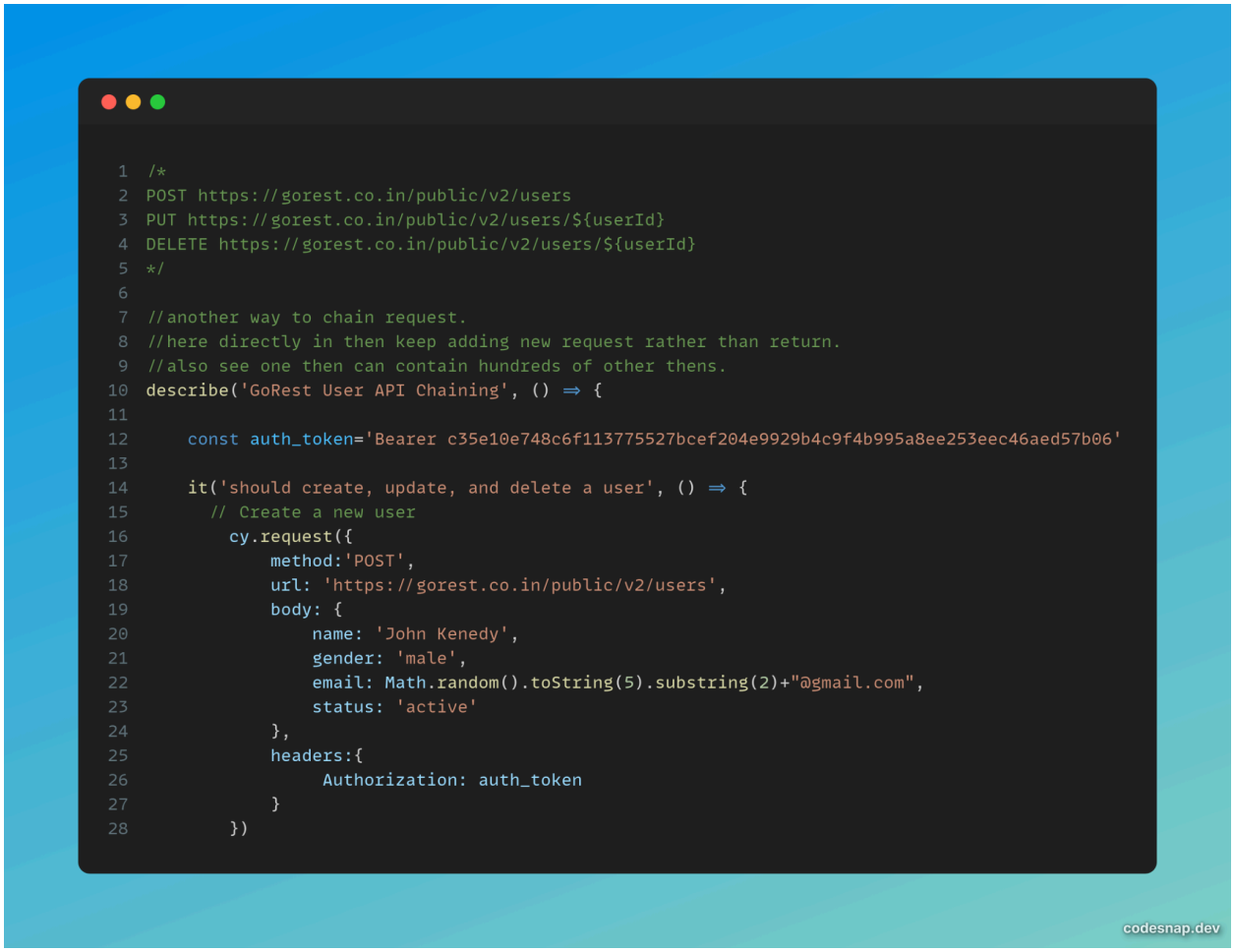
```
describe('API chaining in Cypress', () => {
    it('should make API requests and chain them together', () => {
       cy.request(
        {
           method: 'GET',
           url: 'https://jsonplaceholder.typicode.com/posts'
        })
        .then((response) => {
          expect(response.status).to.eq(200)
          //extract the post id and store in variable.
          const postid = response.body[0].id
          //use return for request chaining.
          //this is the first then statement.
          return postid
         })
         //one request can have multiple then blocks.
         //use the postid in another response as query parameter.
         //then can also have any damn thing inside it,
         //including request.
        .then((postid) => {
          cy.request({
            method:'GET',
            url: `https://jsonplaceholder.typicode.com/comments?postId=${postid}`
          })
          .then((response)=>{
            expect(response.status).to.eq(200)
            expect(response.body).to.have.length(5)
          })
        })
    })


  })
```

codesnap.dev

Another way for chaining using nested thens –



```
/*
POST https://gorest.co.in/public/v2/users
PUT https://gorest.co.in/public/v2/users/${userId}
DELETE https://gorest.co.in/public/v2/users/${userId}
*/

//another way to chain request.
//here directly in then keep adding new request rather than return.
//also see one then can contain hundreds of other thens.
describe('GoRest User API Chaining', () => {

    const auth_token='Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'

    it('should create, update, and delete a user', () => {
      // Create a new user
        cy.request({
            method:'POST',
            url: 'https://gorest.co.in/public/v2/users',
            body: {
                name: 'John Kenedy',
                gender: 'male',
                email: Math.random().toString(5).substring(2)+"@gmail.com",
                status: 'active'
            },
            headers:{
                 Authorization: auth_token
            }
        })
```


Continue from line 28 –
Add from line 29 after the closure of request.

Trying to paste all in one and seeing –

```
/*
POST https://gorest.co.in/public/v2/users
PUT https://gorest.co.in/public/v2/users/${userId}
DELETE https://gorest.co.in/public/v2/users/${userId}
*/

//another way to chain request.
//here directly in then keep adding new request rather than return.
//also see one then can contain hundreds of other thens.
describe('GoRest User API Chaining', () => {

    const auth_token=
'Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'

    it('should create, update, and delete a user', () => {
      // Create a new user
        cy.request({
            method:'POST',
            url: 'https://gorest.co.in/public/v2/users',
            body: {
                name: 'John Kenedy',
                gender: 'male',
                email: Math.random().toString(5).substring(2)+"@gmail.com",
                status: 'active'
            },
            headers:{
                 Authorization: auth_token
            }
        })
        .then((response) => {
            expect(response.status).to.equal(201)
            const userId = response.body.id
            // Update the user's name
            cy.request(
                {
                    method: 'PUT',
                    url: `https://gorest.co.in/public/v2/users/${userId}`,
                    body: {
                        name: 'Scott'
                    },
                    headers:{
                        Authorization: auth_token
                    }
                })
               .then((response) => {
                    expect(response.status).to.equal(200)
                    expect(response.body.name).to.equal('Scott')
                    // Delete the user
                    cy.request(
                        {
                            method:'DELETE',
                            url: `https://gorest.co.in/public/v2/users/${userId}`,
                            headers:{
                                Authorization: auth_token
                            }
                        })
                        .then((response) => {
                                expect(response.status).to.equal(204)
                         })
                 })
            })
        })
  })
```

Below is also time pass to see how it looks if we paste all–

```
/*
POST https://gorest.co.in/public/v2/users
PUT https://gorest.co.in/public/v2/users/${userId}
DELETE https://gorest.co.in/public/v2/users/${userId}
*/

//another way to chain request.
//here directly in then keep adding new request rather than return.
//also see one then can contain hundreds of other thens.
describe('GoRest User API Chaining', () => {

    const auth_token=
'Bearer c35e10e748c6f113775527bcef204e9929b4c9f4b995a8ee253eec46aed57b06'

    it('should create, update, and delete a user', () => {
      // Create a new user
        cy.request({
            method:'POST',
            url: 'https://gorest.co.in/public/v2/users',
            body: {
                name: 'John Kenedy',
                gender: 'male',
                email: Math.random().toString(5).substring(2)+"@gmail.com",
                status: 'active'
            },
            headers:{
                 Authorization: auth_token
            }
        })
        .then((response) => {
            expect(response.status).to.equal(201)
            const userId = response.body.id
            // Update the user's name
            cy.request(
                {
                    method: 'PUT',
                    url: `https://gorest.co.in/public/v2/users/${userId}`,
                    body: {
                        name: 'Scott'
                    },
                    headers:{
                        Authorization: auth_token
                    }
                })
                .then((response) => {
                    expect(response.status).to.equal(200)
                    expect(response.body.name).to.equal('Scott')
                    // Delete the user
                    cy.request(
                        {
                            method:'DELETE',
                            url: `https://gorest.co.in/public/v2/users/${userId}`,
                            headers:{
                                Authorization: auth_token
                            }
                        })
                        .then((response) => {
                                expect(response.status).to.equal(204)
                         })
                 })
```