CONTACT API UNIT TESTING

About Project:

This Application is based on Node JS and MongoDB

About API:

I have used the POSTMAN API for handling all the POST , GET , PUT and DELETE requests.

And I have used MongoDB for storing data.

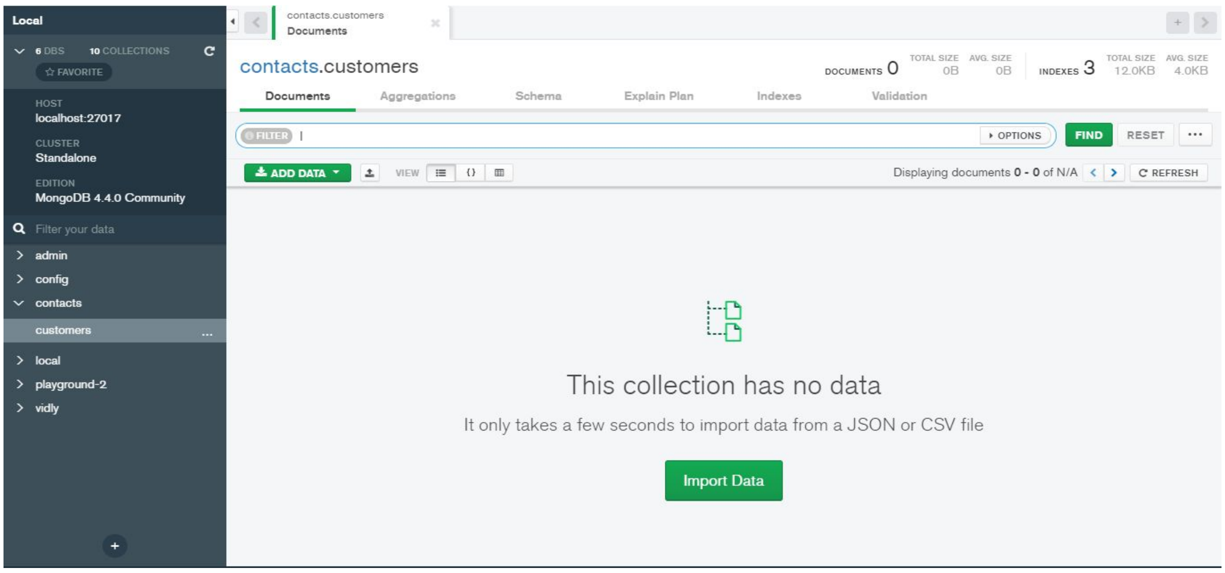
Packages Used:	1. Mongoose(5.0.2)
	2. bcrypt(1.0.3)
	3. express(4.16.2)
	4. joi(13.1.0)

I. Handling POST / Create Requests.

We will pass the below object in the body of request at url=http://localhost:3000/api/customers/

```
{
   "isAdmin": true,
   "Fname": "Atul",
   "Lname": "Verma",
   "phone": "+910000000000",
   "email" : "atulji95@gmail.com",
   "password":"Atul@123"
}
```

Screenshot of database before saving data

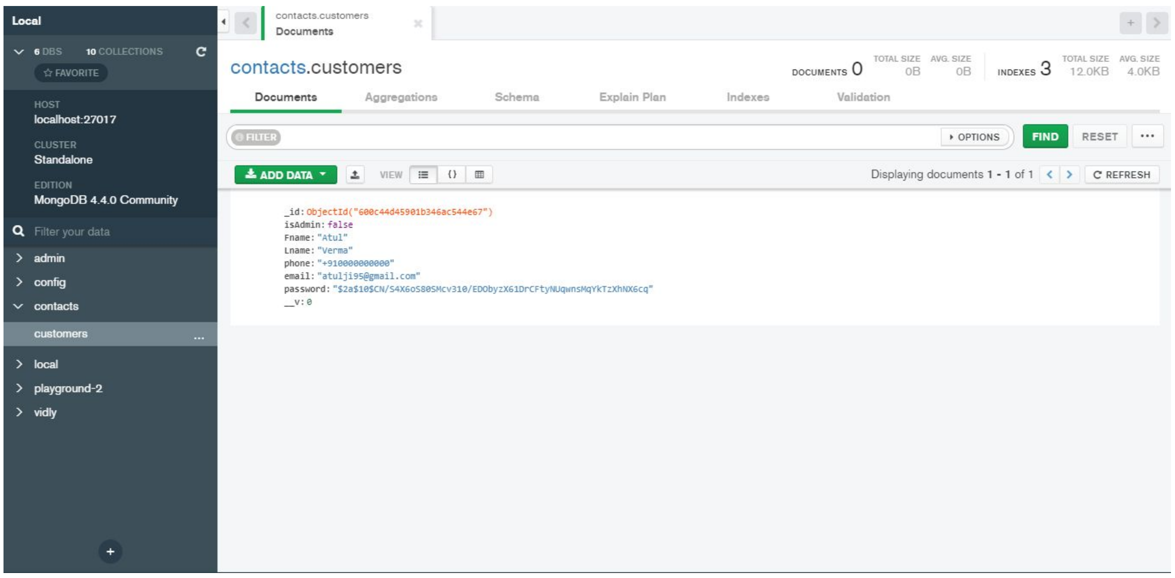


Displaying documents 0 - 0 of N/A

This collection has no data

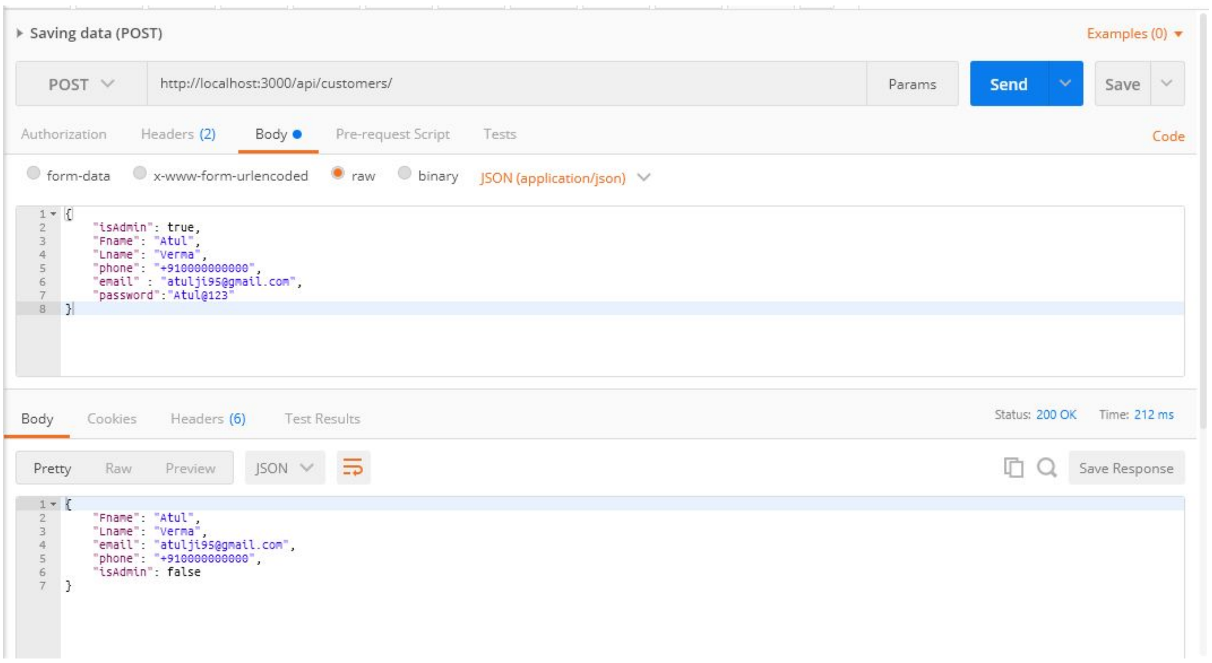
It only takes a few seconds to import data from a JSON or CSV file

Screenshot of database after saving data


Local
6 DBS 10 COLLECTIONS
FAVORITE
HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 4.4.0 Community
Filter your data
admin
config
contacts
customers
local
playground-2
vidly
contacts.customers Documents
contacts.customers
DOCUMENTS 0 TOTAL SIZE 0B AVG. SIZE 0B
INDEXES 3 TOTAL SIZE 12.0KB AVG. SIZE 4.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 1 of 1 REFRESH

```
_id: ObjectId("600c44d45901b346ac544e67")
isAdmin: false
Fname: "Atul"
Lname: "Verma"
phone: "+910000000000"
email: "atulji95@gmail.com"
password: "$2a$10$CN/S4X6oS80SMcv310/EDObyzX61DrCFtyNUqwnsMqYkTzXhNX6cq"
__v: 0
```

Screenshot of postman after saving data


Saving data (POST)
Examples (0)
POST
Params
Send
Save
Authorization Headers (2) Body Pre-request Script Tests
Code
form-data x-www-form-urlencoded raw binary JSON (application/json)
Body Cookies Headers (6) Test Results
Status: 200 OK Time: 212 ms
Pretty Raw Preview JSON
Save Response

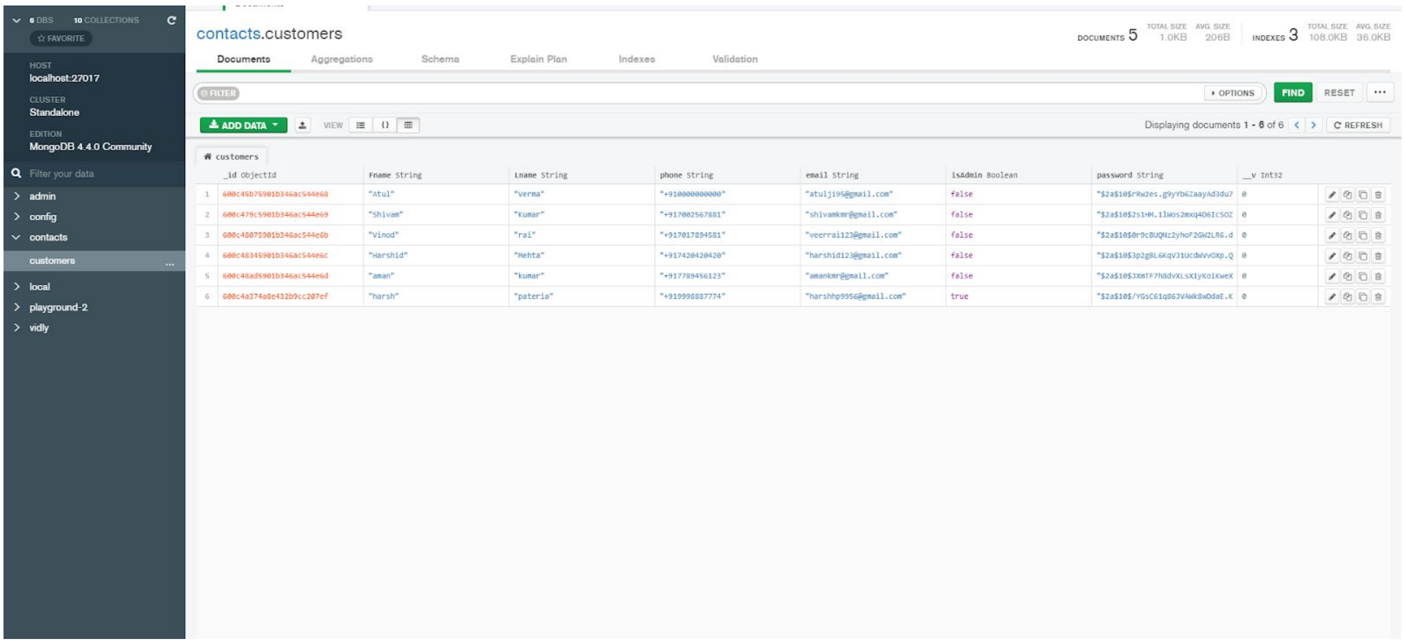
http://localhost:3000/api/customers/

```
{
    "isAdmin": true,
    "Fname": "Atul",
    "Lname": "Verma",
    "phone": "+910000000000",
    "email" : "atulji95@gmail.com",
    "password":"Atul@123"
}
```

```
{
    "Fname": "Atul",
    "Lname": "Verma",
    "email": "atulji95@gmail.com",
    "phone": "+910000000000",
    "isAdmin": false
}
```

II. Handling GET / Read Requests.

Screenshot of database after inserting some more data

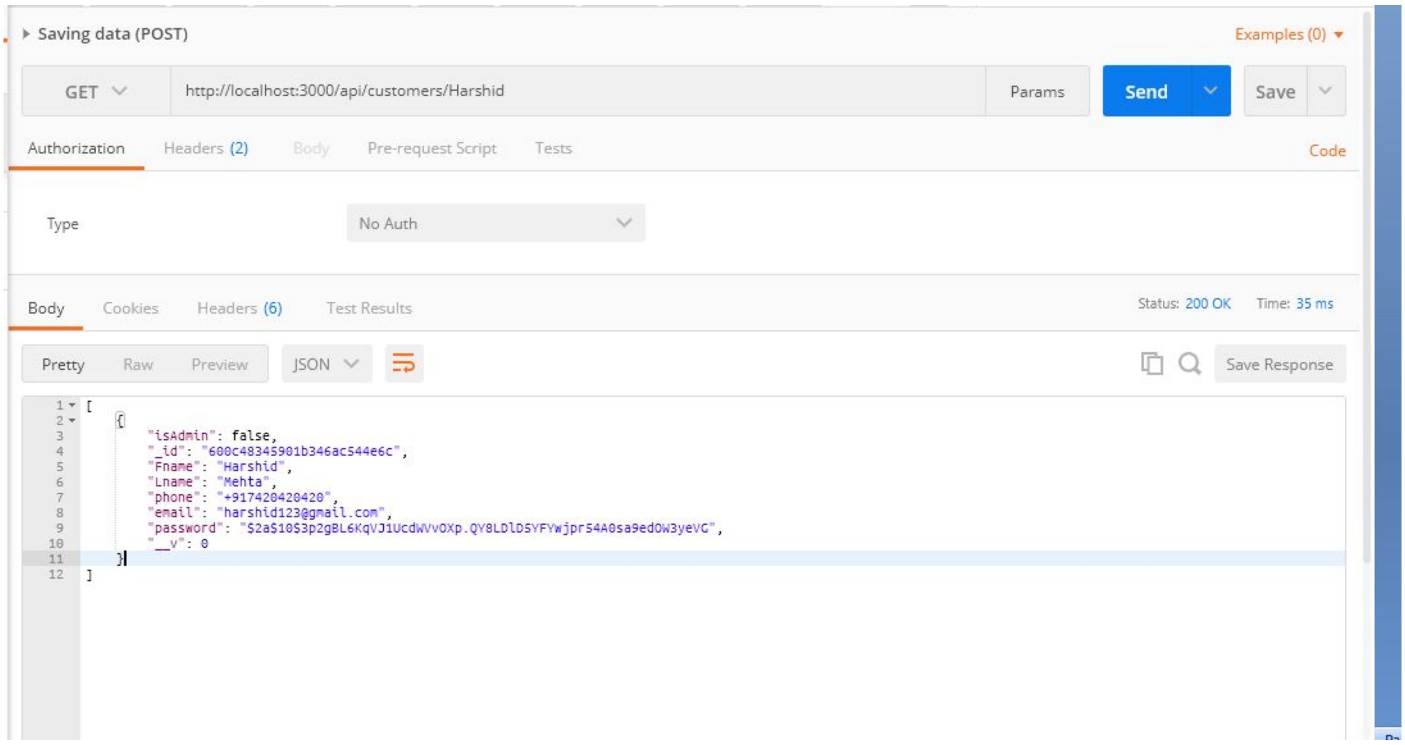


	_id ObjectId	Fname String	Lname String	phone String	email String	isAdmin Boolean	password String	__v Int32
1	600c45b75901b346ac544e68	"Atul"	"Verma"	"+910000000000"	"atulji95@gmail.com"	false	"$2a$10$rRw2es.g9yYb6ZaayAd3du7	0
2	600c479c5901b346ac544e69	"Shivam"	"Kumar"	"+917002567881"	"shivamkmr@gmail.com"	false	"$2a$10$2s1HM.1lWos2mxq4O6IcSOZ	0
3	600c48075901b346ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false	"$2a$10$0r9cBUQNz2yhoF2GW2LR6.d	0
4	600c48345901b346ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false	"$2a$10$3p2gBL6KqVJ1UcdWVvOXp.Q	0
5	600c48ad5901b346ac544e6d	"aman"	"kumar"	"+917789456123"	"amankmr@gmail.com"	false	"$2a$10$JXmTF7h8dvXLsX1yKoiKweX	0
6	600c4a374a8e432b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true	"$2a$10$/YGsC61q86JVAWk8wDdaE.K	0

A. Fetching data by username (Fname)

We will fetch data whose username is Harshid by using URL: http://localhost:3000/api/customers/Harshid

Screenshot of postman after Fetching data



GET http://localhost:3000/api/customers/Harshid

Type: No Auth

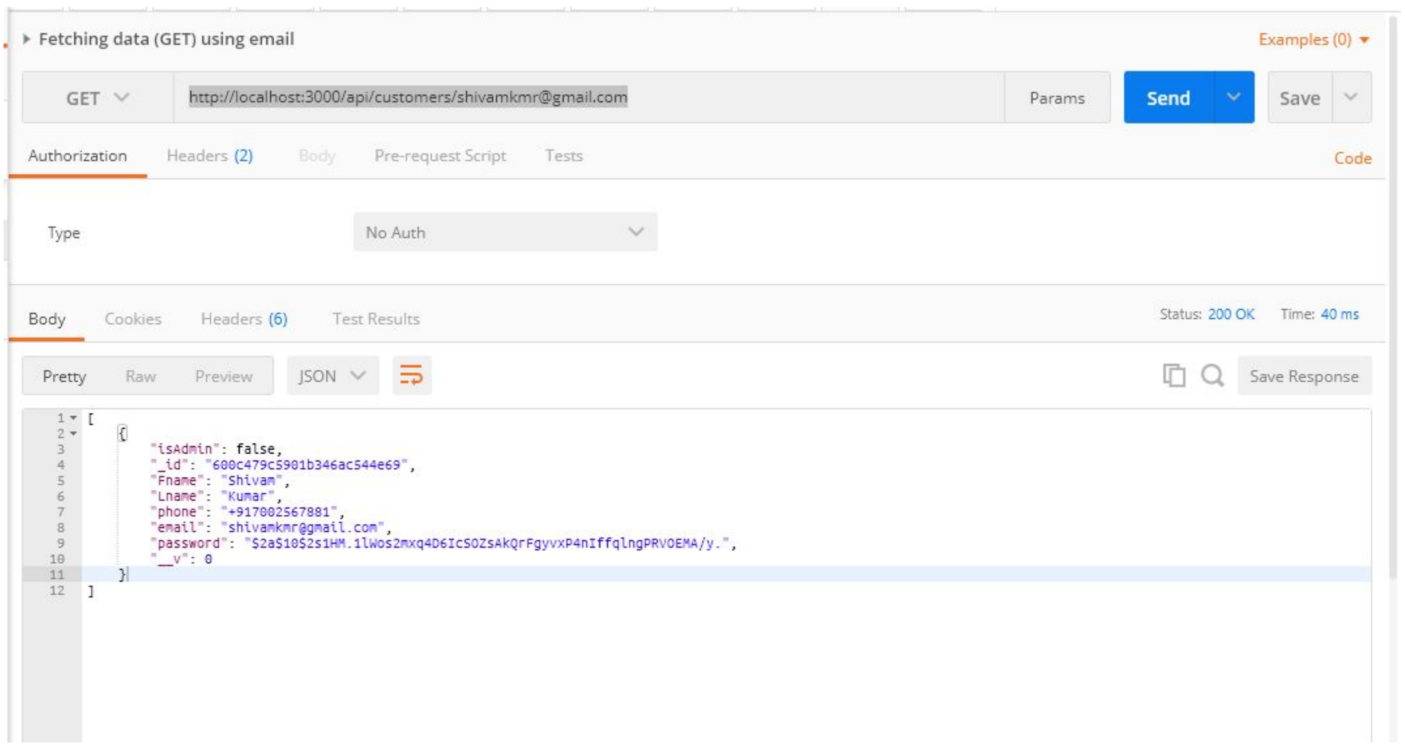
Status: 200 OK Time: 35 ms

```
[
    {
        "isAdmin": false,
        "_id": "600c48345901b346ac544e6c",
        "Fname": "Harshid",
        "Lname": "Mehta",
        "phone": "+917420420420",
        "email": "harshid123@gmail.com",
        "password": "$2a$10$3p2gBL6KqVJ1UcdWVvOXp.QY8LDlD5YFYwjpr54A0sa9edOW3yeVG",
        "__v": 0
    }
]
```

B. Fetching data by email ID (email)

We will fetch data whose email is shivamkmr@gmail.com by using
URL: http://localhost:3000/api/customers/shivamkmr@gmail.com

Screenshot of postman after Fetching data



http://localhost:3000/api/customers/shivamkmr@gmail.com

Type: No Auth

Status: 200 OK Time: 40 ms

```
[
    {
        "isAdmin": false,
        "_id": "600c479c5901b346ac544e69",
        "Fname": "Shivam",
        "Lname": "Kumar",
        "phone": "+917002567881",
        "email": "shivamkmr@gmail.com",
        "password": "$2a$10$2s1HM.1lWos2mxq4D6IcSOZsAkQrFgyvxP4nIffqlngPRVOEMA/y.",
        "__v": 0
    }
]
```

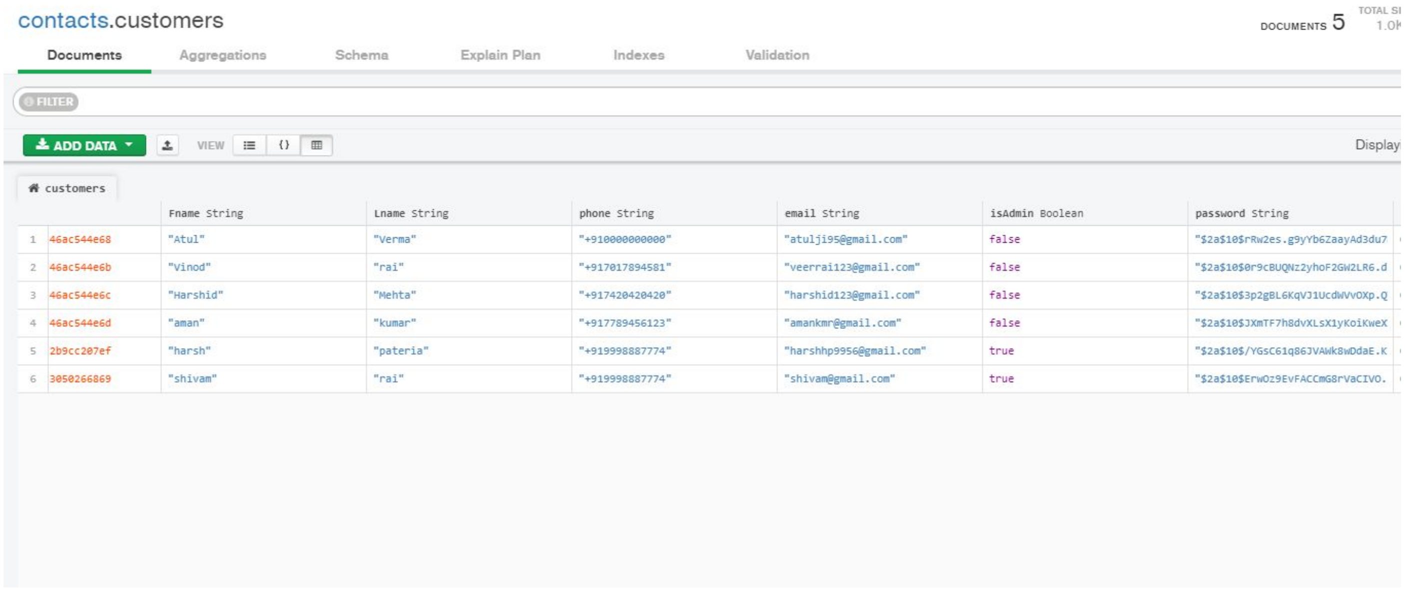
III. Handling PUT / Update Requests.

Note: Here we are passing the email ID in the URL and also checking if the password in the body of the HTTP request actually matches with the password of that actual saved data. So you cannot update your mail ID and password.

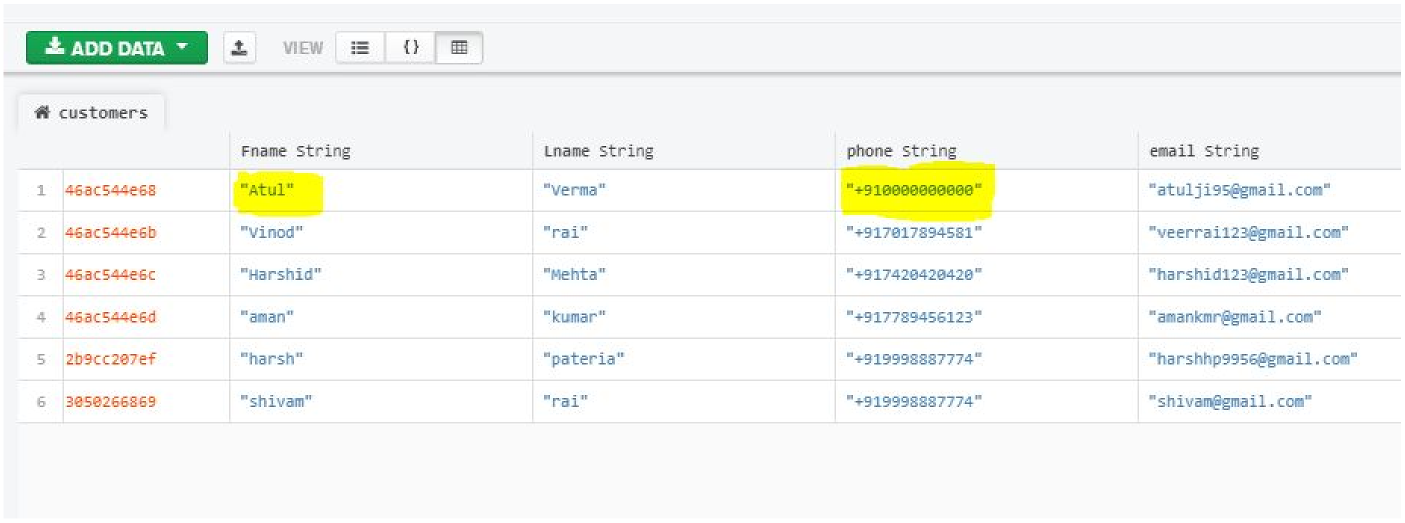
A. Updating data by email ID (email) and with correct password

Here we are changing the name of atul to Abhishek and his phone number to +918008008123

Screenshot of database before update

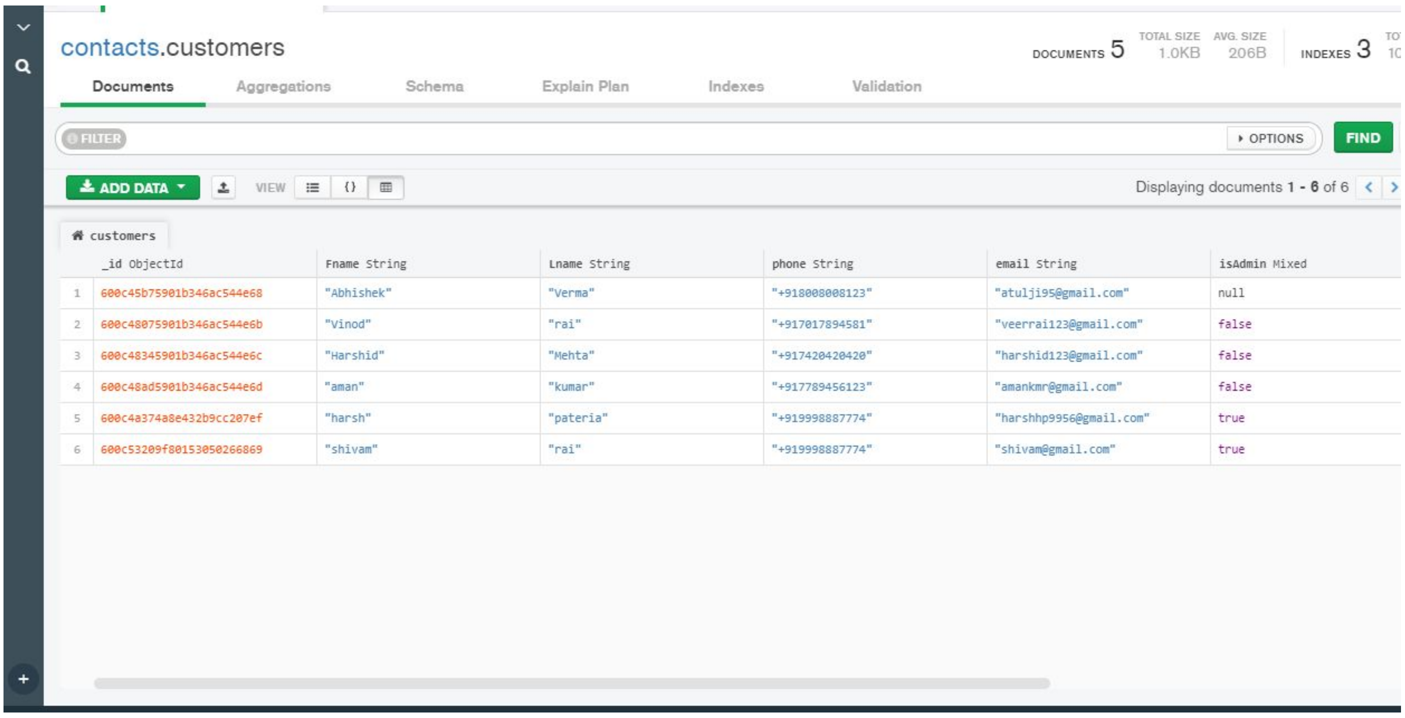


		Fname String	Lname String	phone String	email String	isAdmin Boolean	password String
1	46ac544e68	"Atul"	"Verma"	"+910000000000"	"atulji95@gmail.com"	false	"$2a$10$rRw2es.g9yYb6ZaayAd3du7
2	46ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false	"$2a$10$0r9cBUQNz2yhoF2GW2LR6.d
3	46ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false	"$2a$10$3p2gBL6KqVJ1UcdWVvOXp.Q
4	46ac544e6d	"aman"	"kumar"	"+917789456123"	"amankmr@gmail.com"	false	"$2a$10$JXmTF7h8dvXLsX1yKoiKweX
5	2b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true	"$2a$10$/YGsC61q86JVAWk8wDdaE.K
6	3050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	true	"$2a$10$ErwOz9EvFACCmG8rVaCIVO.



		Fname String	Lname String	phone String	email String
1	46ac544e68	"Atul"	"Verma"	"+910000000000"	"atulji95@gmail.com"
2	46ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"
3	46ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"
4	46ac544e6d	"aman"	"kumar"	"+917789456123"	"amankmr@gmail.com"
5	2b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"
6	3050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"

Screenshot of database after update


contacts.customers

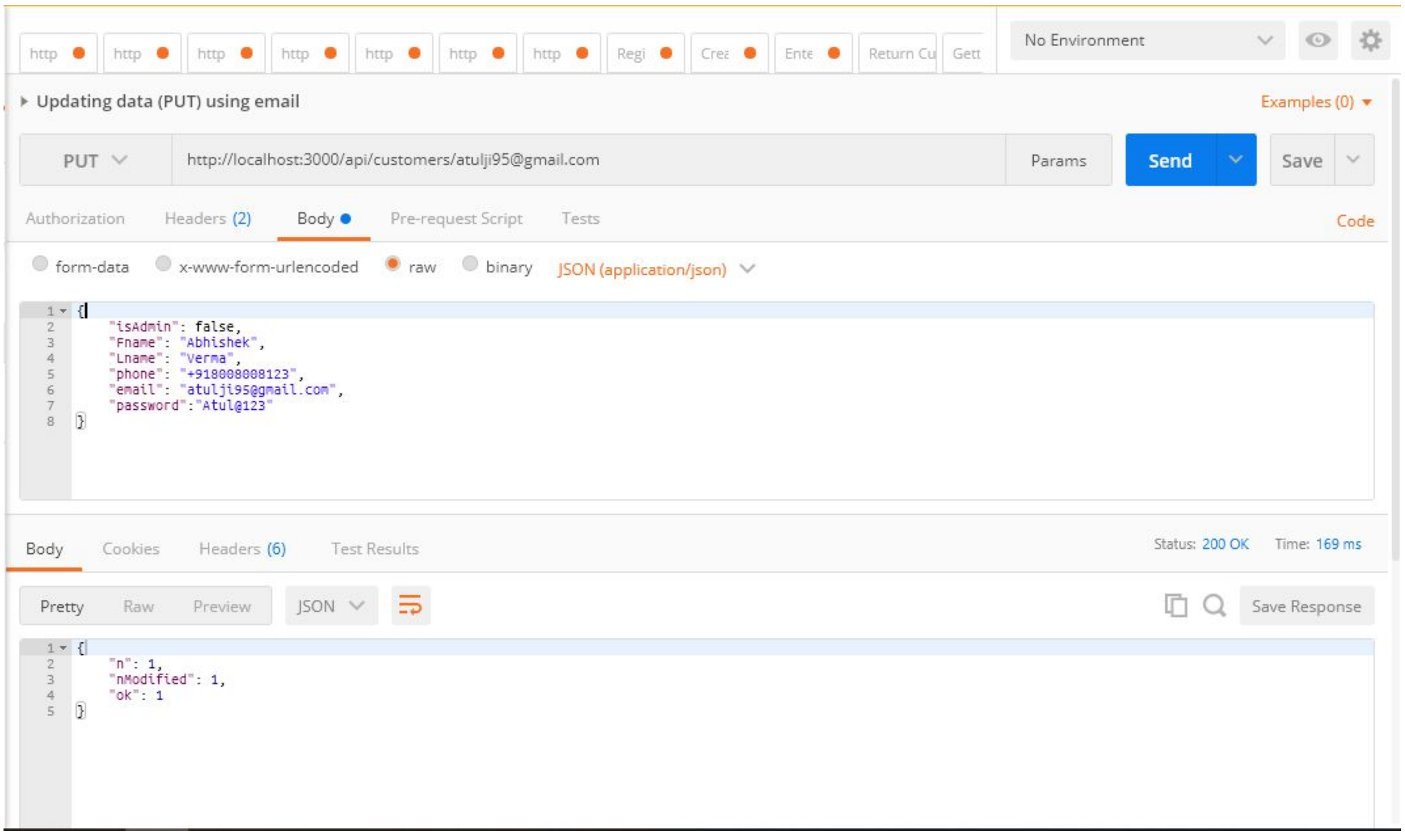
DOCUMENTS 5 TOTAL SIZE 1.0KB AVG. SIZE 206B INDEXES 3

Displaying documents 1 - 6 of 6

customers

	_id ObjectId	Fname String	Lname String	phone String	email String	isAdmin Mixed
1	600c45b75901b346ac544e68	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	null
2	600c48075901b346ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false
3	600c48345901b346ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false
4	600c48ad5901b346ac544e6d	"aman"	"kumar"	"+917789456123"	"amankmr@gmail.com"	false
5	600c4a374a8e432b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true
6	600c53209f80153050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	true

Screenshot of postman after Updating database


Updating data (PUT) using email

PUT http://localhost:3000/api/customers/atulji95@gmail.com

Body: raw — JSON (application/json)

```
{
    "isAdmin": false,
    "Fname": "Abhishek",
    "Lname": "Verma",
    "phone": "+918008008123",
    "email": "atulji95@gmail.com",
    "password":"Atul@123"
}
```

Status: 200 OK Time: 169 ms

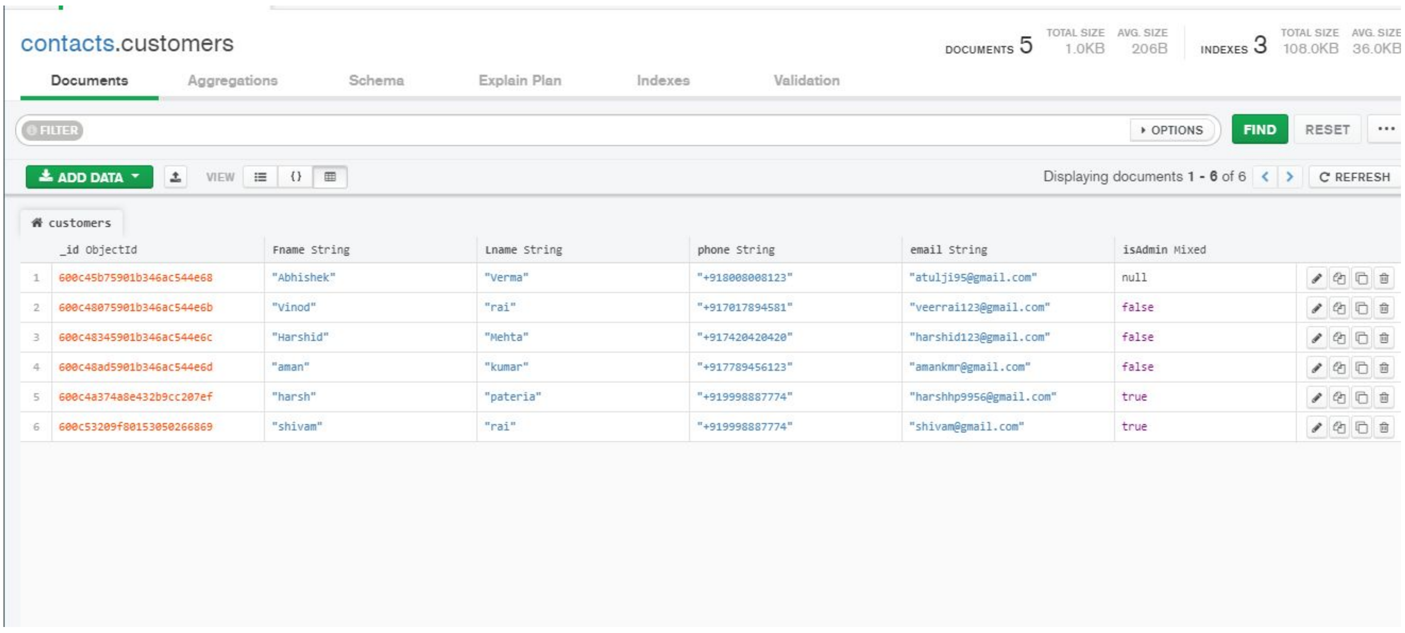
```
{
    "n": 1,
    "nModified": 1,
    "ok": 1
}
```

IV. Handling DELETE Requests.

Note: Here we are passing email ID in the URL and also passing password in the body of HTTP request, if it matches with the password of that actual saved data then we delete that data else error message is displayed. So you cannot delete a contact if you don't know its password and email ID.

Here we a deleting a contact whose email ID is amankmr@gmail.com and whose password is aman@123

Screenshot of database before delete operation



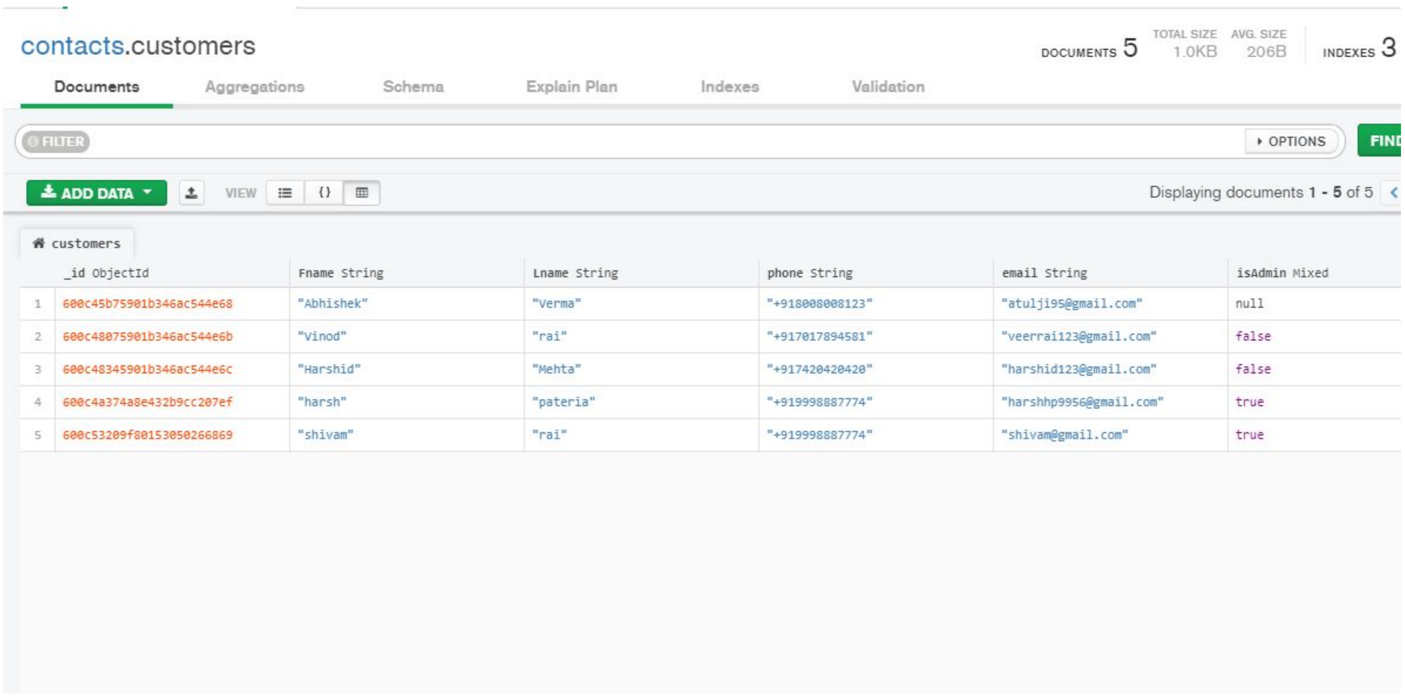
contacts.customers — DOCUMENTS 5, TOTAL SIZE 1.0KB, AVG. SIZE 206B; INDEXES 3, TOTAL SIZE 108.0KB, AVG. SIZE 36.0KB

Displaying documents 1 - 6 of 6

customers

	_id ObjectId	Fname String	Lname String	phone String	email String	isAdmin Mixed
1	600c45b75901b346ac544e68	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	null
2	600c48075901b346ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false
3	600c48345901b346ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false
4	600c48ad5901b346ac544e6d	"aman"	"kumar"	"+917789456123"	"amankmr@gmail.com"	false
5	600c4a374a8e432b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true
6	600c53209f80153050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	true

Screenshot of database after delete operation



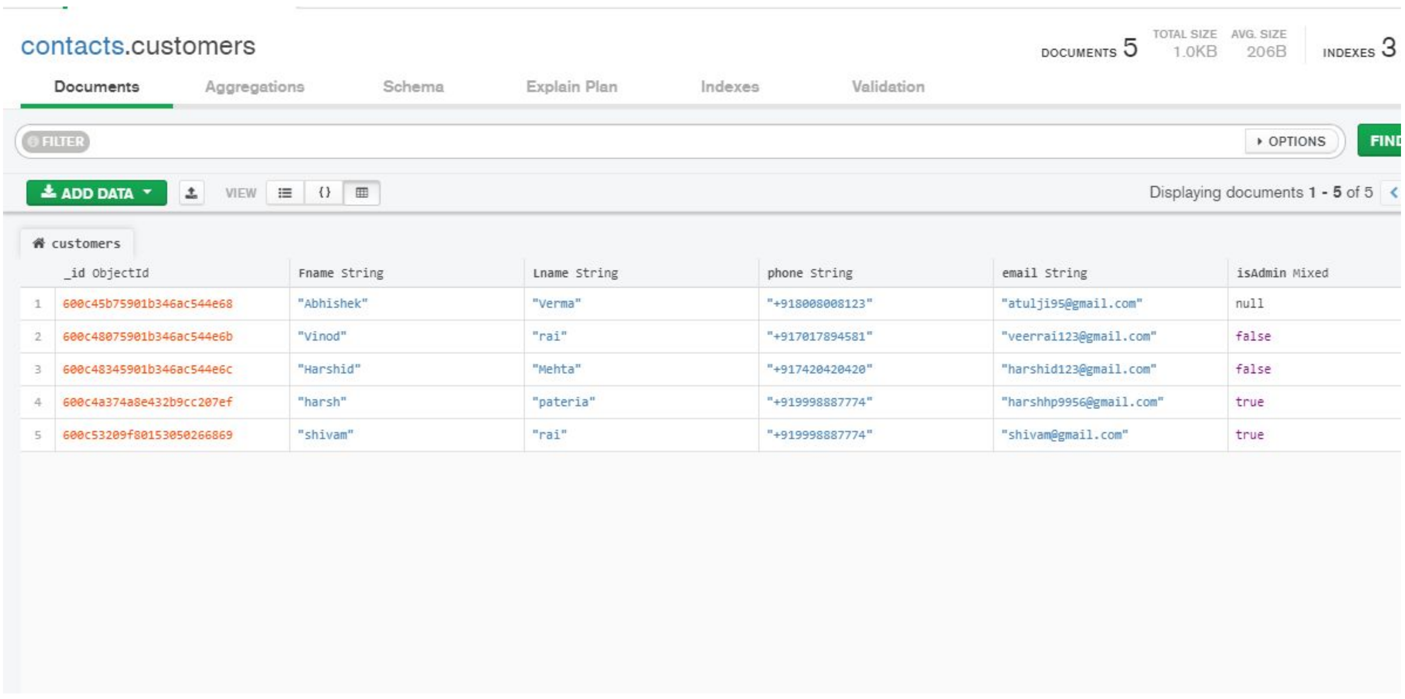
contacts.customers — DOCUMENTS 5, TOTAL SIZE 1.0KB, AVG. SIZE 206B; INDEXES 3

Displaying documents 1 - 5 of 5

customers

	_id ObjectId	Fname String	Lname String	phone String	email String	isAdmin Mixed
1	600c45b75901b346ac544e68	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	null
2	600c48075901b346ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false
3	600c48345901b346ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false
4	600c4a374a8e432b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true
5	600c53209f80153050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	true

Screenshot of postman after DELETE Operation



	_id ObjectId	Fname String	Lname String	phone String	email String	isAdmin Mixed
1	600c45b75901b346ac544e68	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	null
2	600c48075901b346ac544e6b	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	false
3	600c48345901b346ac544e6c	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	false
4	600c4a374a8e432b9cc207ef	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	true
5	600c53209f80153050266869	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	true

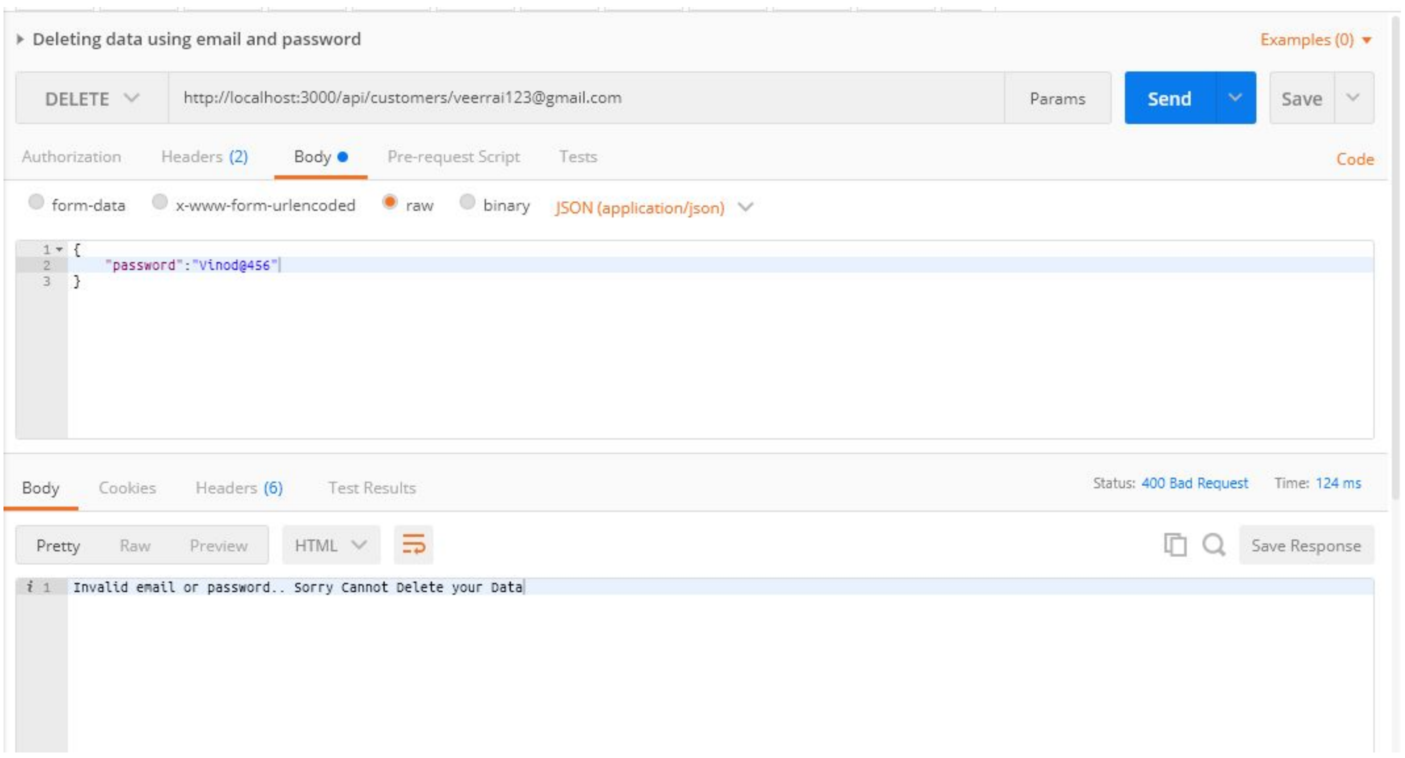
AUTHENTICATION

About: here 'bcrypt' npm library is for hashing the password given by the user and then this hashed password is saved into the database which is no longer usable by the user.

1.**For DELETE Operation**: While deleting a contact , one must know the email Id and original password in order to delete that contact from the database. This password is send through the body of the HTTPS request send by the server

Here we will pass a deleted request from the server with the wrong password to delete contact with mail Id veerrai123@gmail.com whose actual password is Vinod@123 but we will pass wrong password Vinod@456

Screenshot of postman after DELETE Operation with wrong password



http://localhost:3000/api/customers/veerrai123@gmail.com

```
{
    "password":"Vinod@456"
}
```

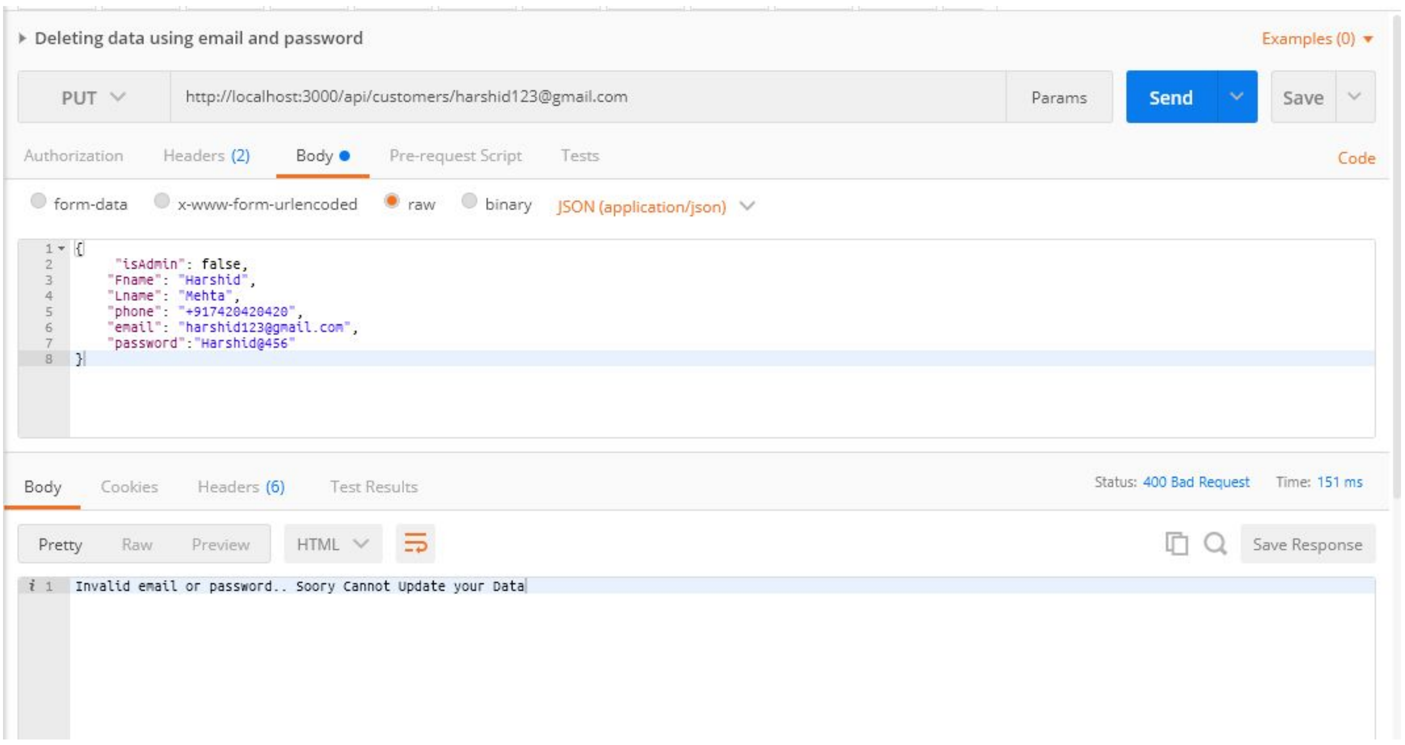
Status: 400 Bad Request Time: 124 ms

```
Invalid email or password.. Sorry Cannot Delete your Data
```

2. **<u>For PUT / Update Operation:</u>** While updating a contact, user must the correct password and email Id in the body of HTTP request and URL respectively. If this password doesn't match the original password then that contact is not updated.

Here we will pass a PUT request from the server containing the wrong password to UPDATE contact with mail Id `harshid123@gmail.com` whose actual password is Harshid@123 but we will pass wrong password Harshid@456

<u>Screenshot of postman after PUT Operation with wrong password</u>



PUT http://localhost:3000/api/customers/harshid123@gmail.com

Body: raw, JSON (application/json)

```
{
    "isAdmin": false,
   "Fname": "Harshid",
   "Lname": "Mehta",
   "phone": "+917420420420",
   "email": "harshid123@gmail.com",
   "password":"Harshid@456"
}
```

Status: 400 Bad Request Time: 151 ms

```
Invalid email or password.. Soory Cannot Update your Data
```

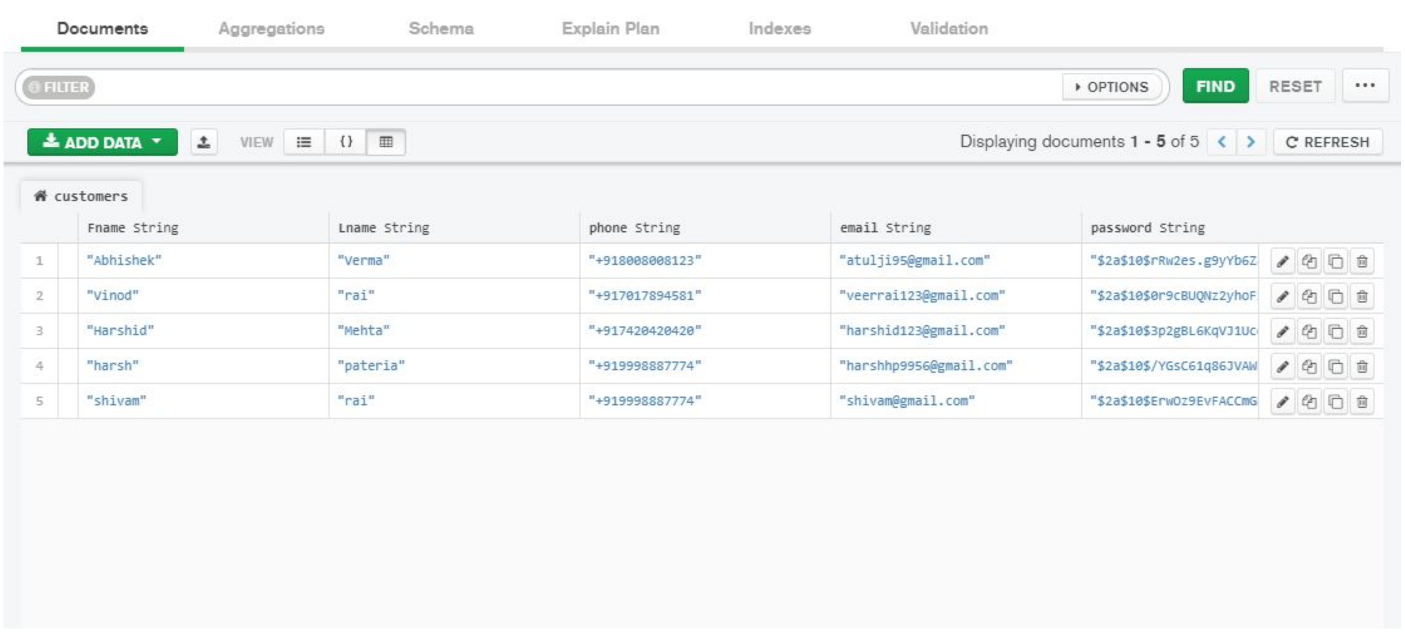
Handling Errors and Exceptions

1. Handling POST / Create Request Errors

A. Handling insertion of Duplicate mail ID's

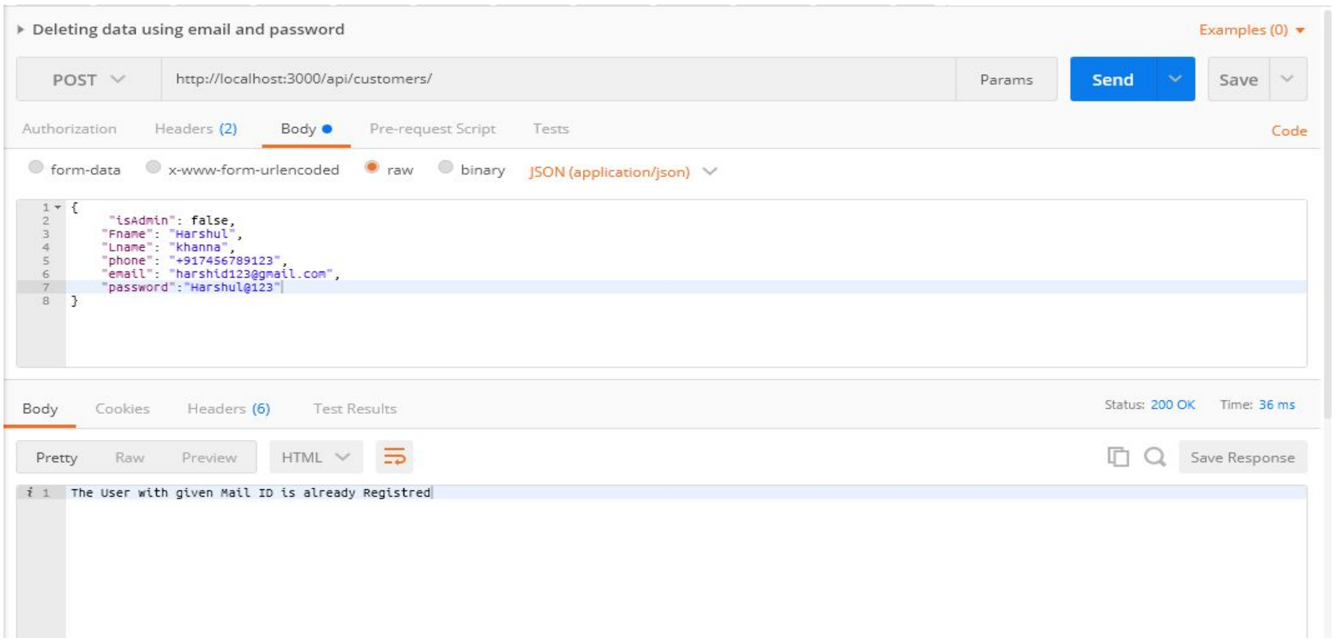
A contact with mail Id= harshid123@gmail.com already exists in the database. So we will willingly try to insert another data with the same mail Id into the database.

Screenshot of database before Invalid POST Operation



	Fname String	Lname String	phone String	email String	password String
1	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	"$2a$10$rRw2es.g9yYb6Z
2	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	"$2a$10$0r9cBUQNz2yhoF
3	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	"$2a$10$3p2gBL6KqVJ1Uc
4	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	"$2a$10$/YGsC61q86JVAW
5	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	"$2a$10$ErwOz9EvFACCmG

Screenshot of postman after Invalid POST Operation



POST http://localhost:3000/api/customers/

JSON (application/json)

```
{
     "isAdmin": false,
    "Fname": "Harshul",
    "Lname": "khanna",
    "phone": "+917456789123",
    "email": "harshid123@gmail.com",
    "password":"Harshul@123"
}
```

Status: 200 OK Time: 36 ms

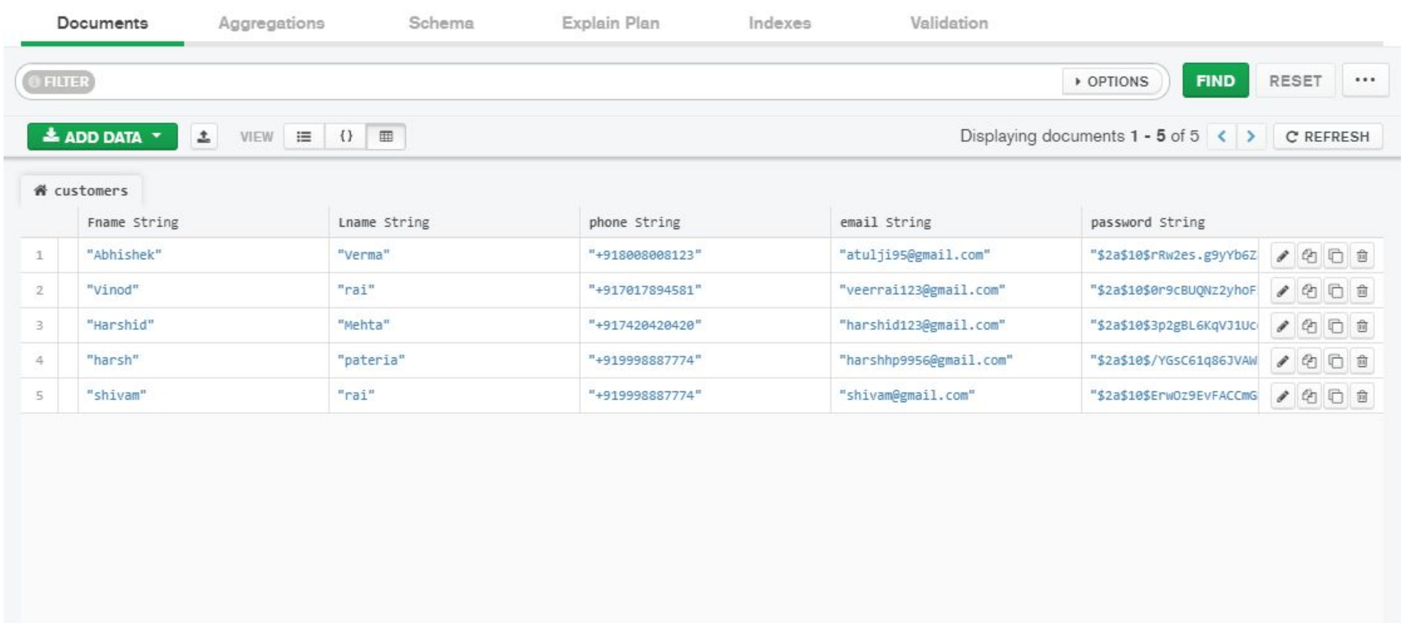
```
The User with given Mail ID is already Registred
```

2. Handling PUT / Update Request Errors

A. Handling error when user passes Non-Existing mail ID in HTTP request body

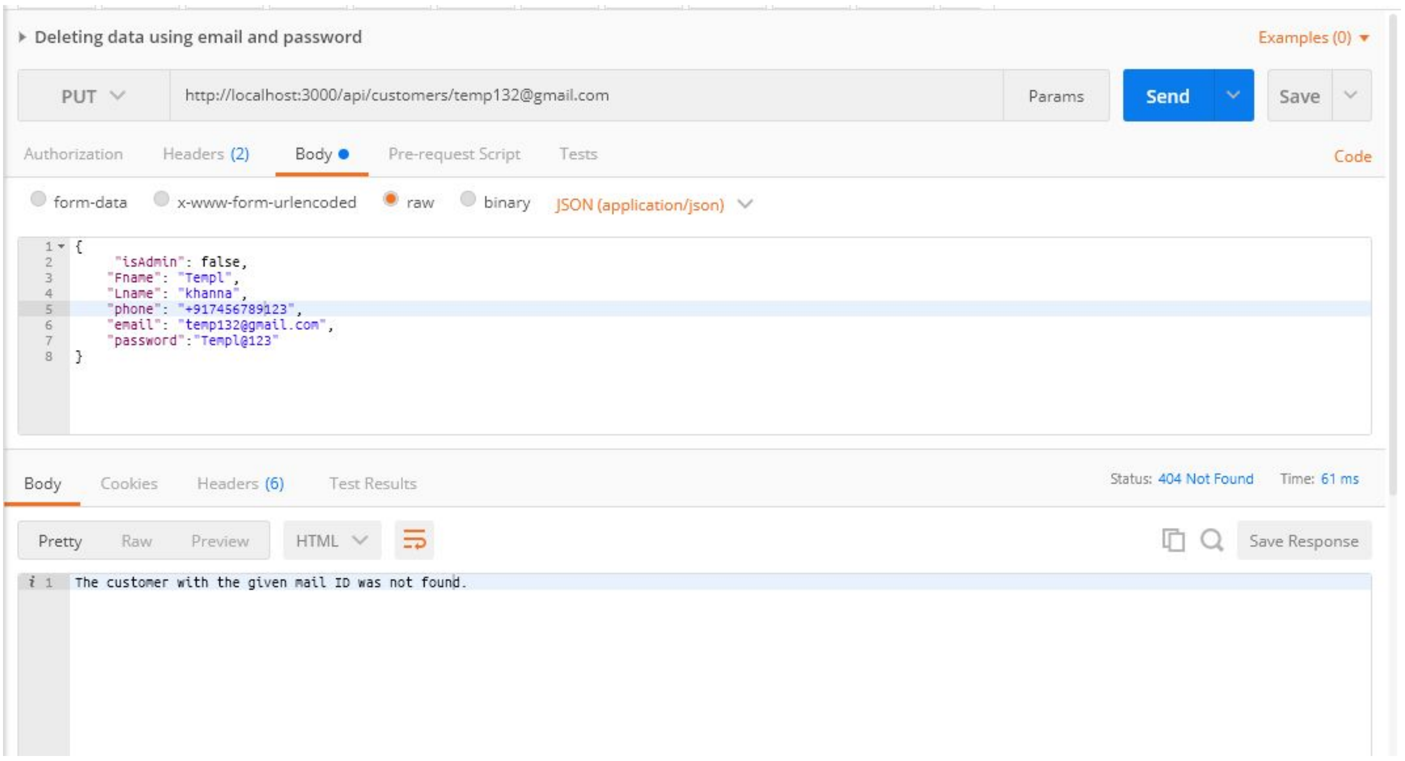
We will try to update a contact while passing a mail Id = temp132@gmail.com of which contact doesn't exist in the database.

Screenshot of database before Invalid PUT Operation



	Fname String	Lname String	phone String	email String	password String
1	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	"$2a$10$rRw2es.g9yYb6Z
2	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	"$2a$10$0r9cBUQNz2yhoF
3	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	"$2a$10$3p2gBL6KqVJ1Uc
4	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	"$2a$10$/YGsC61q86JVAW
5	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	"$2a$10$ErwOz9EvFACCmG

Screenshot of postman after Invalid PUT Operation



PUT http://localhost:3000/api/customers/temp132@gmail.com

```
{
    "isAdmin": false,
   "Fname": "Templ",
   "Lname": "khanna",
   "phone": "+917456789123",
   "email": "temp132@gmail.com",
   "password":"Templ@123"
}
```

Status: 404 Not Found Time: 61 ms

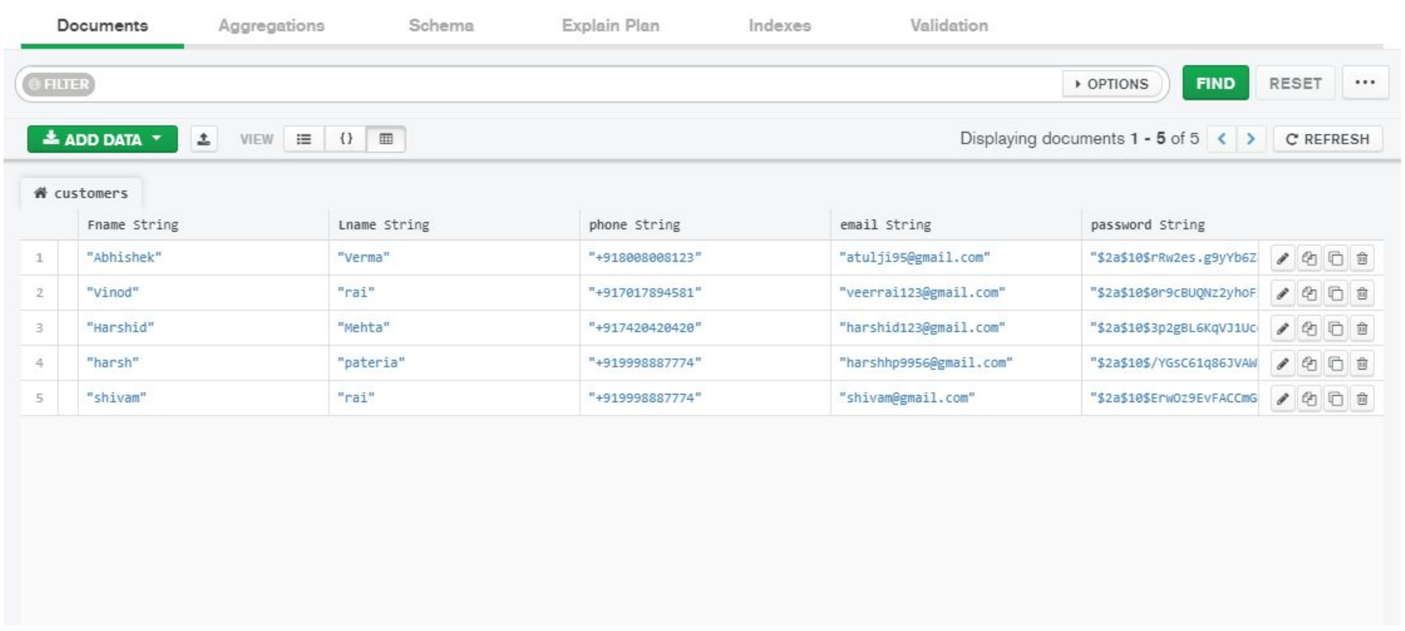
```
The customer with the given mail ID was not found.
```

3. Handling GET Request Errors

Handling error when user passes non-existing mail ID in HTTP request URL

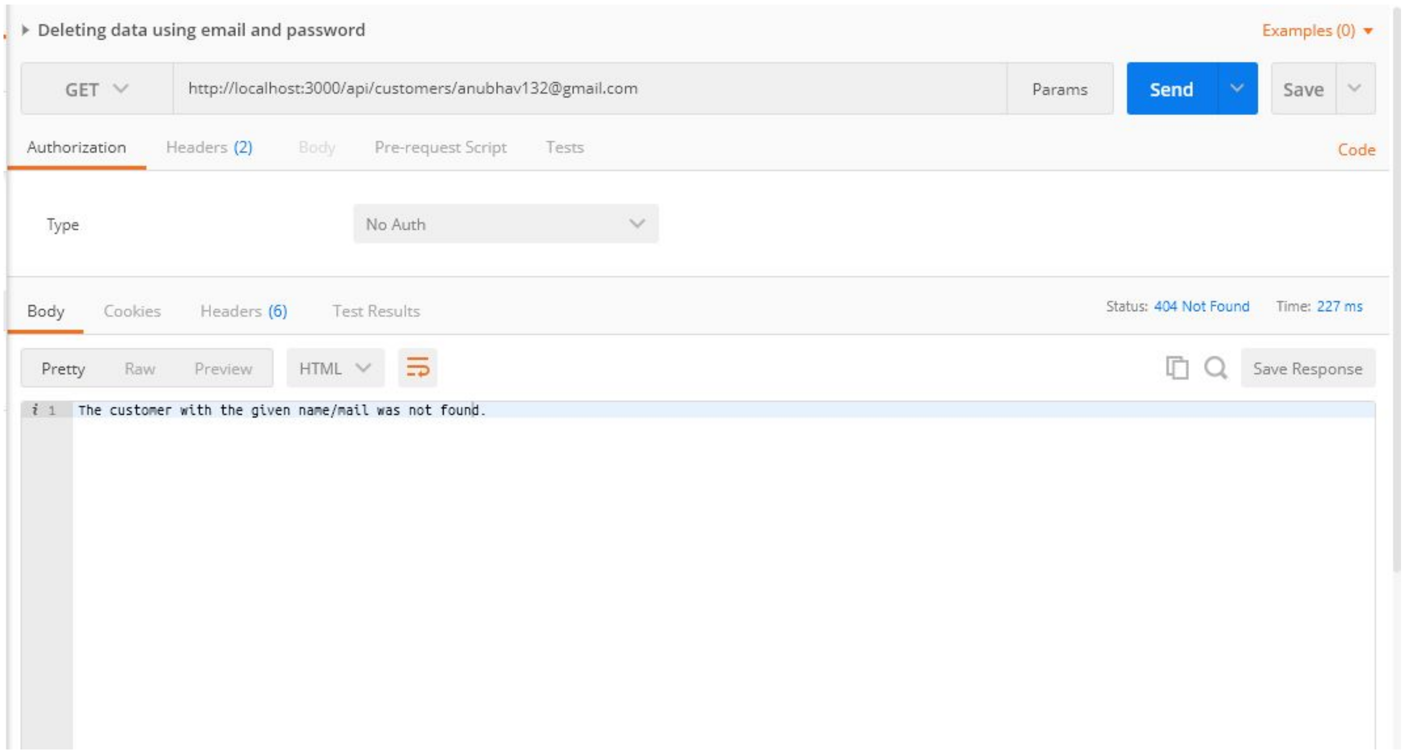
We will try to fetch a contact while passing a mail Id = anubhav132@gmail.com of which contact doesn't exist in the database.

Screenshot of database before Invalid GET Operation



	Fname String	Lname String	phone String	email String	password String
1	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	"$2a$10$rRw2es.g9yYb6Z
2	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	"$2a$10$0r9cBUQNz2yhoF
3	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	"$2a$10$3p2gBL6KqVJ1Uc
4	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	"$2a$10$/YGsC61q86JVAW
5	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	"$2a$10$ErwOz9EvFACCmG

Screenshot of postman after Invalid GET Operation



GET http://localhost:3000/api/customers/anubhav132@gmail.com

Type: No Auth

Status: 404 Not Found Time: 227 ms

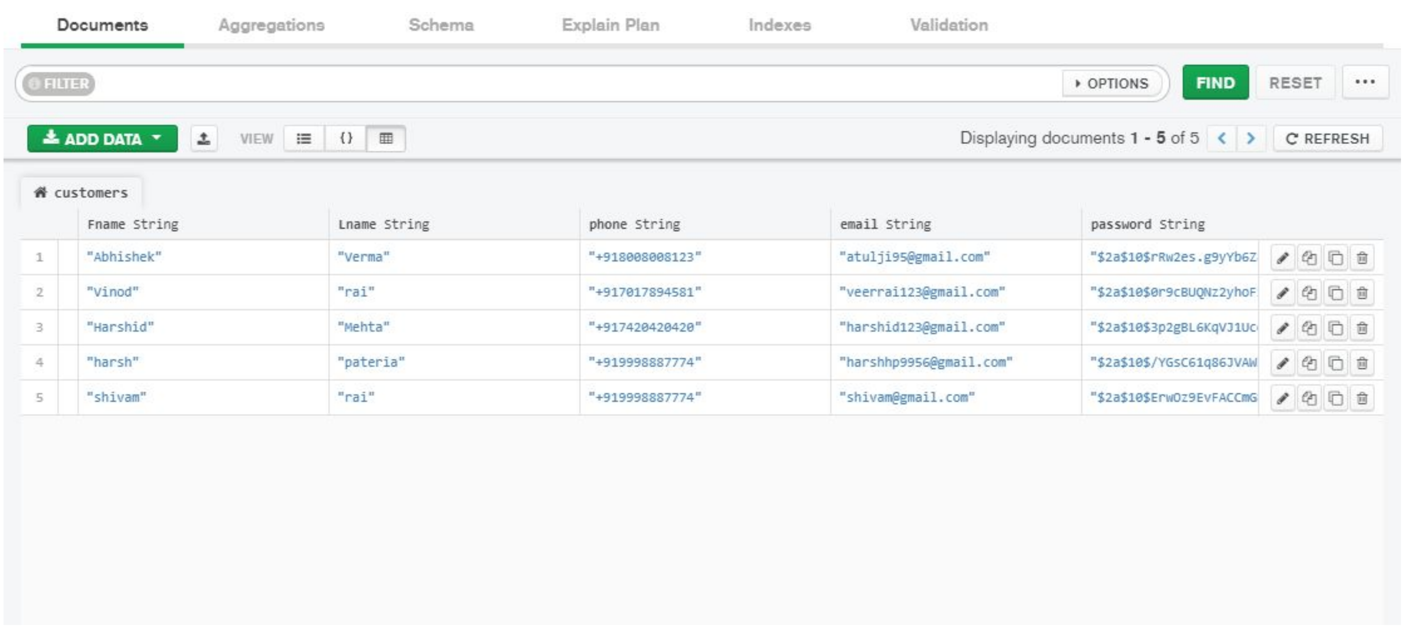
```
The customer with the given name/mail was not found.
```

4. Handling DELETE Request Errors

A. Handling error when user passes non-existing mail ID in HTTP request body

We will try to delete a contact while passing a mail Id = anubhav132@gmail.com of which contact doesn't exist in the database.

Screenshot of database before Invalid DELETE Operation

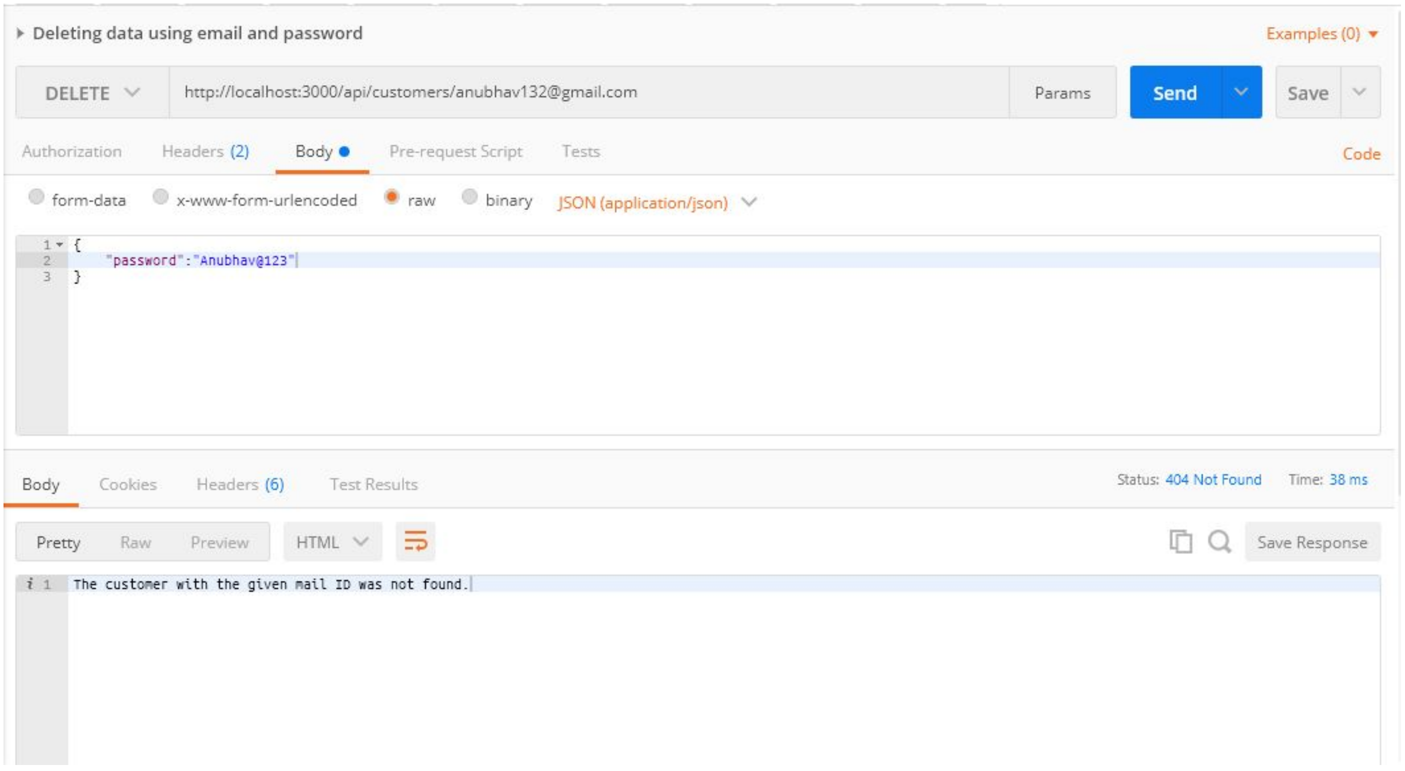


Displaying documents 1 - 5 of 5

customers

	Fname String	Lname String	phone String	email String	password String
1	"Abhishek"	"Verma"	"+918008008123"	"atulji95@gmail.com"	"$2a$10$rRw2es.g9yYb6Z
2	"Vinod"	"rai"	"+917017894581"	"veerrai123@gmail.com"	"$2a$10$0r9cBUQNz2yhoF
3	"Harshid"	"Mehta"	"+917420420420"	"harshid123@gmail.com"	"$2a$10$3p2gBL6KqVJ1Uc
4	"harsh"	"pateria"	"+919998887774"	"harshhp9956@gmail.com"	"$2a$10$/YGsC61q86JVAW
5	"shivam"	"rai"	"+919998887774"	"shivam@gmail.com"	"$2a$10$ErwOz9EvFACCmG

Screenshot of postman after Invalid DELETE Operation



Deleting data using email and password

DELETE http://localhost:3000/api/customers/anubhav132@gmail.com

Body: raw, JSON (application/json)

```
{
    "password":"Anubhav@123"
}
```

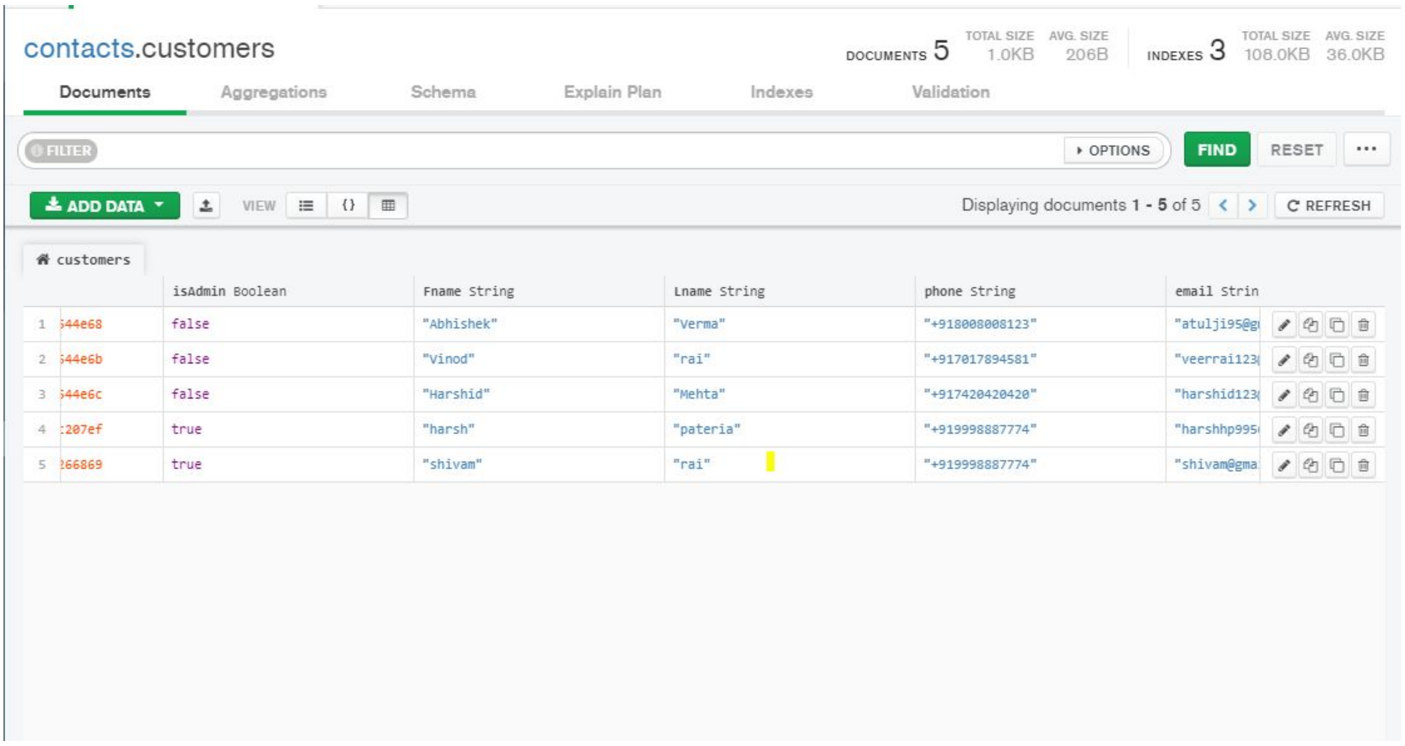
Status: 404 Not Found Time: 38 ms

```
The customer with the given mail ID was not found.
```

B. Handling error when user passes email Id and password of a admin contact.

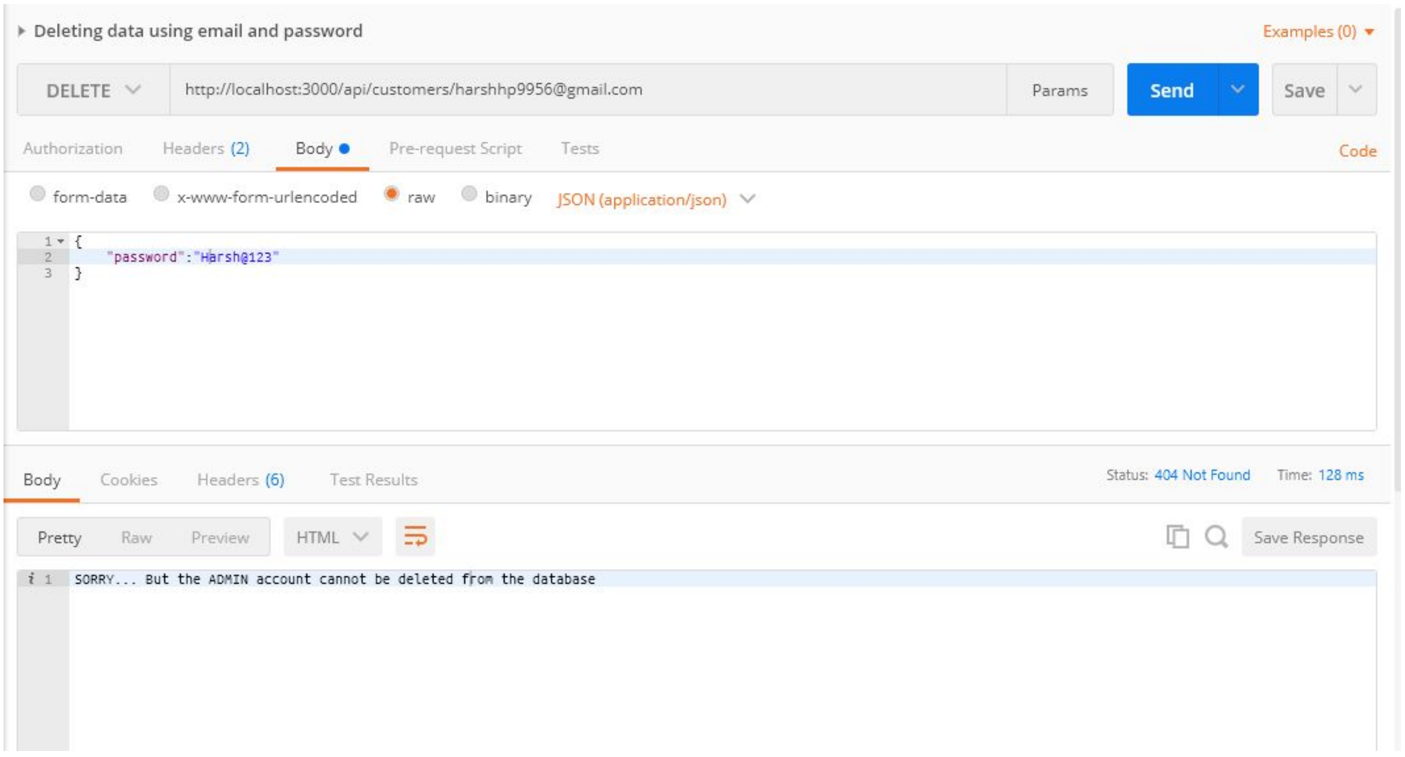
We will try to delete a contact whose mail Id = `harshhp9956@gmail.com` and password=harsh@123 and he is an admin as isAdmin=true. Here it is assumed that a admin account cannot be deleted from database

Screenshot of database before Invalid DELETE Operation



		isAdmin Boolean	Fname String	Lname String	phone String	email Strin
1	;44e68	false	"Abhishek"	"Verma"	"+918008008123"	"atulji95@g
2	;44e6b	false	"Vinod"	"rai"	"+917017894581"	"veerrai123
3	;44e6c	false	"Harshid"	"Mehta"	"+917420420420"	"harshid123
4	:207ef	true	"harsh"	"pateria"	"+919998887774"	"harshhp995
5	:66869	true	"shivam"	"rai"	"+919998887774"	"shivam@gma

Screenshot of postman after Invalid DELETE Operation



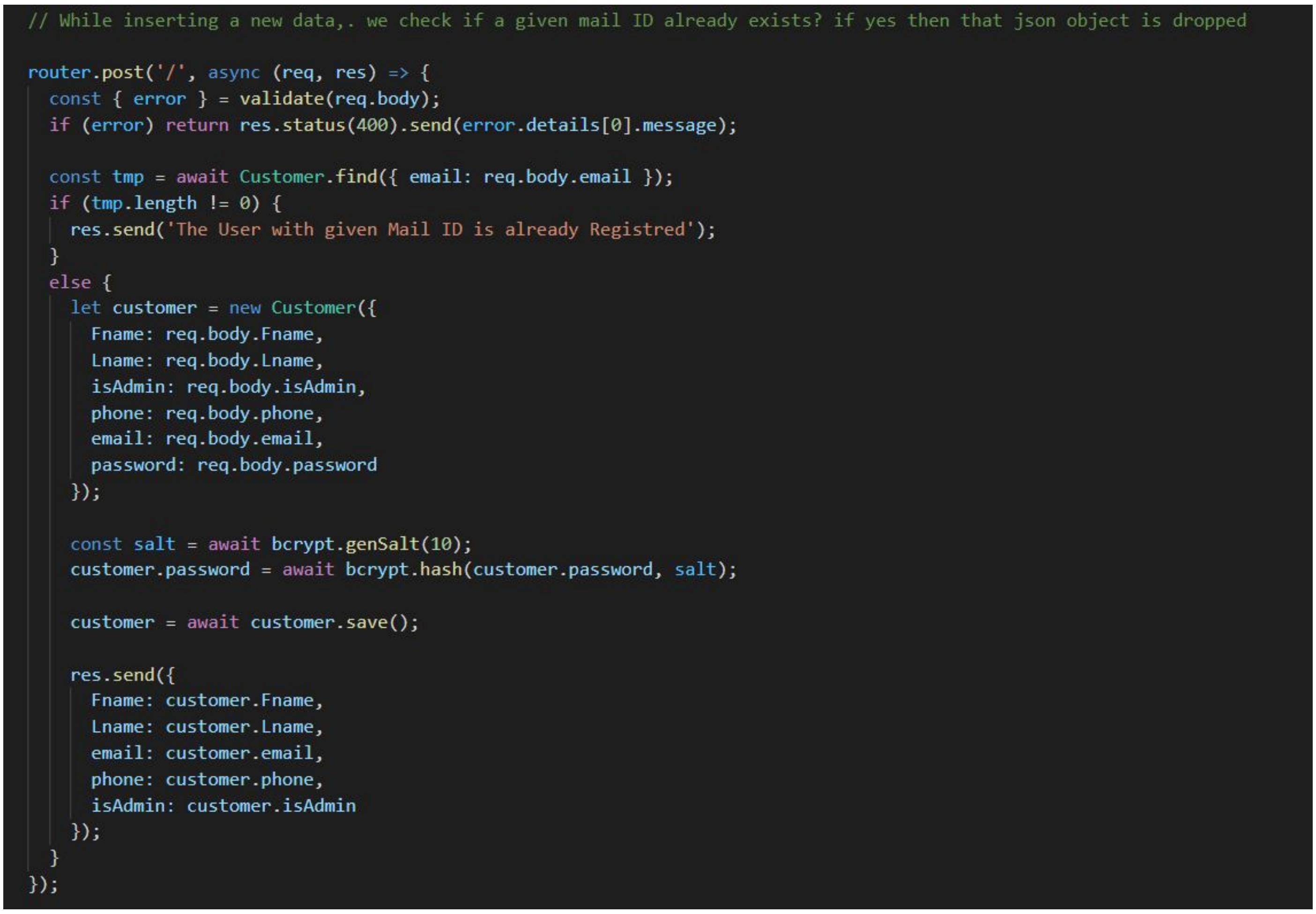
```
http://localhost:3000/api/customers/harshhp9956@gmail.com
```

```
{
    "password":"Harsh@123"
}
```

```
SORRY... But the ADMIN account cannot be deleted from the database
```

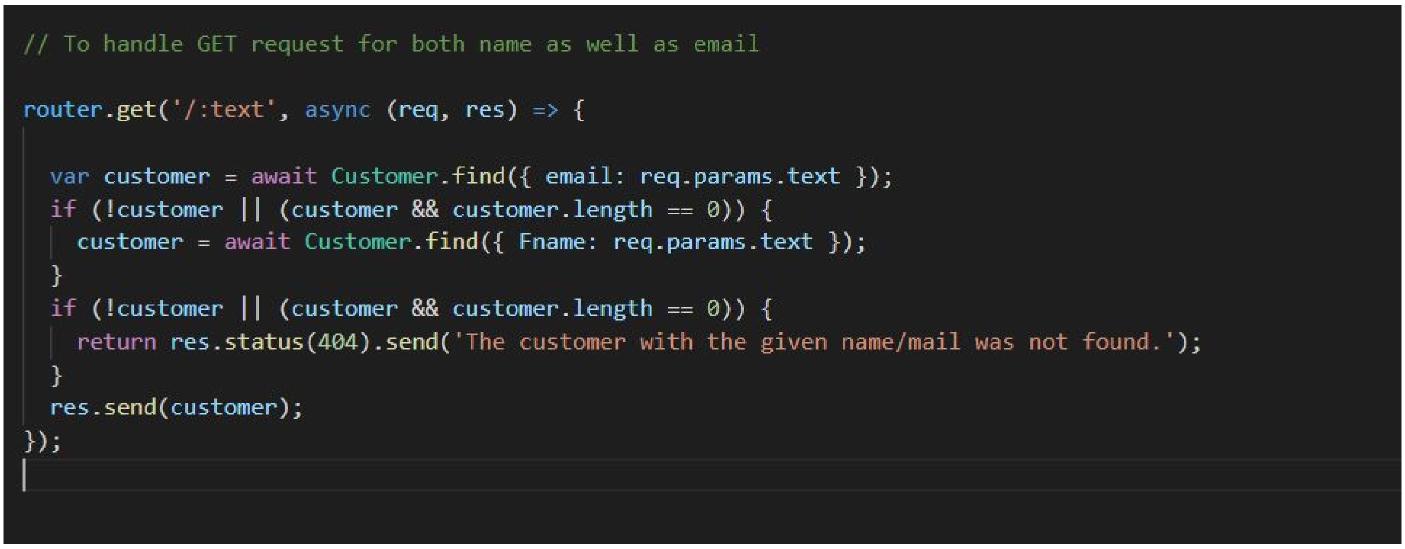
Codes of various route handlers

1. POST



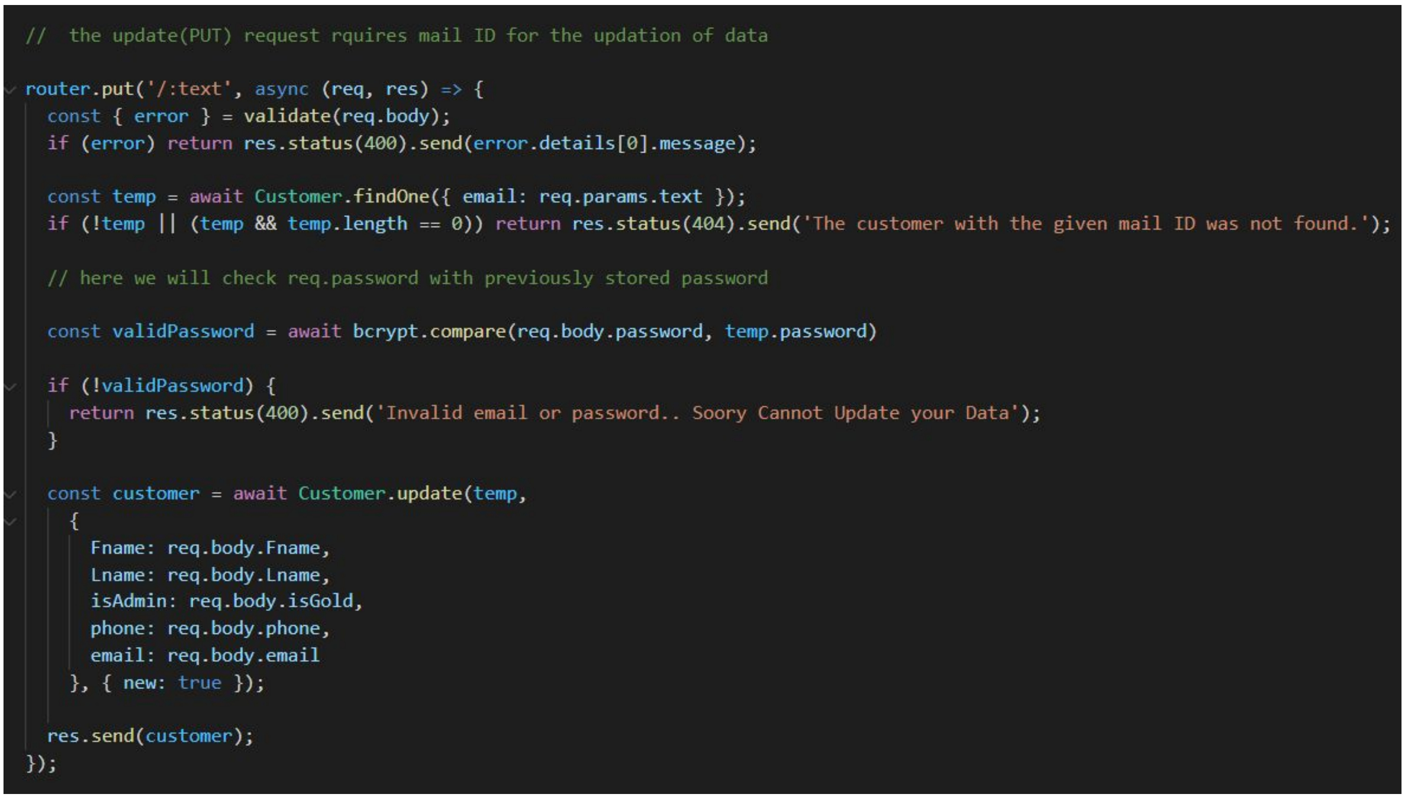
```
// While inserting a new data,. we check if a given mail ID already exists? if yes then that json object is dropped

router.post('/', async (req, res) => {
  const { error } = validate(req.body);
  if (error) return res.status(400).send(error.details[0].message);

  const tmp = await Customer.find({ email: req.body.email });
  if (tmp.length != 0) {
    res.send('The User with given Mail ID is already Registred');
  }
  else {
    let customer = new Customer({
      Fname: req.body.Fname,
      Lname: req.body.Lname,
      isAdmin: req.body.isAdmin,
      phone: req.body.phone,
      email: req.body.email,
      password: req.body.password
    });

    const salt = await bcrypt.genSalt(10);
    customer.password = await bcrypt.hash(customer.password, salt);

    customer = await customer.save();

    res.send({
      Fname: customer.Fname,
      Lname: customer.Lname,
      email: customer.email,
      phone: customer.phone,
      isAdmin: customer.isAdmin
    });
  }
});
```

2. GET



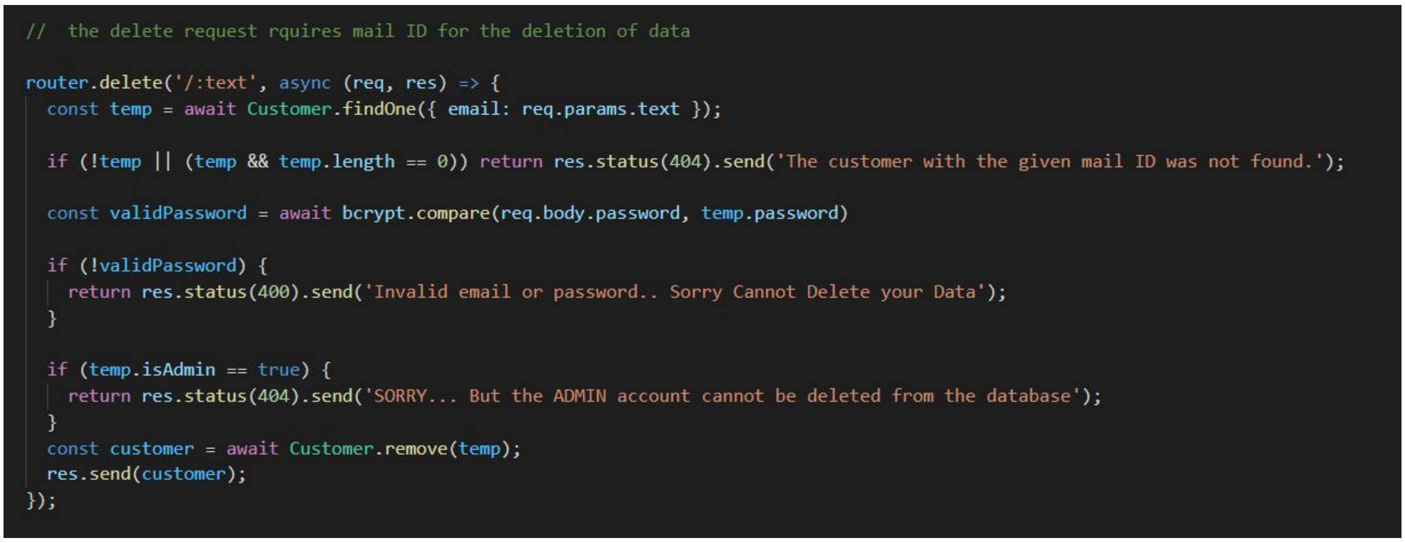
```
// To handle GET request for both name as well as email

router.get('/:text', async (req, res) => {

  var customer = await Customer.find({ email: req.params.text });
  if (!customer || (customer && customer.length == 0)) {
    customer = await Customer.find({ Fname: req.params.text });
  }
  if (!customer || (customer && customer.length == 0)) {
    return res.status(404).send('The customer with the given name/mail was not found.');
  }
  res.send(customer);
});
```

3. PUT



```javascript
//  the update(PUT) request rquires mail ID for the updation of data

router.put('/:text', async (req, res) => {
  const { error } = validate(req.body);
  if (error) return res.status(400).send(error.details[0].message);

  const temp = await Customer.findOne({ email: req.params.text });
  if (!temp || (temp && temp.length == 0)) return res.status(404).send('The customer with the given mail ID was not found.');

  // here we will check req.password with previously stored password

  const validPassword = await bcrypt.compare(req.body.password, temp.password)

  if (!validPassword) {
    return res.status(400).send('Invalid email or password.. Soory Cannot Update your Data');
  }

  const customer = await Customer.update(temp,
    {
      Fname: req.body.Fname,
      Lname: req.body.Lname,
      isAdmin: req.body.isGold,
      phone: req.body.phone,
      email: req.body.email
    }, { new: true });

  res.send(customer);
});
```

4. DELETE



```javascript
//  the delete request rquires mail ID for the deletion of data

router.delete('/:text', async (req, res) => {
  const temp = await Customer.findOne({ email: req.params.text });

  if (!temp || (temp && temp.length == 0)) return res.status(404).send('The customer with the given mail ID was not found.');

  const validPassword = await bcrypt.compare(req.body.password, temp.password)

  if (!validPassword) {
    return res.status(400).send('Invalid email or password.. Sorry Cannot Delete your Data');
  }

  if (temp.isAdmin == true) {
    return res.status(404).send('SORRY... But the ADMIN account cannot be deleted from the database');
  }
  const customer = await Customer.remove(temp);
  res.send(customer);
});
```