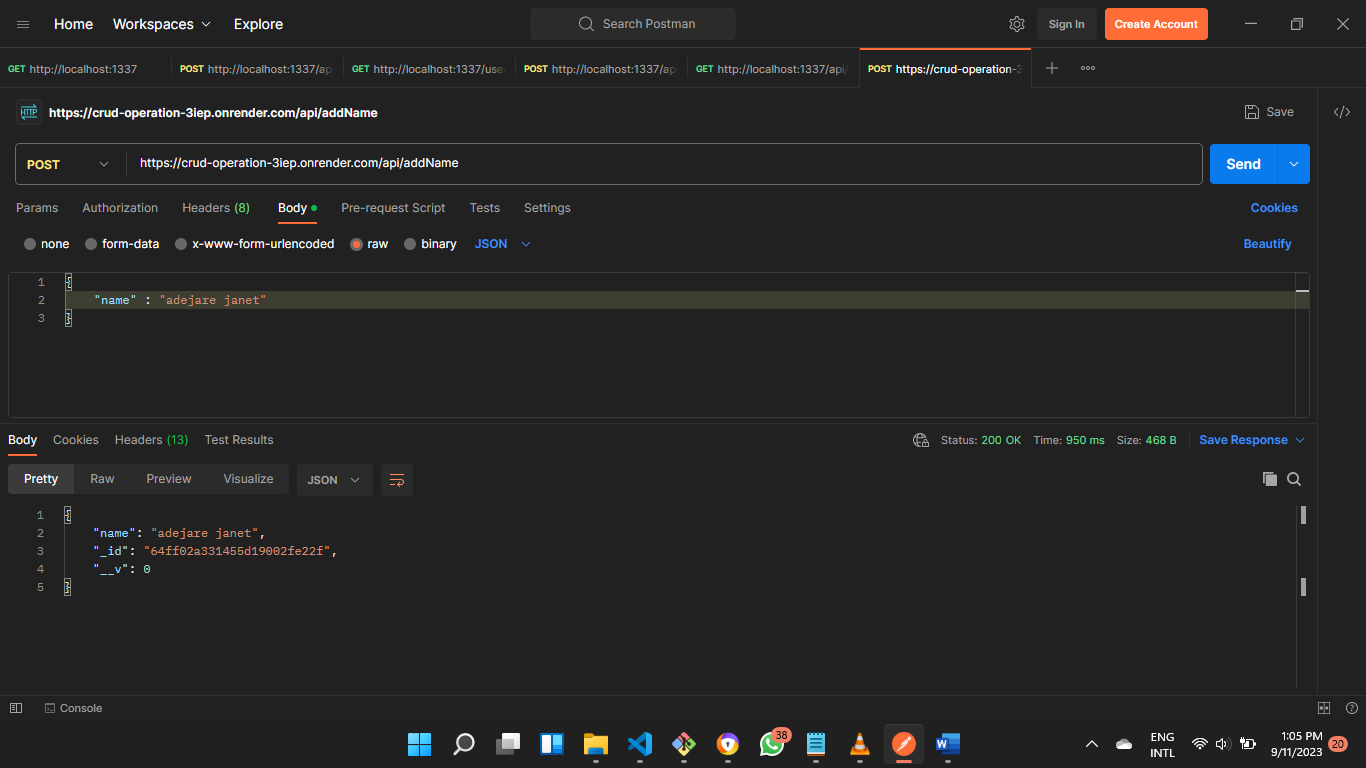
**EXPECTED OUTPUT**



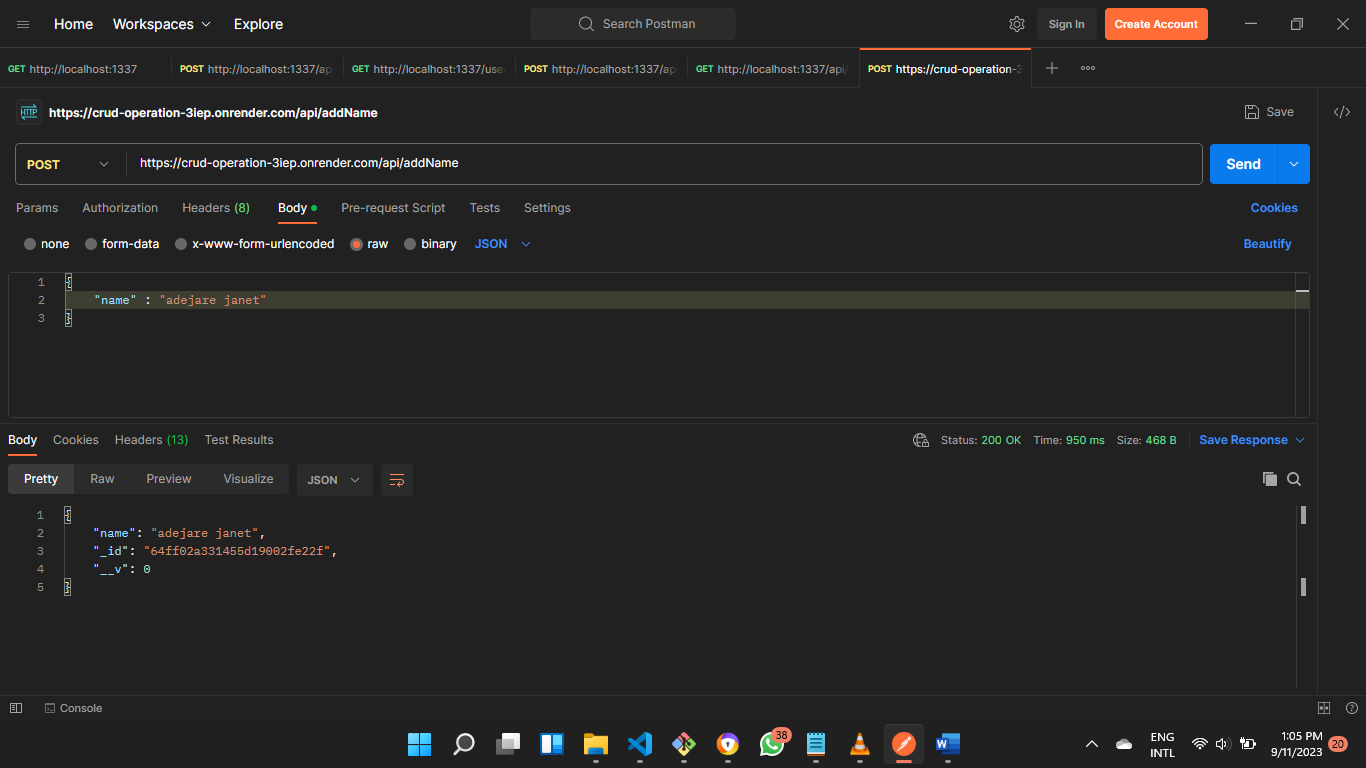
On successful creation of the name, it should return status code 201.

If there is no such name, it should return "NO SUCH NAME"

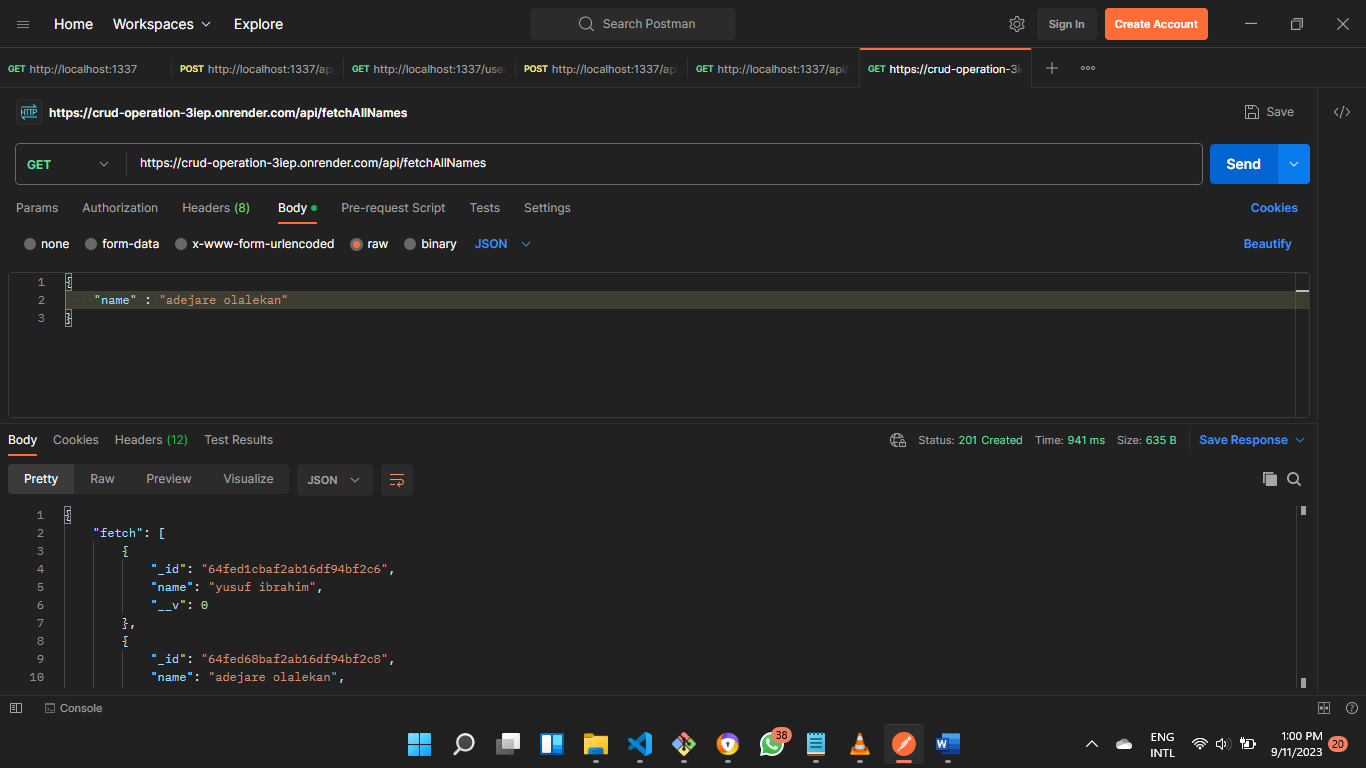
ADD

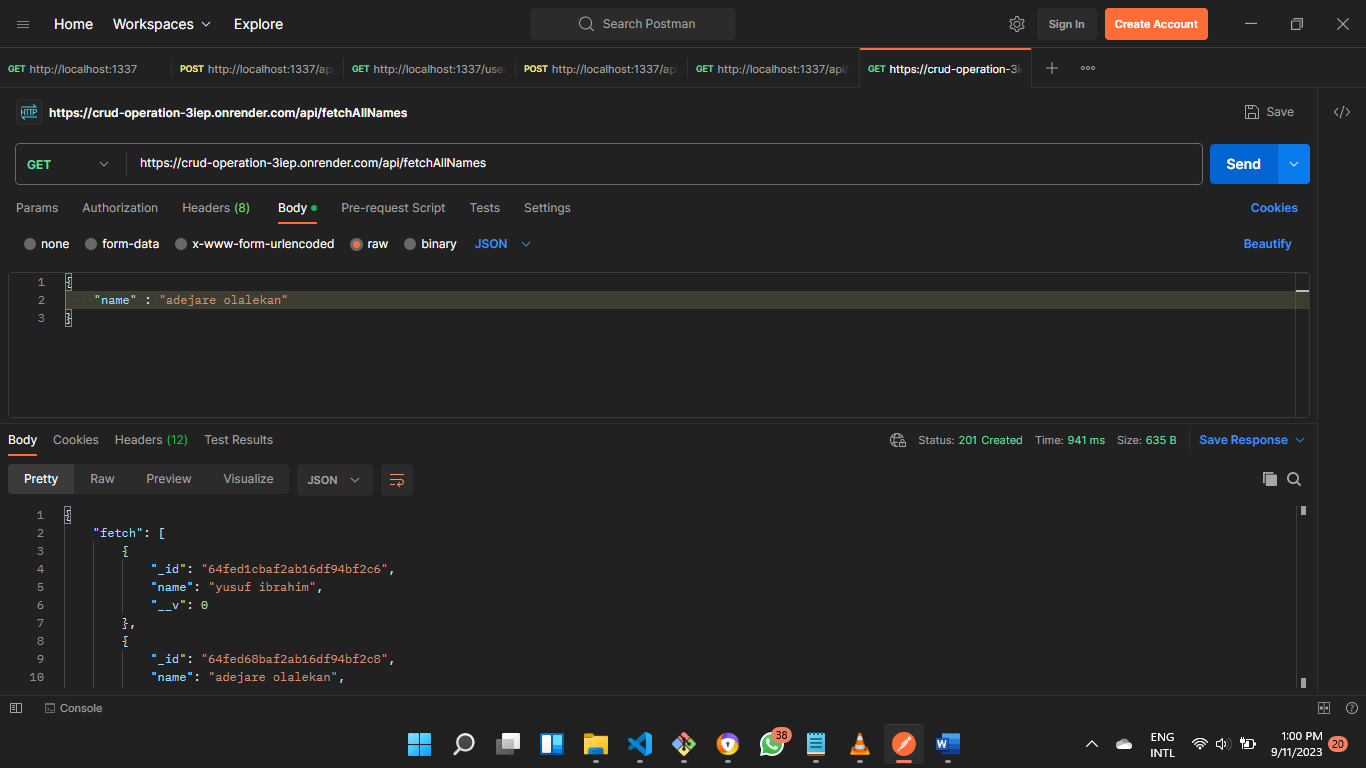
BODY

RESPOSE

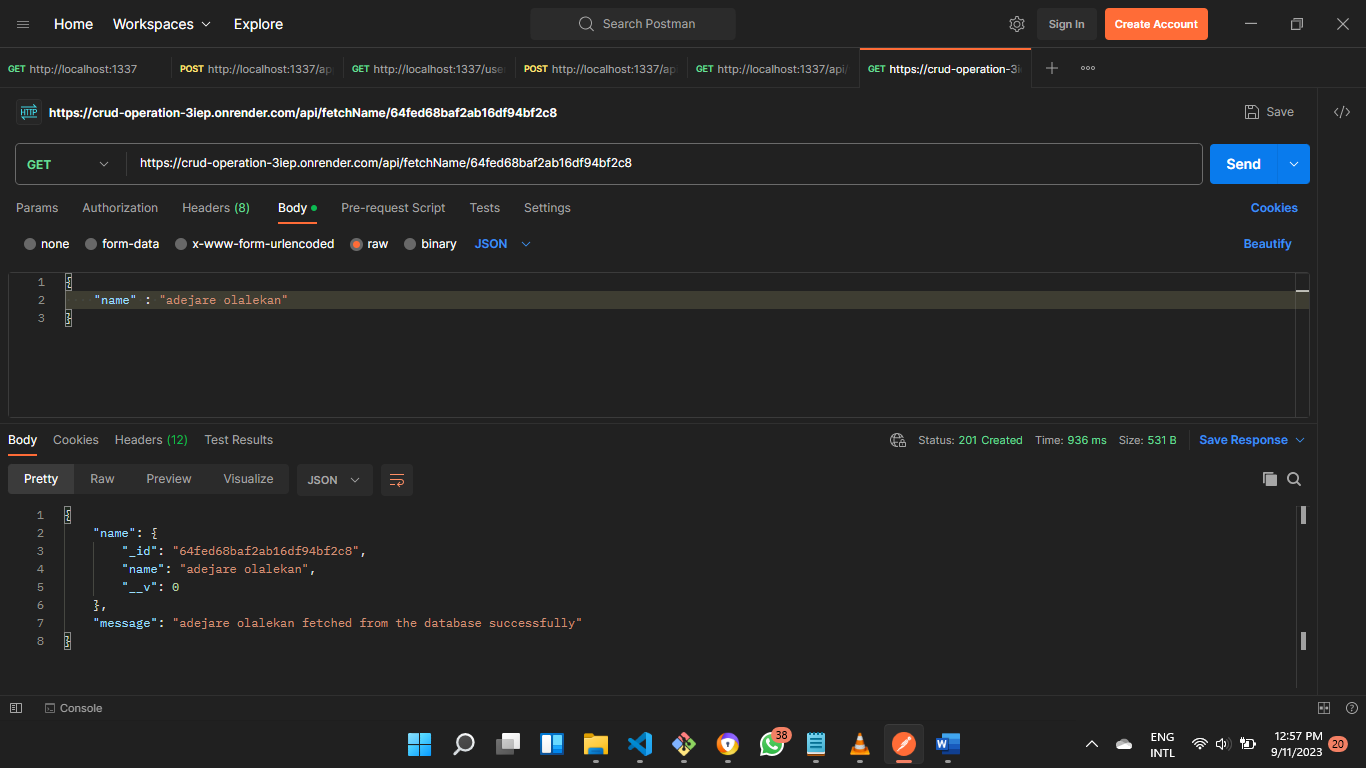


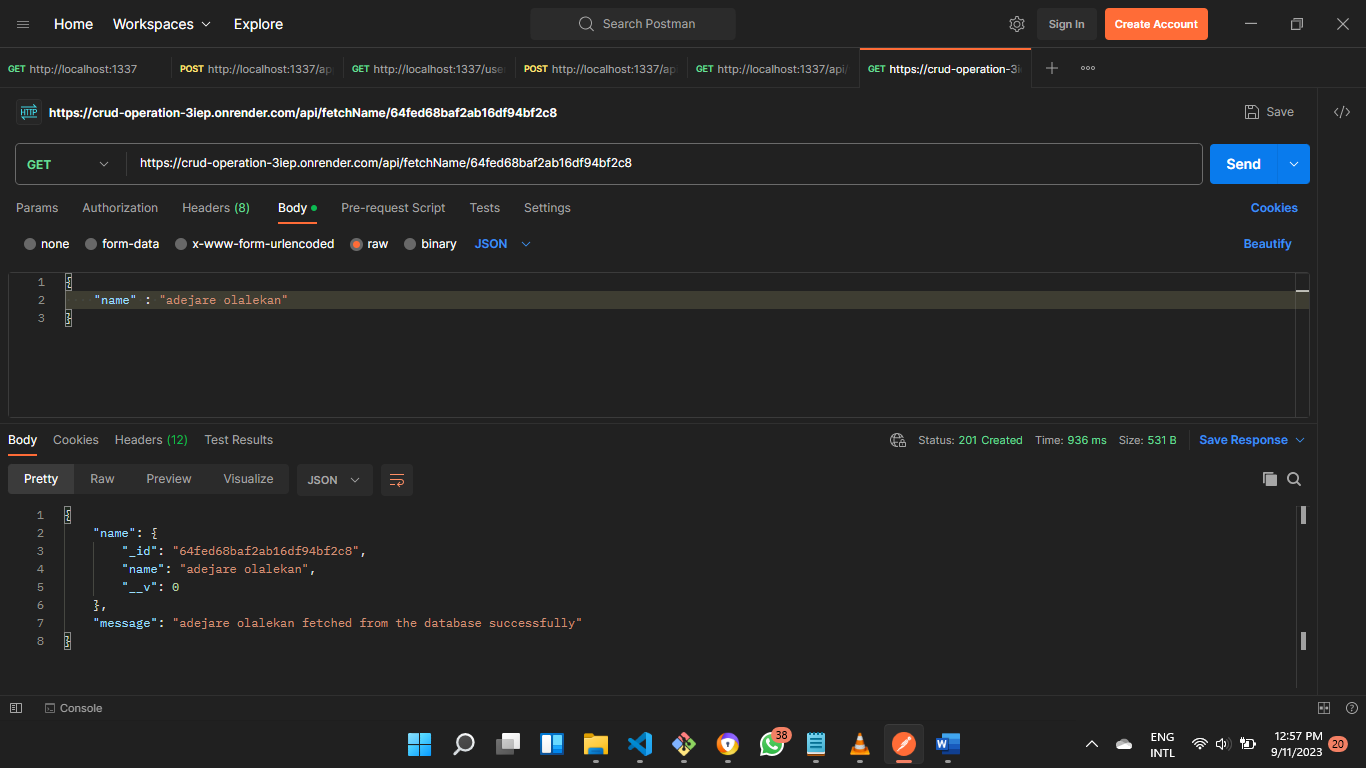
FETCH ALL NAMES

BODY

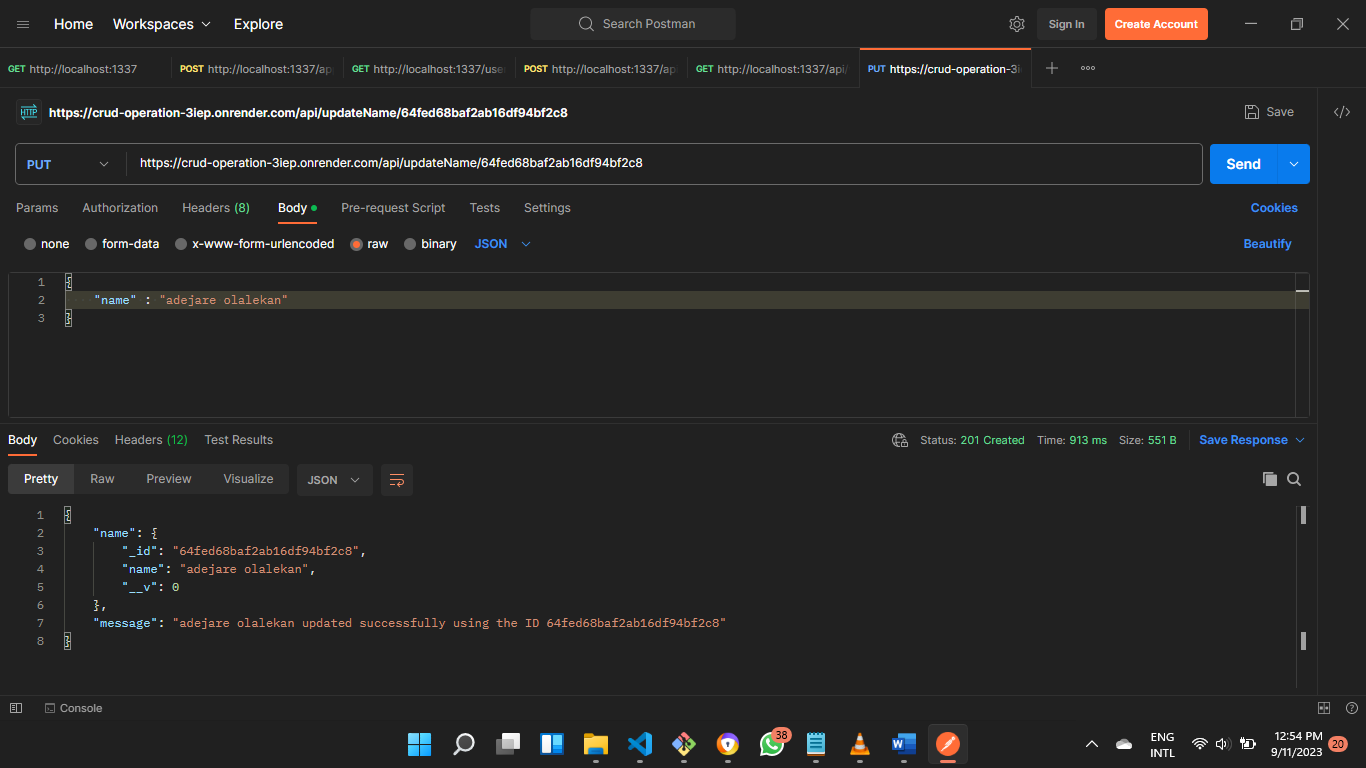
RESPONSE

FETCH NAME

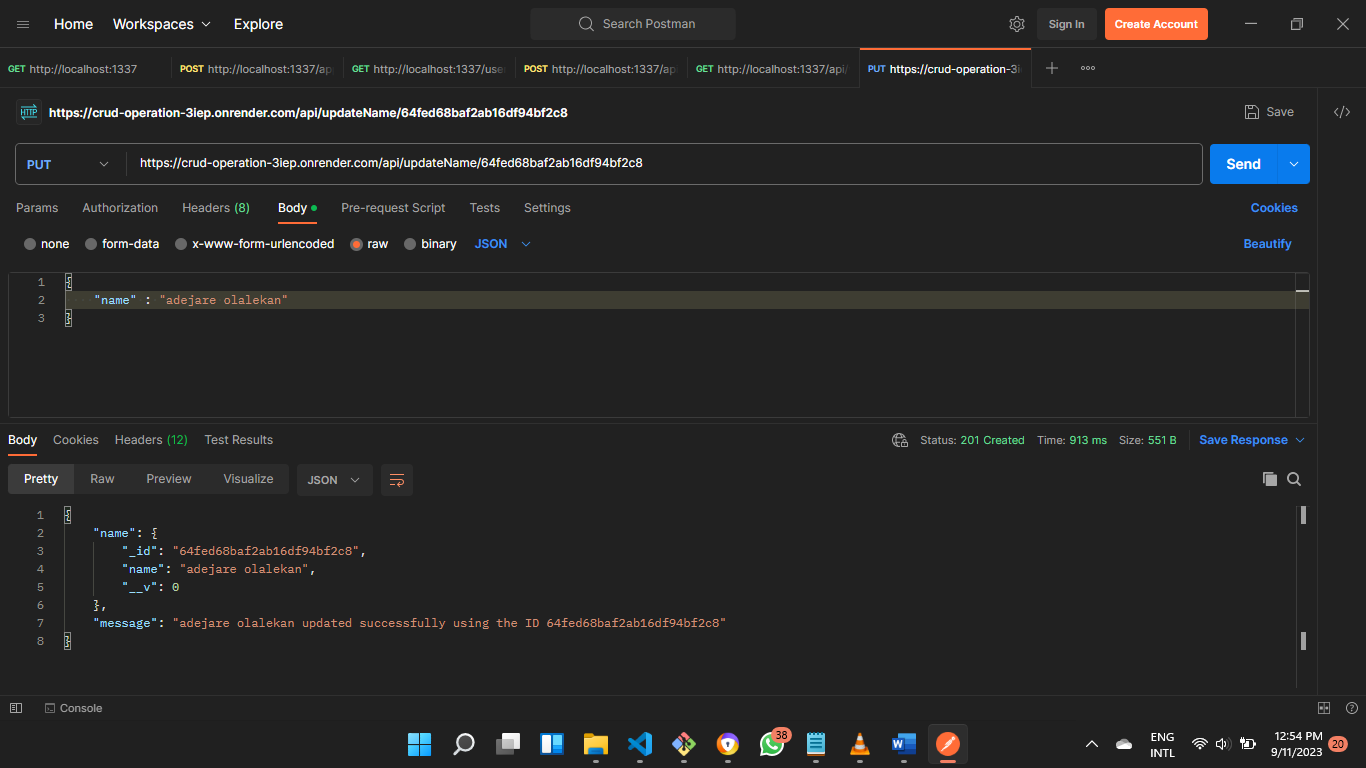
BODY

RESPONSE

UPDATE

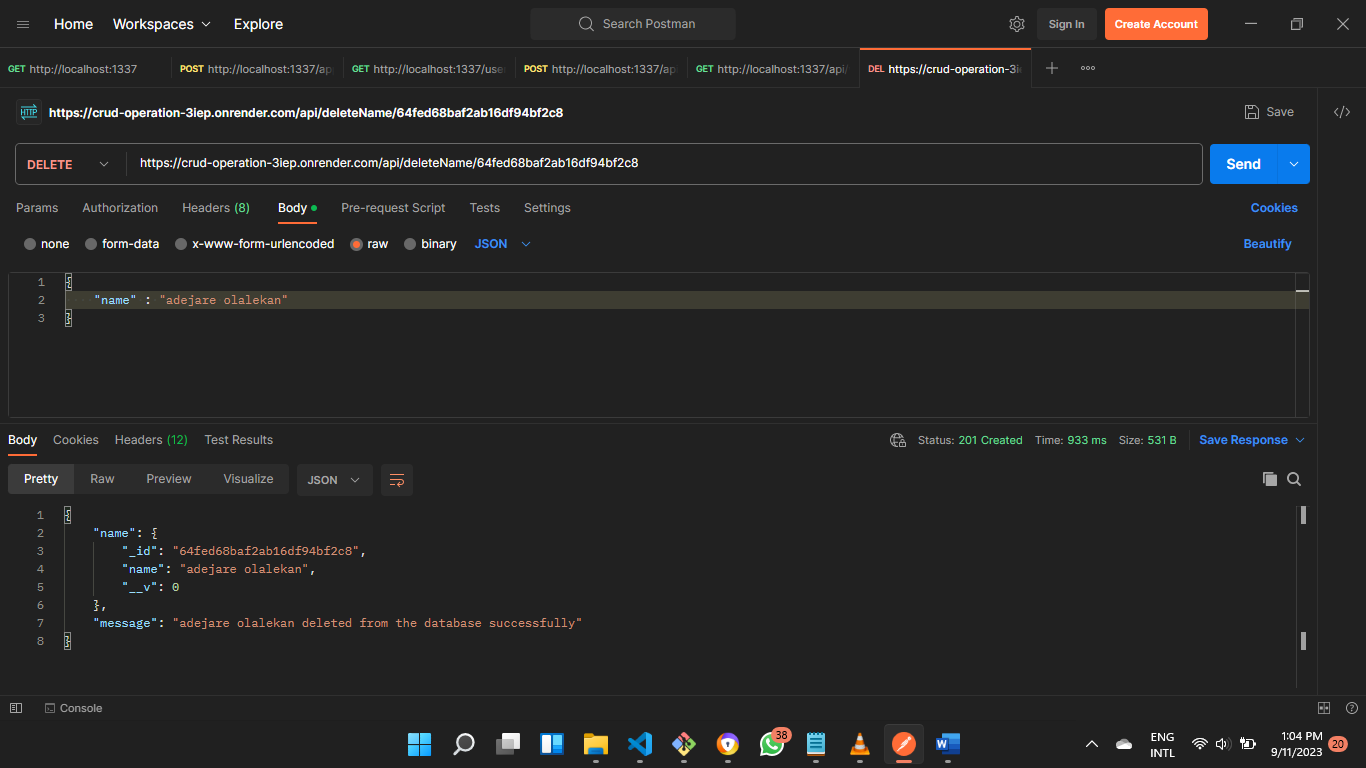
BODY

RESPONSE



DELETE

BODY



RESPONSE

