REST API Part – 2

We have two routes properly working. Now we want to update. This has to be a put request. But before that let us create resource. We have 10 users in our users.json file. We would like to create another entry in this list from the front-end. The method should be post, the URL is same but the method itself is get request.

//Create

We want to add the object to the users.json file. So, for now, we would put some data and we'll go back to insomnia and will put our URL so that it will give us the list of all the data in it.

In insomnia, we will get the list of all the users.

Create a post request to users

In post we need to send some data as body object. There are different formats but we will be using json. The idea is to receive entire data in the new user object. So, we will copy one user details and put it in the json file of insomnia. We can make few changes as well.

We are uniquely identifying each user by using the userid. If we are adding the 11th user, then we need to have property of 11 and then the user object. The server is serving at once 10 clients. My front end does not know how many users and user id needs to be added. We never generate id and send it to the frontend. It should always be done by the back end.

This data is posted to the server. We will use the json middleware to read the post requests coming to the server.

In the post request,

The data is coming inside the server. We need to generate a unique id and add this data to that id. We will get the users key and it will give us userid. Then we will find the maximum of the numbers, but before that it is a string, so we need to convert that into a number. It is coming as a string, so if we do the math, then it wil depend on the length of the string. We will have to say,

```
app.post('/users/:userId', (req, res) => {
    let keys = Object.keys(users)
    keys = keys.map(k => Number(k))
    console.log(keys)
    res.json({success: true})
})
```

The numbers represent the ids of individual users.

```
app.post('/users/:userId', (req, res) => {
    let keys = Object.keys(users)
    keys = keys.map(k => Number(k))
    const newId = Math.max(...keys) + 1
    res.json({success: true})
})
```

As the new id should increment every time, we add a new user we can put +1. We have generated a new id. All we need to do is,

```
app.post('/users/:userId', (req, res) => {
    let keys = Object.keys(users)
    keys = keys.map(k => Number(k))
    const newId = Math.max(...keys) + 1
    user[newId] = req.body
    res.json({success: true})
})
```

We are not updating the file currently, as everything is happening inside the memory.

Post route is supposed to be creating everything.

We have created something and now we need to update something.

//Update

Whenever we think about updating the basic request is put. In some way, we need to specify the id and then send the fresh data so that I can update it. We want to update the user 1. We will create another request.

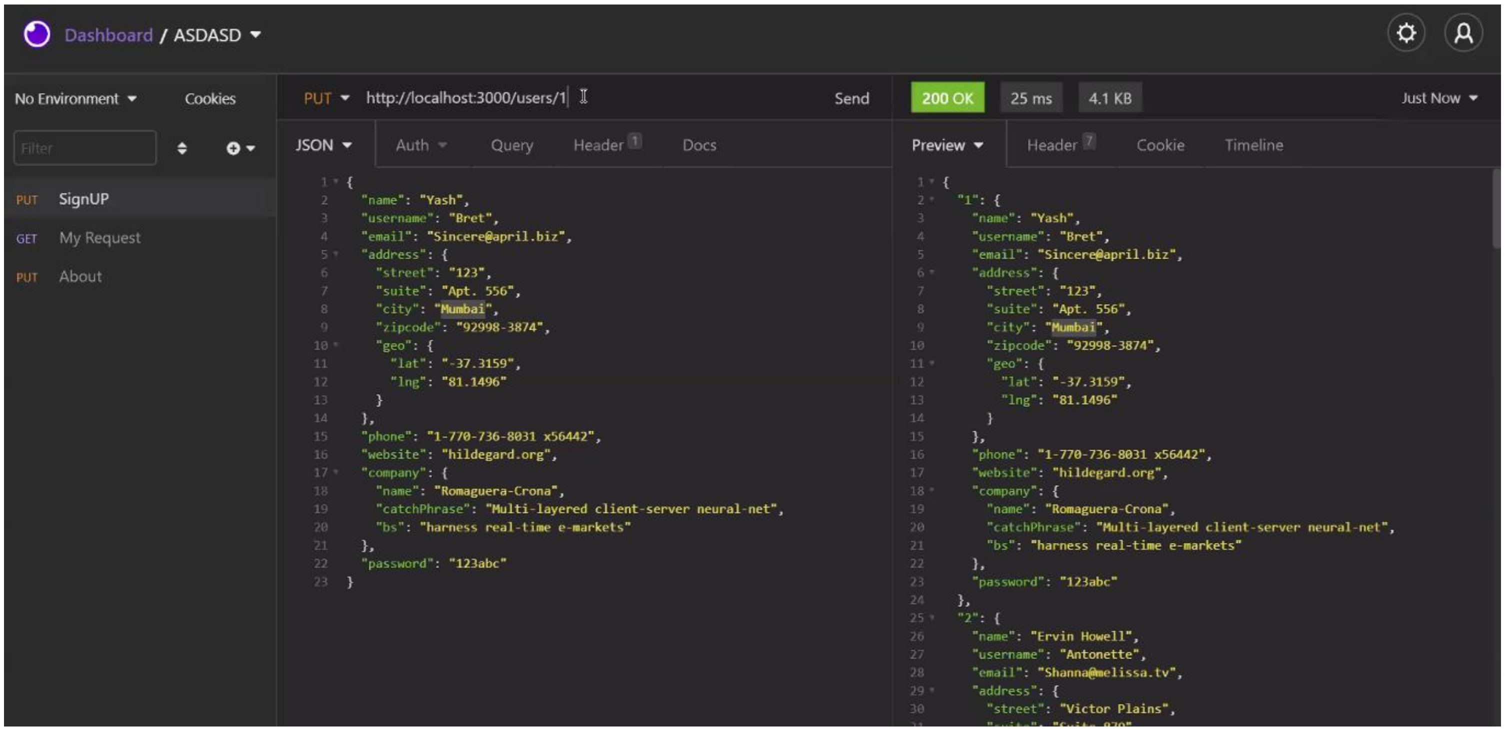
We should be getting all the values that have been changed as well as the values that are not changed. Whatever earlier data was remains as it is and changes would be reflected. Based on this logic, we will try to implement it.

```
app.put('/users/:userId', (req, res) => {
    const {userId} = req.params
    users[userId] = req.body

    res.json(users)
})
```

We are updating which is already existing. POST is used for new entry and PUT is to make changes to the entries.

Img1



PUT http://localhost:3000/users/1

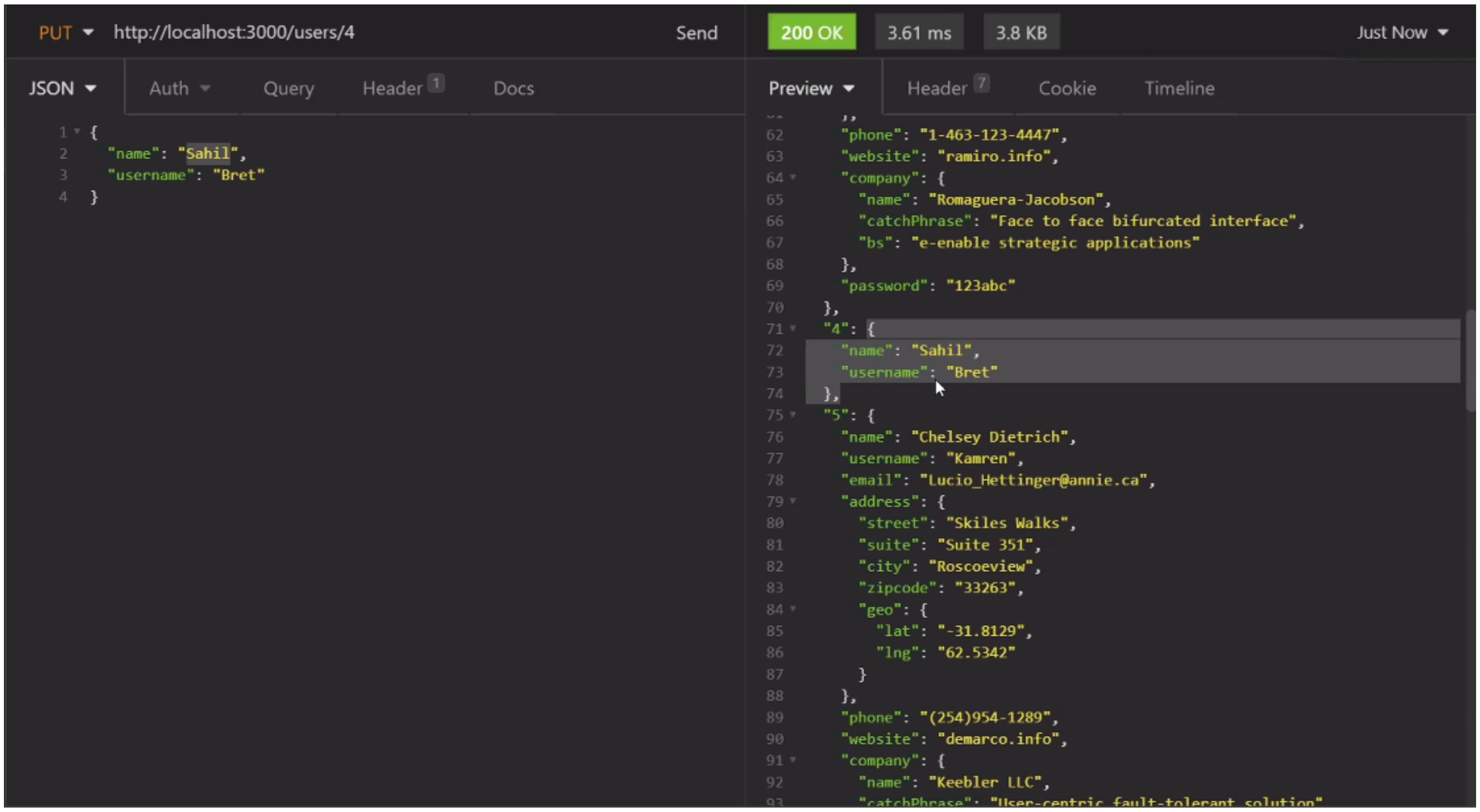
```
{
  "name": "Yash",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "123",
    "suite": "Apt. 556",
    "city": "Mumbai",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  },
  "password": "123abc"
}
```

200 OK 25 ms 4.1 KB

```
{
  "1": {
    "name": "Yash",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "123",
      "suite": "Apt. 556",
      "city": "Mumbai",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    },
    "password": "123abc"
  },
  "2": {
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
```

We have to send the entire object or else,

Img2



http://localhost:3000/users/4

```
{
  "name": "Sahil",
  "username": "Bret"
}
```

```
      },
      "phone": "1-463-123-4447",
      "website": "ramiro.info",
      "company": {
        "name": "Romaguera-Jacobson",
        "catchPhrase": "Face to face bifurcated interface",
        "bs": "e-enable strategic applications"
      },
      "password": "123abc"
    },
    "4": {
      "name": "Sahil",
      "username": "Bret"
    },
    "5": {
      "name": "Chelsey Dietrich",
      "username": "Kamren",
      "email": "Lucio_Hettinger@annie.ca",
      "address": {
        "street": "Skiles Walks",
        "suite": "Suite 351",
        "city": "Roscoeview",
        "zipcode": "33263",
        "geo": {
          "lat": "-31.8129",
          "lng": "62.5342"
        }
      },
      "phone": "(254)954-1289",
      "website": "demarco.info",
      "company": {
        "name": "Keebler LLC",
        "catchPhrase": "User-centric fault-tolerant solution"
```

So, we need to provide the entire object. But when we put in a patch request, we only send the data that needs to be updated and it is backend who will make changes.

We are following an idealistic approach. We have not done any checks in this right now. We have userid from 1 to 10, but if someone gives more than 10, then we need to put a check if the number mentioned is between 1 to 10.

//Delete

We do not need the entire body to delete a user data. We just have to specify the userID to get it deleted. So, all we need to do is,

```
app.delete('/users/:userId', (req, res) => {
    delete users[req.params.userId]
    res.json(users)
})
```

File Handling: