# Contact Management System

## Introduction

The Contact Management System is a simple web application built with Express.js and MongoDB that allows users to manage their contacts. Users can create, read, update, and delete (CRUD) contact information through a RESTful API. This project serves as a foundational example of how to implement basic CRUD operations using modern web technologies.

## Technology Information

### Backend

* **Node.js**: A JavaScript runtime built on Chrome's V8 JavaScript engine.
* **Express.js**: A minimal and flexible Node.js web application framework that provides robust features for web and mobile applications.
* **MongoDB**: A NoSQL database program that uses JSON-like documents with optional schemas.
* **Mongoose**: An Object Data Modeling (ODM) library for MongoDB and Node.js.

### Middleware

* **body-parser**: A middleware to parse incoming request bodies before your handlers, available under the req.body property.

Step-by-Step Procedure

### 1. Create the project directory

mkdir contact-management

cd contact-management

**2.Initialize a new Node.js project**

npm init -y

**3.Install Necessary dependencies**

npm install express body-parser mongoose

**4.Create a file named ‘index.js’**

touch index.js

**5.Set up a basic Express server and connect to MongoDB:**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

const PORT = 3000;

app.use(bodyParser.json());

const dbURI = 'mongodb+srv://username:password@cluster.mongodb.net/';

mongoose.connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('Connected to MongoDB'))

.catch(err => console.error('Failed to connect to MongoDB', err));

const contactSchema = new mongoose.Schema({

name: String,

email: String,

phone: String

});

const Contact = mongoose.model('Contact', contactSchema);

app.listen(PORT, () => {

console.log(`The server is listening on port ${PORT}`);

});

**6.Implement CRUD Operations**

**> Create a new Contact:**

app.post('/contacts', async (req, res) => {

try {

const contact = new Contact(req.body);

await contact.save();

res.status(201).json(contact);

} catch (error) {

res.status(400).json({ error: error.message });

}

});

**> Get all contact :**

app.get('/contacts', async (req, res) => {

try {

const contacts = await Contact.find();

res.status(200).json(contacts);

} catch (error) {

res.status(500).json({ error: error.message });

}

});

**>Update a contact :**

app.put('/contacts/:id', async (req, res) => {

try {

const contact = await Contact.findByIdAndUpdate(req.params.id, req.body, { new: true, runValidators: true });

if (!contact) {

return res.status(404).json({ error: 'Contact not found' });

}

res.status(200).json(contact);

} catch (error) {

res.status(400).json({ error: error.message });

}

});

**>Delete a contact:**

app.delete('/contacts/:id', async (req, res) => {

try {

const contact = await Contact.findByIdAndDelete(req.params.id);

if (!contact) {

return res.status(404).json({ error: 'Contact not found' });

}

res.status(204).send();

} catch (error) {

res.status(500).json({ error: error.message });

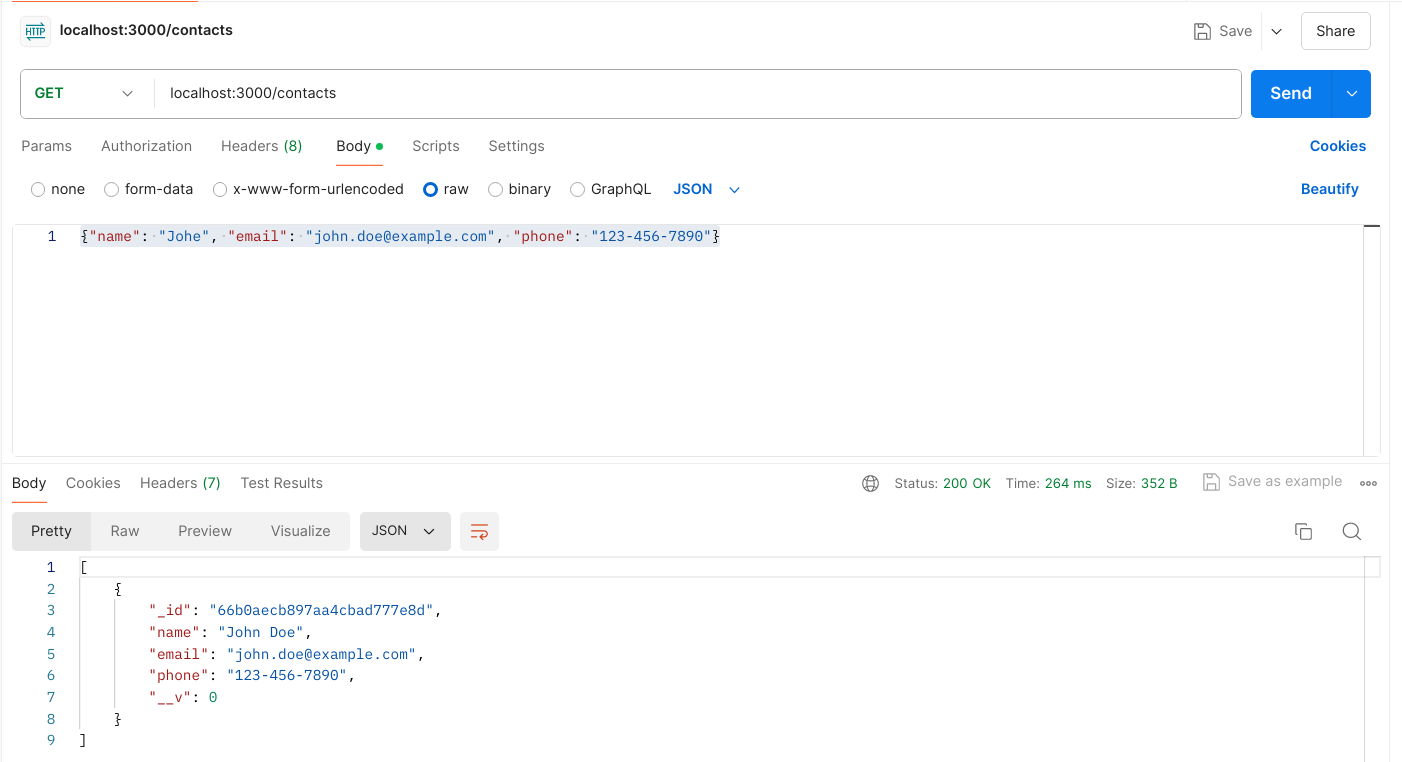
}

});

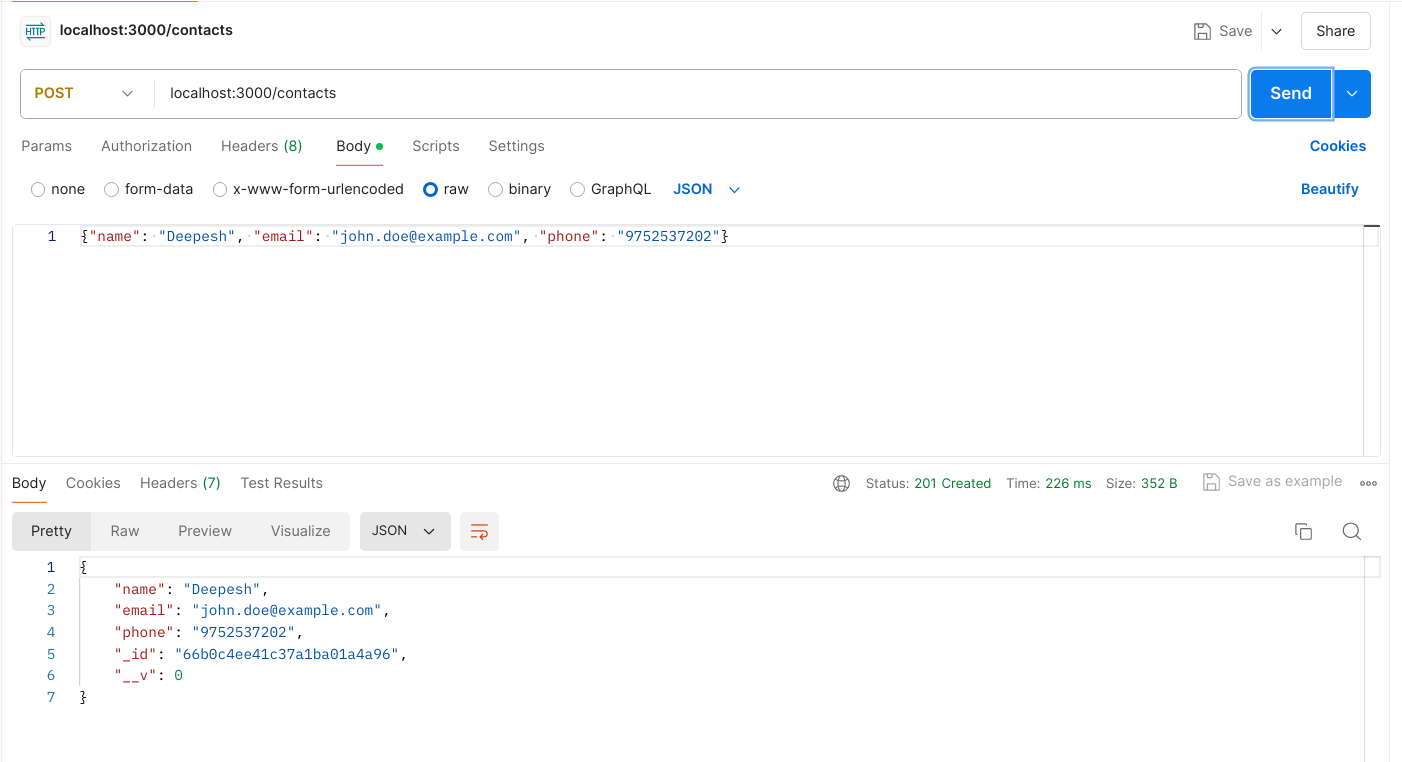
**7. Testing the API**

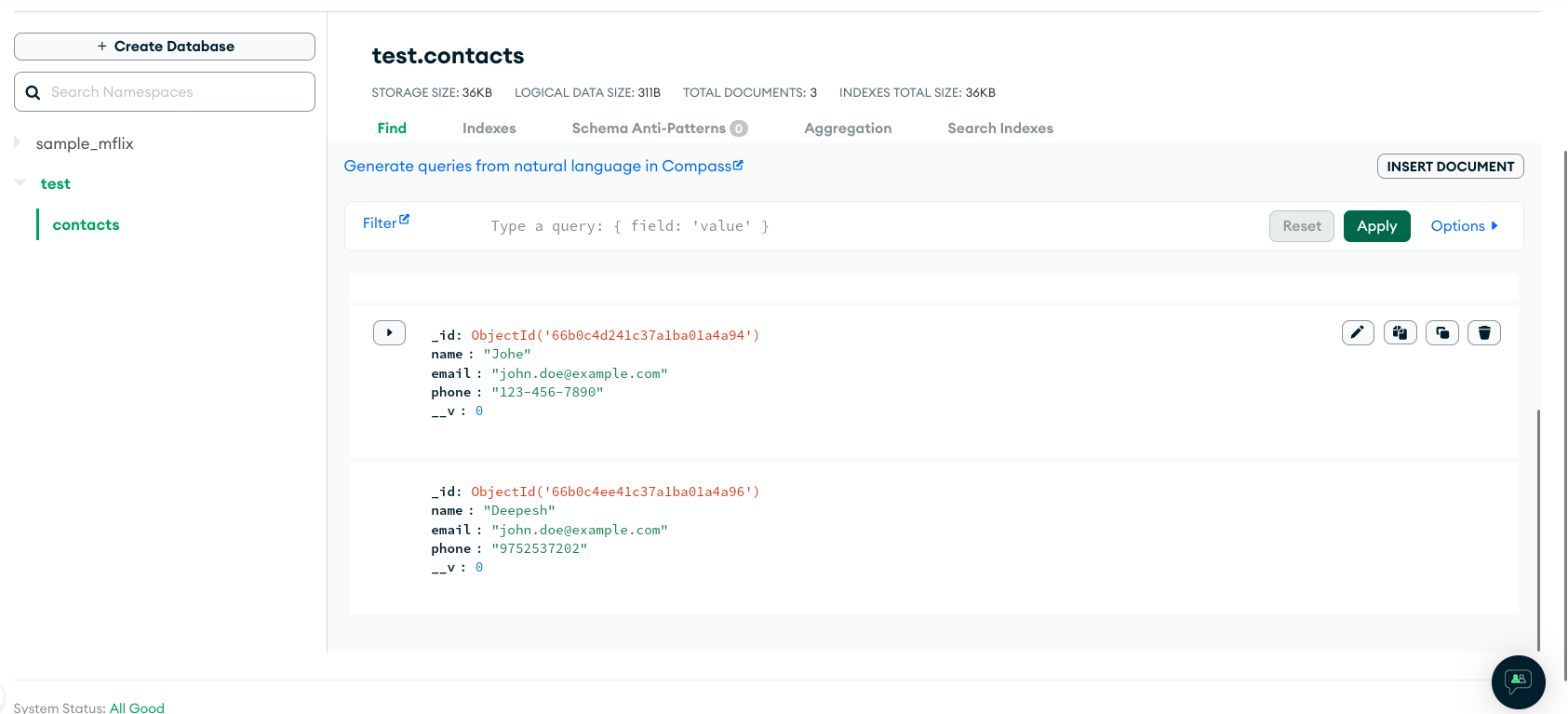
**>** Open postman .

GET :

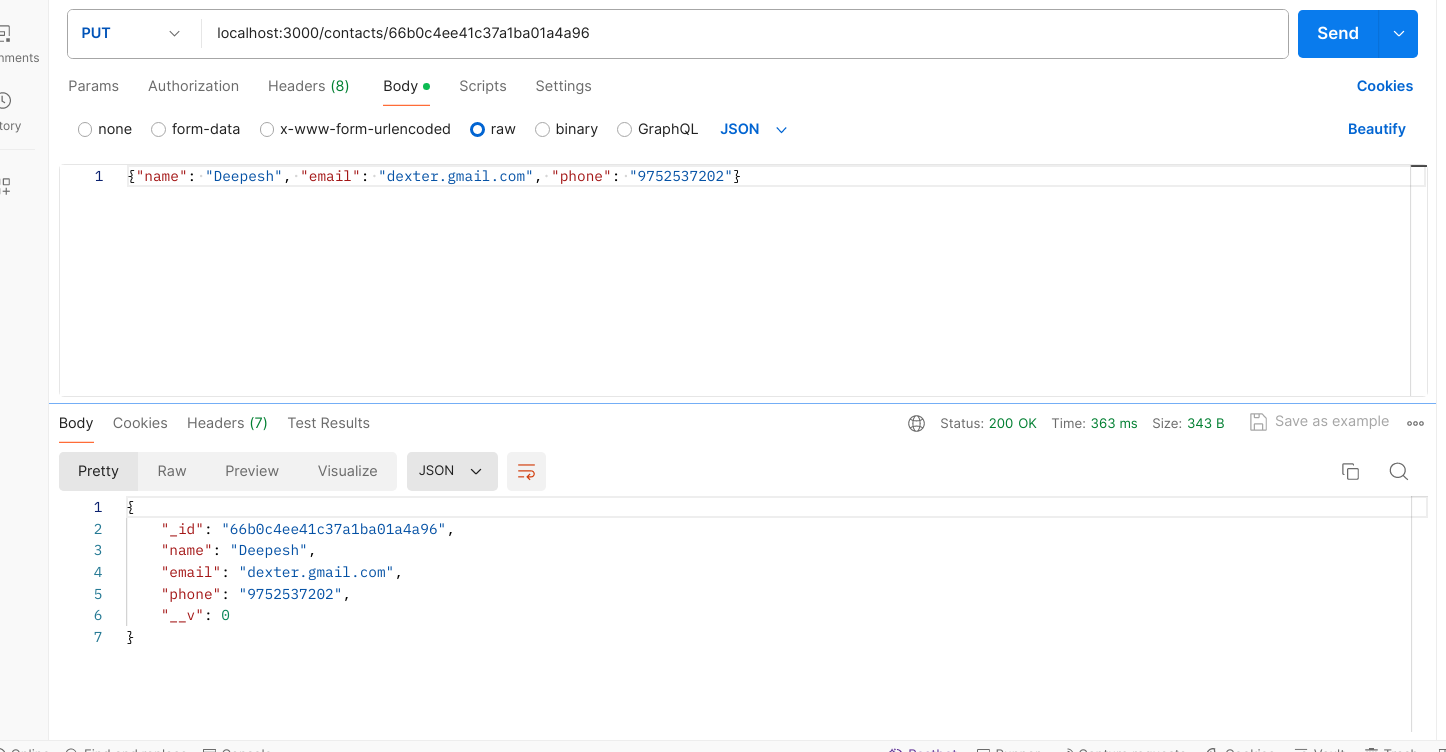
****

POST



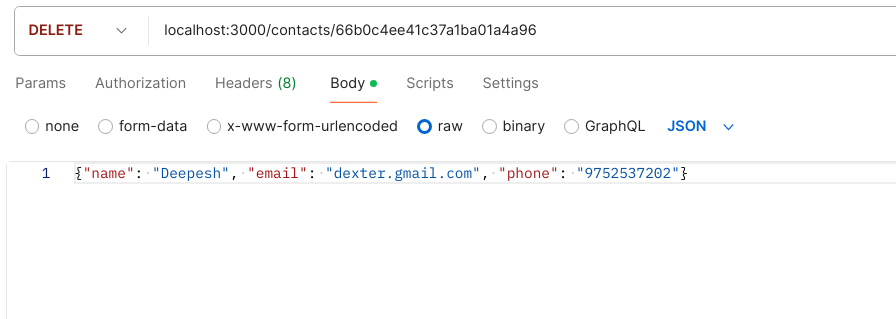
****

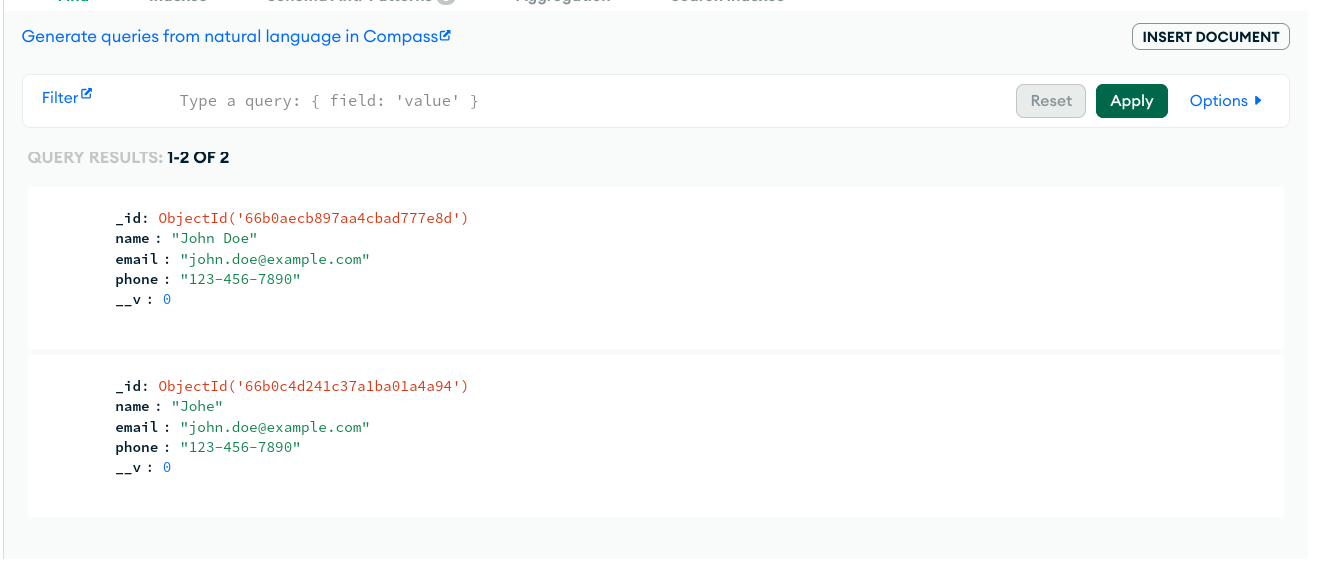
PUT





DELETE





**BIBILOGRAPHY**

**1. Chatgpt**

**2.Youtube  
3.MongoDB Atlas Documentation**

**4. Express Documentation**