Fetch, Authentication and Database

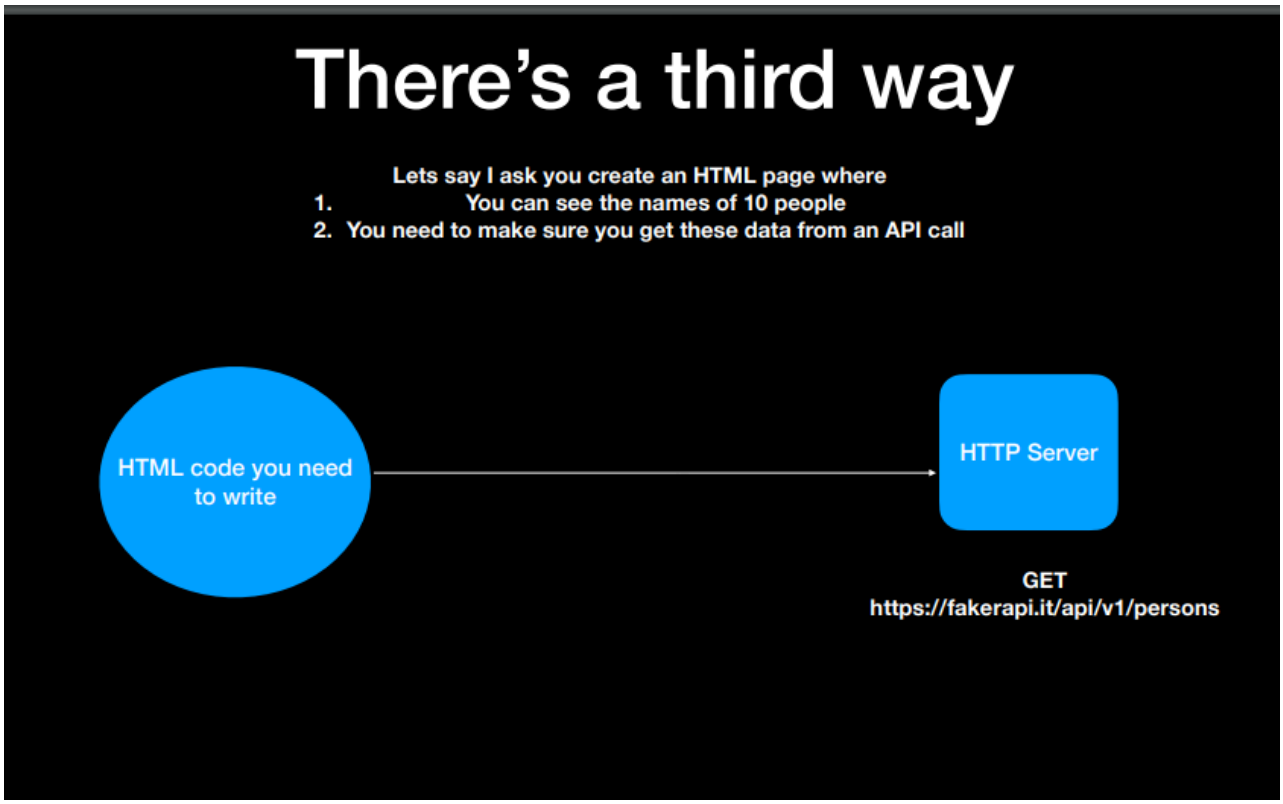
Until now we have make project which can be accessed by others easily.

The fetch API

Till now we have hit the route either by POSTMAN and browser.

How do chatgpt send request etc when a button is clicked.

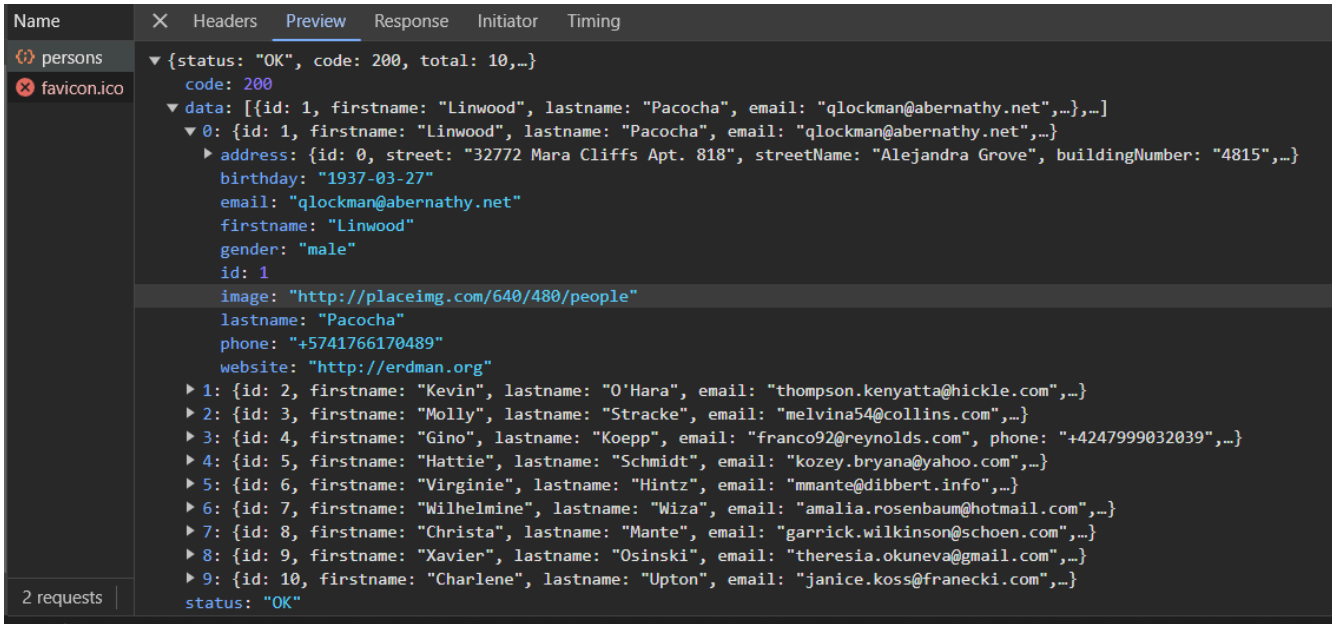
Browser has a feature called fetch



Write some frontend code which hits the endpoint and get some data and put it on screen.

Simple way can be like we simple write ten name and display it.

On going into the https://fakerapi.it/api/v1/persons and then going on network site and then seeing the response we can see that its response contains json object contains name of 10. people.

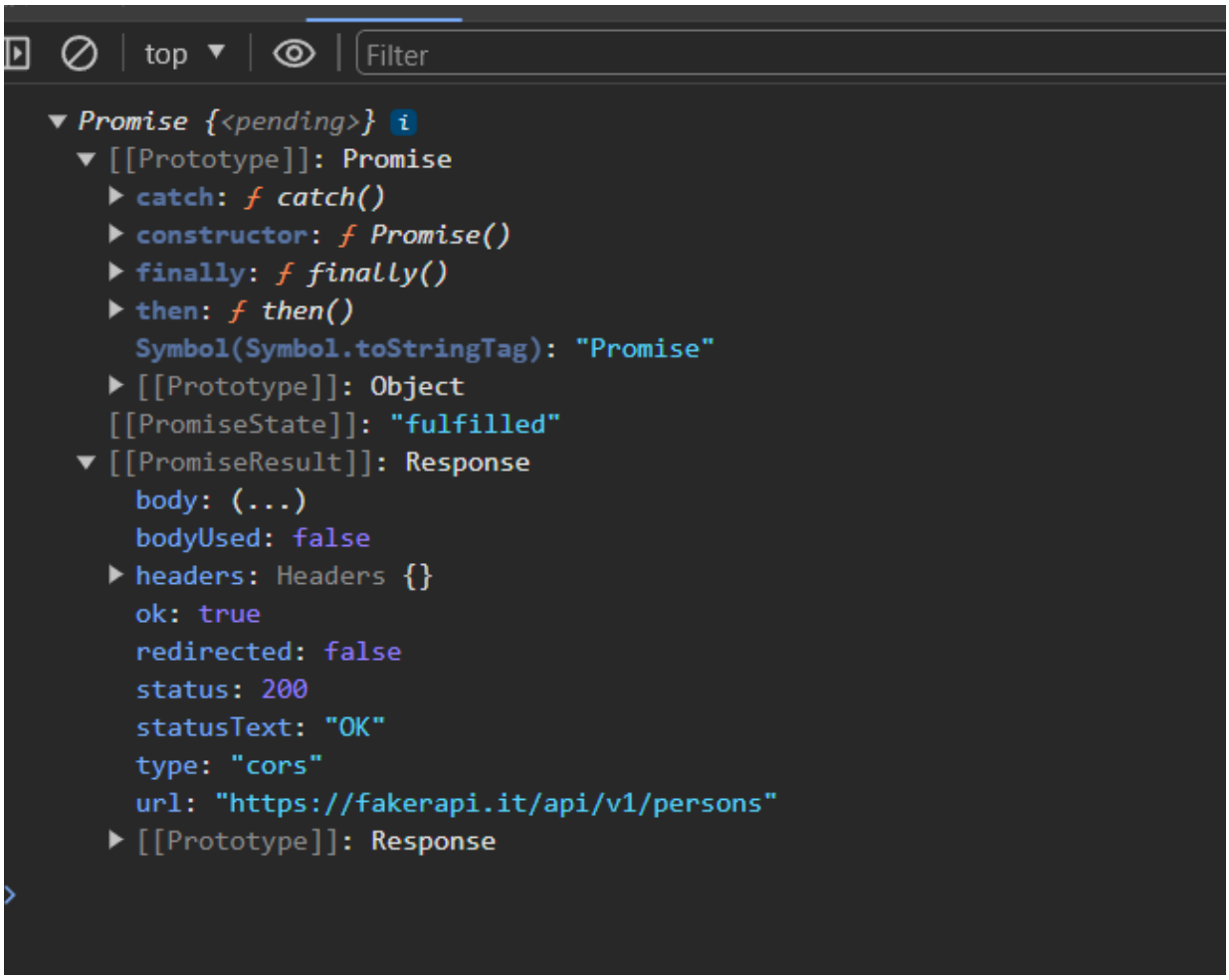


I need to create an HTML page that will hit this api, similar to what happens when we write a prompt in chatgpt.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function getAnimalData(){
            // To check if the function is working or not
            //    alert("You are inside the getAnimalData function")
           const reponse = fetch('https://fakerapi.it/api/v1/persons')
        //by default fetch is a get request
            // but if we want it to be a post request then we can do it
like this
            // fetch('',{
            //     method: 'POST',
            // })
            console.log(reponse)
```

```
        }
    </script>
    <button onclick="getAnimalData()">
        Get Animal Data
    </button>
</body>
</html>
```

Console



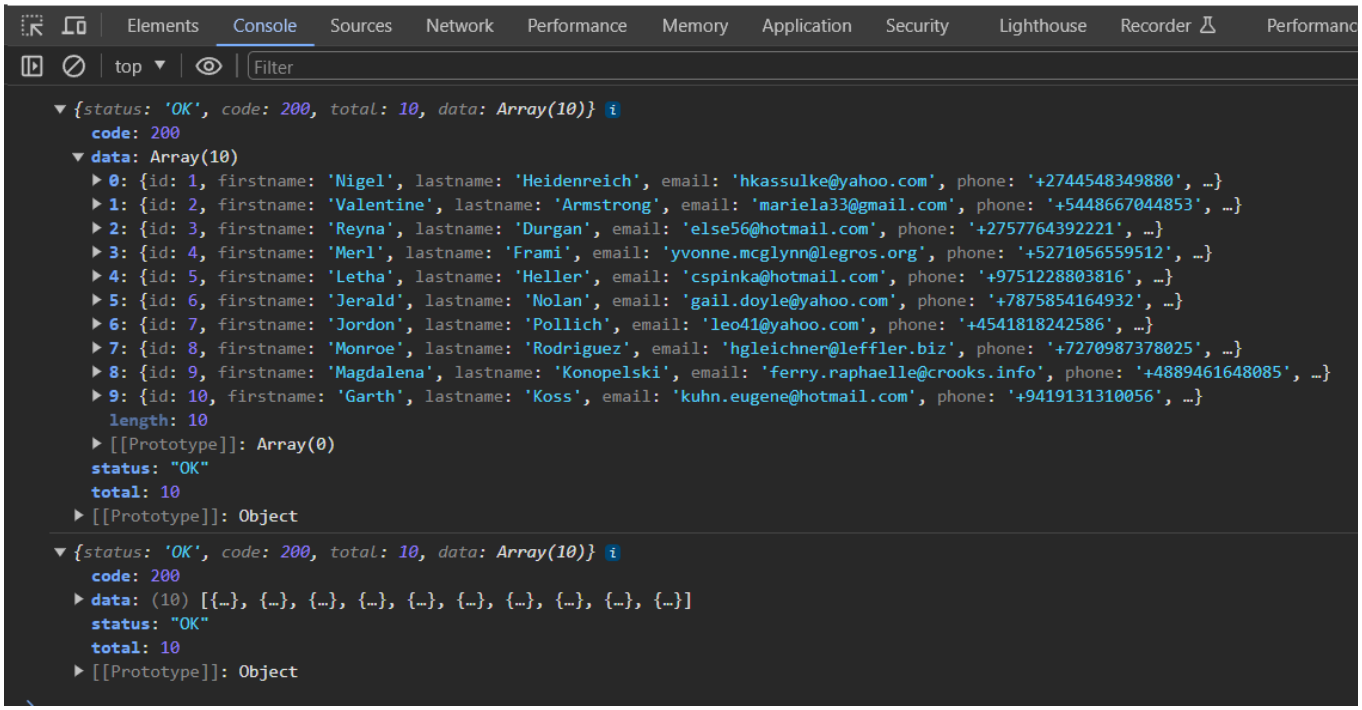
It console.log() promise

So we have to use promise syntax

```
function getAnimalData(){
            fetch('https://fakerapi.it/api/v1/persons')
            .then(function(response){
            //this also is promise
```

```
            response.json()
            .then(function(finalData){
             console.log(finalData)
            })
        })
    }
```

It will console



This is very similar to chat gpt. Now we can say we are frontend developer we havent wrote the code at the backend but we have write the code to hit the backend server and log it .

Another way to access(get the data) is

```
async function getAnimalData(){
            const response = fetch('https://fakerapi.it/api/v1/persons')
            const finalData = await response.json()
            console.log(finalData)
        }
```

To render it we need to know **DOM.**

We don't need to click the button to access the api.

We can also make it such that it refreshes.

Final code

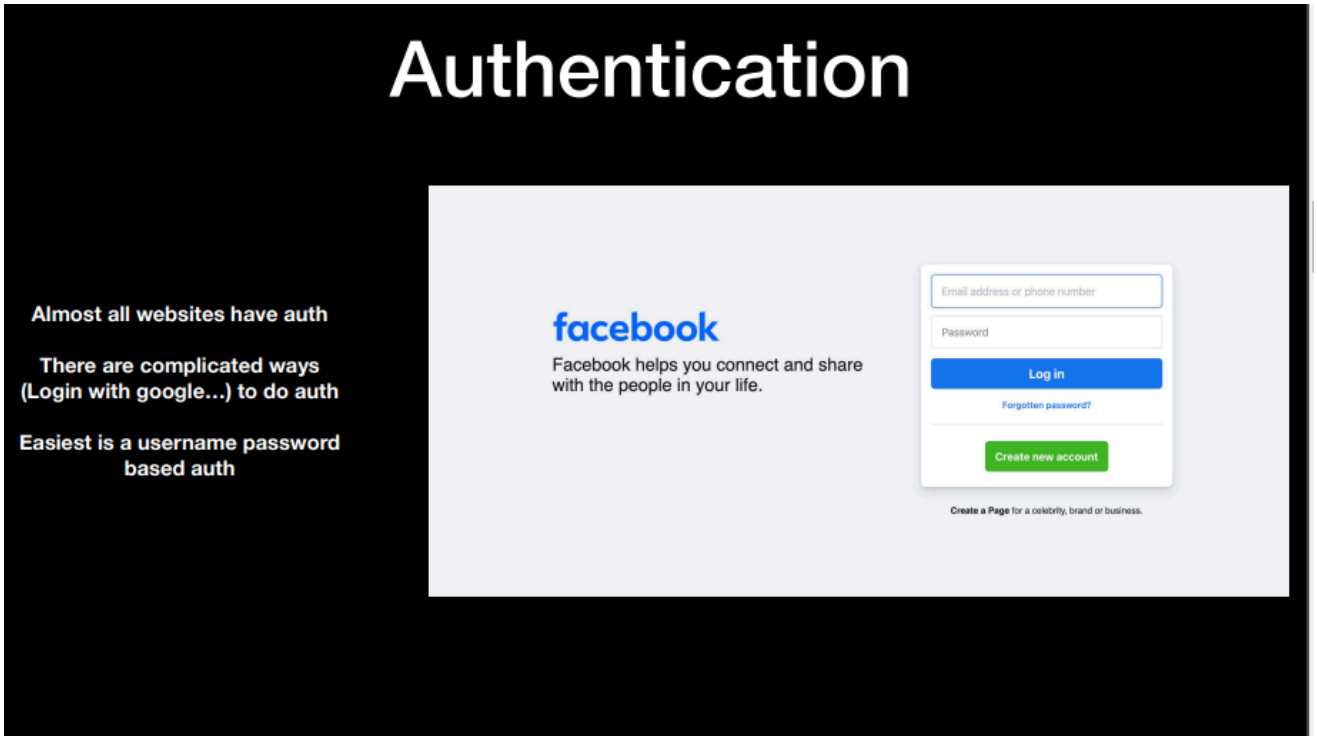
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function getAnimalData(){
            fetch('https://fakerapi.it/api/v1/persons')
            .then(function(response){
            //this also is a promise
               response.json()
               .then(function(finalData){
                console.log(finalData)
               })
            })
        }

        async function getAnimalData(){
            const response = fetch('https://fakerapi.it/api/v1/persons')
            const finalData = await response.json()
            console.log(finalData)
        }
    </script>
    <button onclick="getAnimalData()">
        Get Animal Data
    </button>
</body>
</html>
```

Authentication

Project for today –

Let people sign up to your website

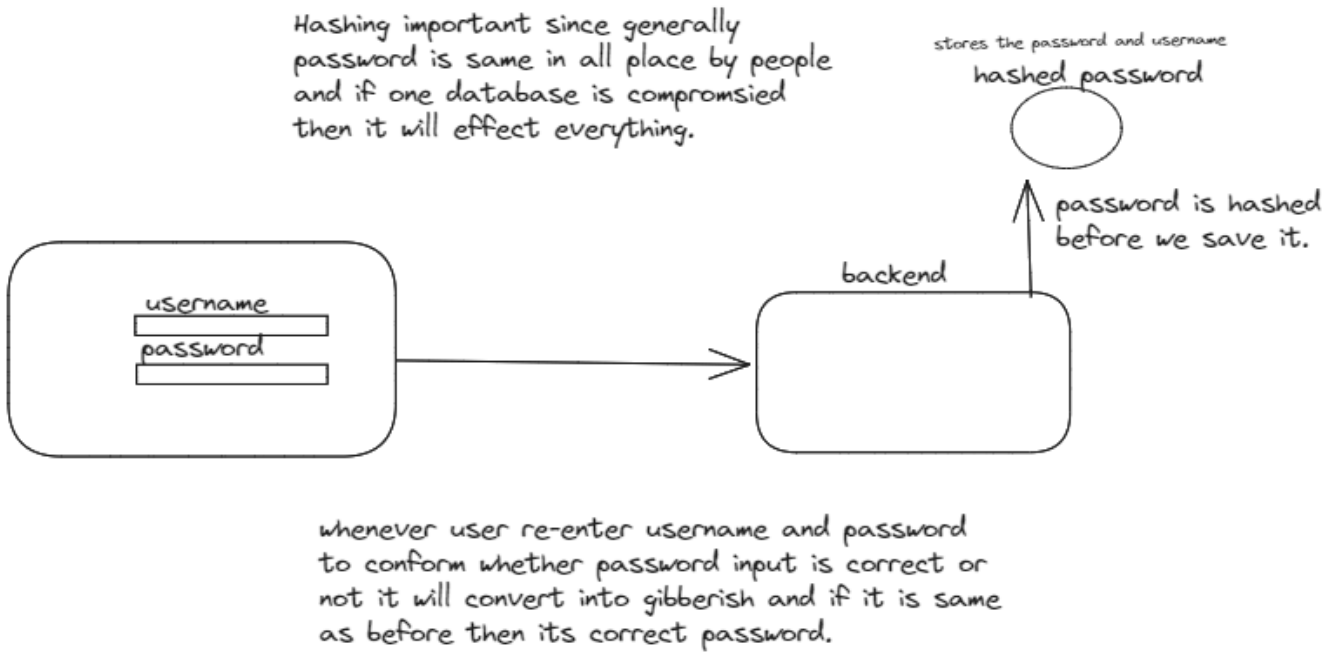
Only allow signed in users to see people (create a dummy people list)



Before we get into authentication . Lets understand some cryptography jargon

1. Hashing
2. Encryption
3. Json web tokens
4. Local storage

Hashing



Hashing is one way.

Hashing is one directional. Given the output no one can find out the input.

Changing the input a lil bit changes the output by lot.

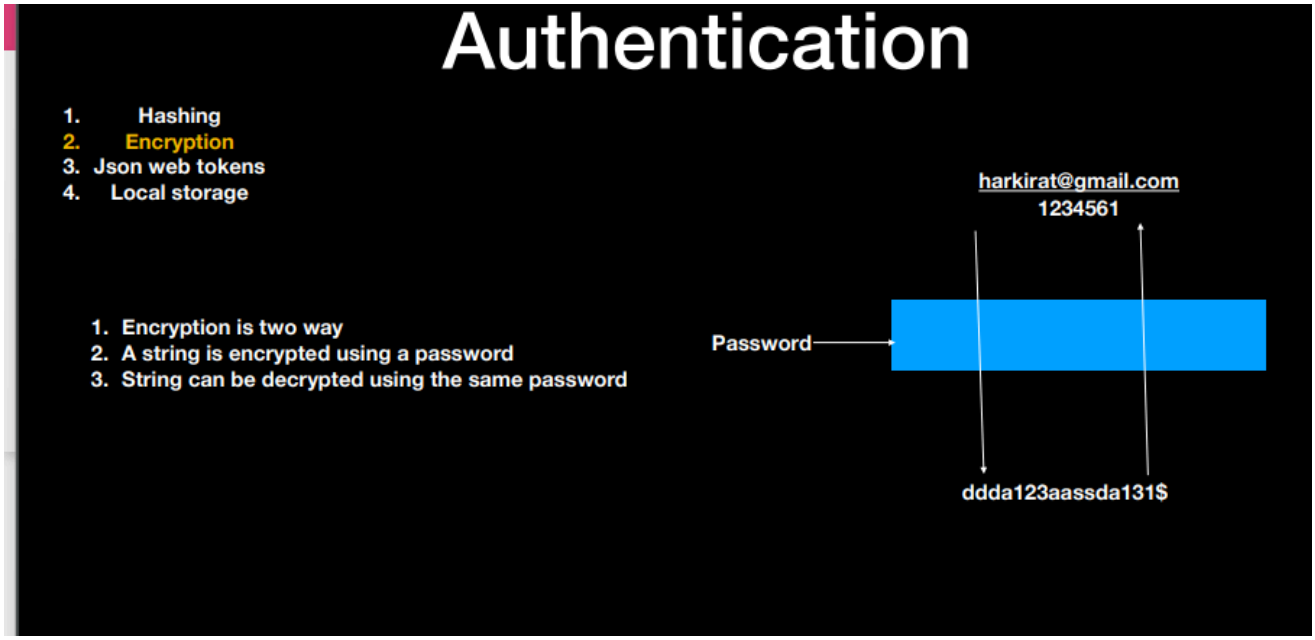
Encryption:

It is two way provided we have a key.

Suppose we upload any video and photos it wont be stored directly in form of mp4 , it will be encrypted and then stored in database and when we have to fetch it we will decrypt it.

Encryption does require password

If someone has password then it can be decrypted.



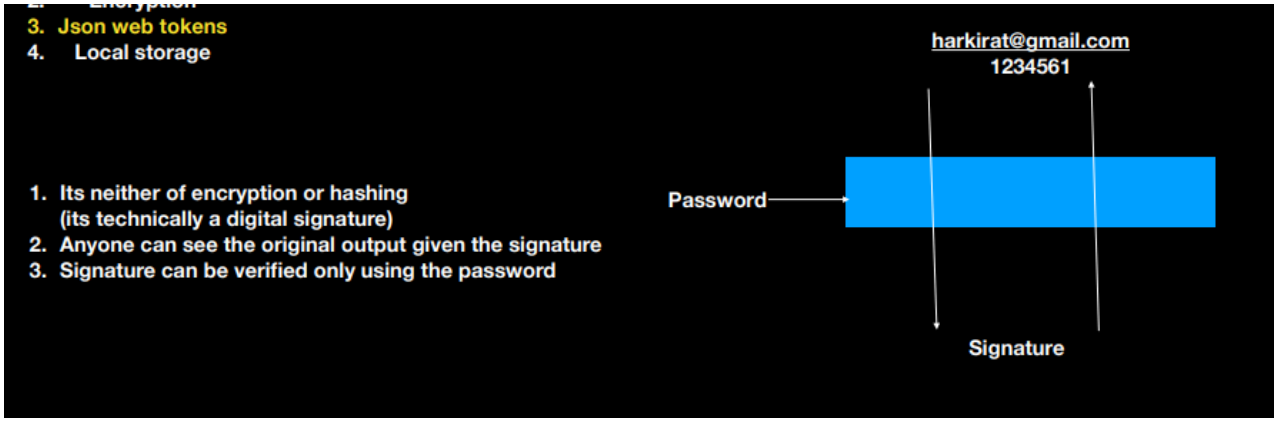
Json web tokens(JWT)

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

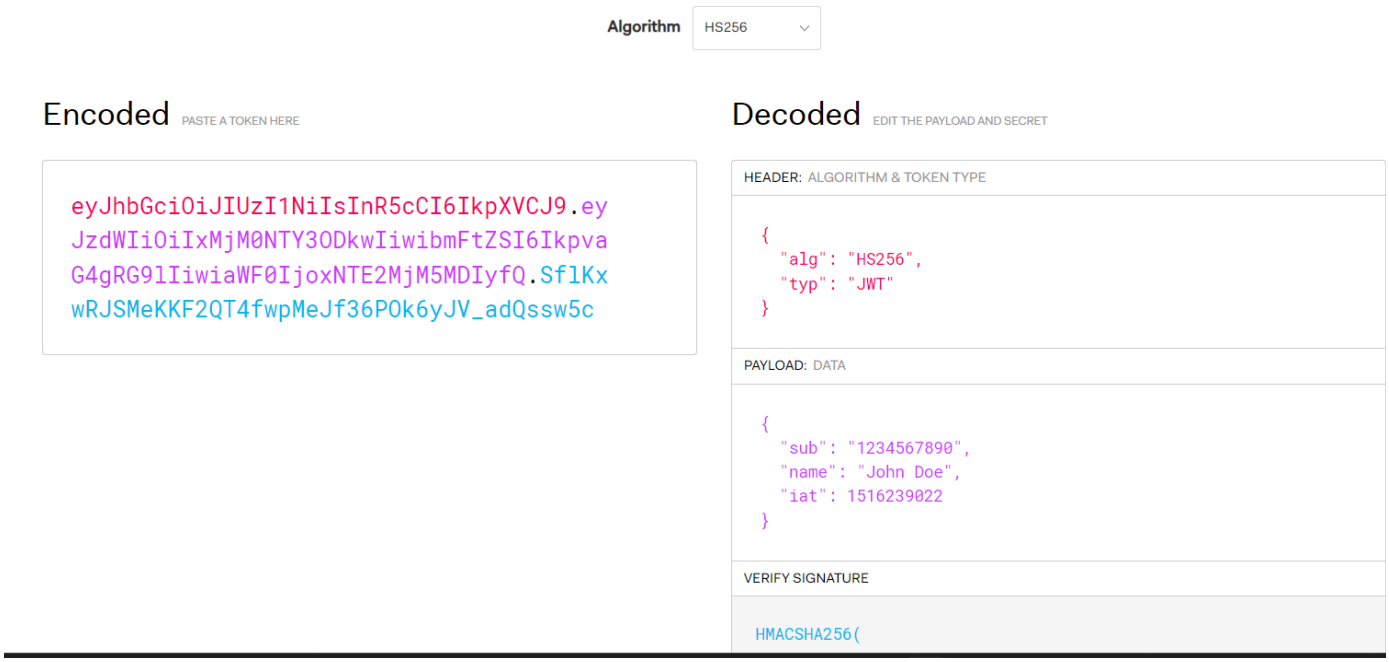
Json : it is some hashing function but it works only for json input
It give long string it has three parts etc
Web
token : the long string it give is called token

We give it input and it give output and if someone has output then it can be accessed.
The very long thing which we see it in the chatgpt headers section of network. Is the JWT

We can decode JWT by going to jet.io
Why will we even do this if someone can look into this can decode it???



To decode we need password



Conversion anyone can do , but verification can be done by chatgpt only.

jwt.verify(__)

// this code only runs when it will be correct.

Local Storage

We get the token when we signed in. from backend

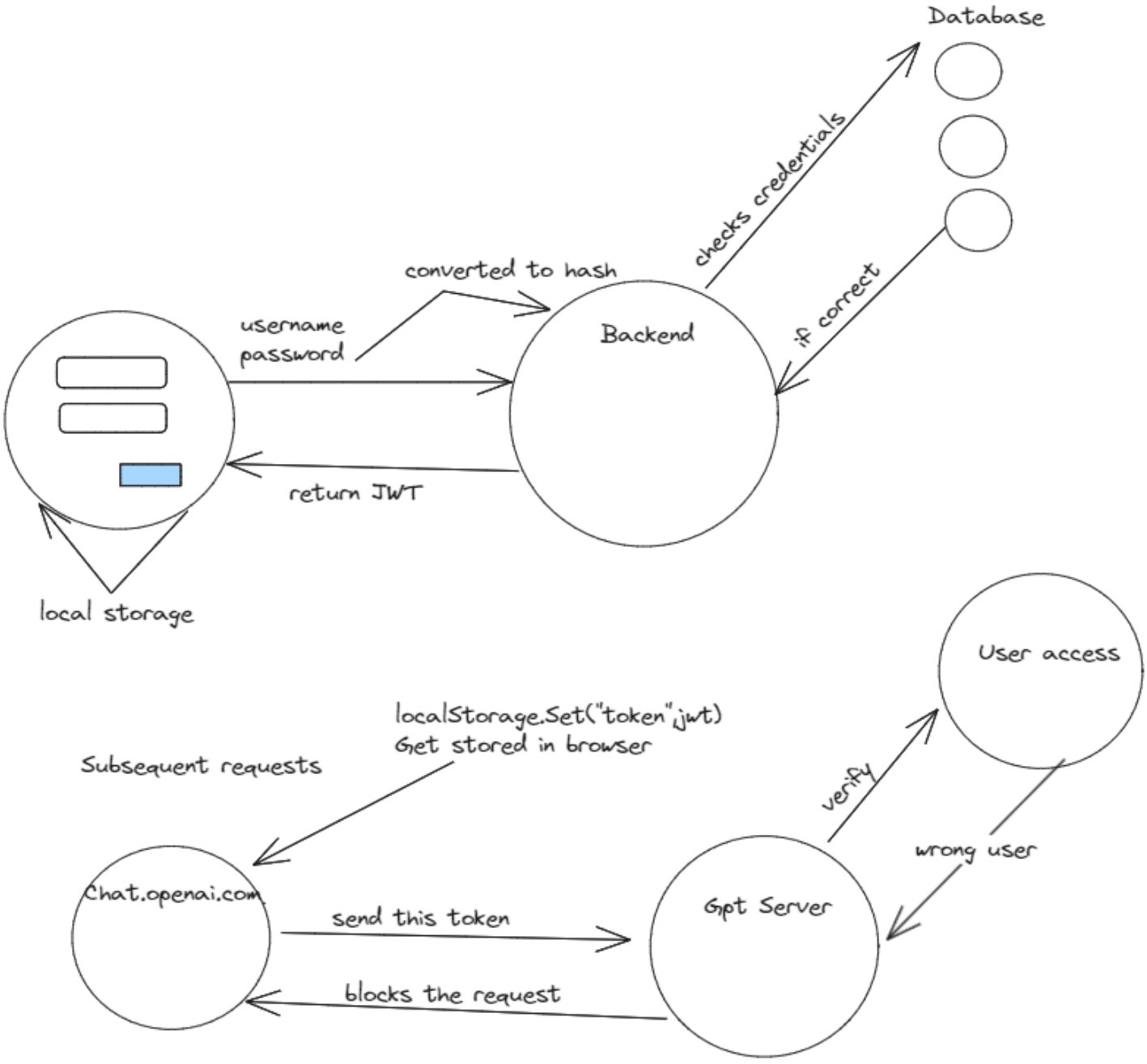
How is token getting rellaled back in every request.

Storage we can see Local storage we store inside it (machine) even if we close the browser or restart the machine the token will still be there

Frontend dev write code like this

(Any request which me make will carry this token with itself) and when i logout then they will just remove token from local storage

```
async function getAnimalData() {
        const response = await fetch(“https://fakeapri.it/conversation ”,{
                headers:{
                        “Authorization”: localStorage.read(“token”)
}
});
Const finalData = await response,json()
}
```



Q/Na

1. Why are we using JWT instead of encryption or others

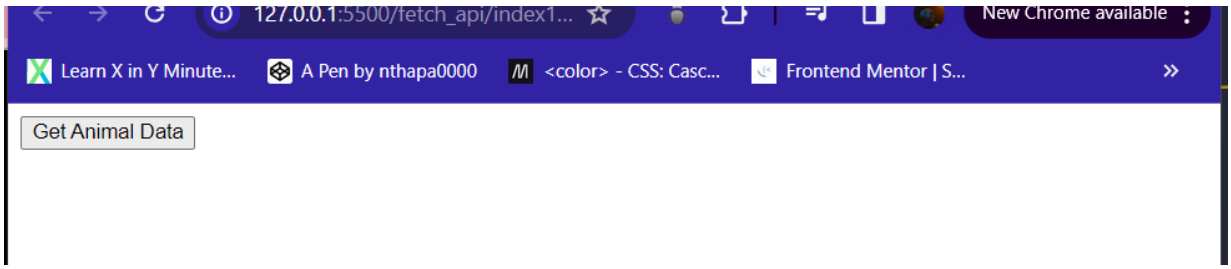
It save us a database calls. (check on internet) also mode adopted way

2. Fetch recall

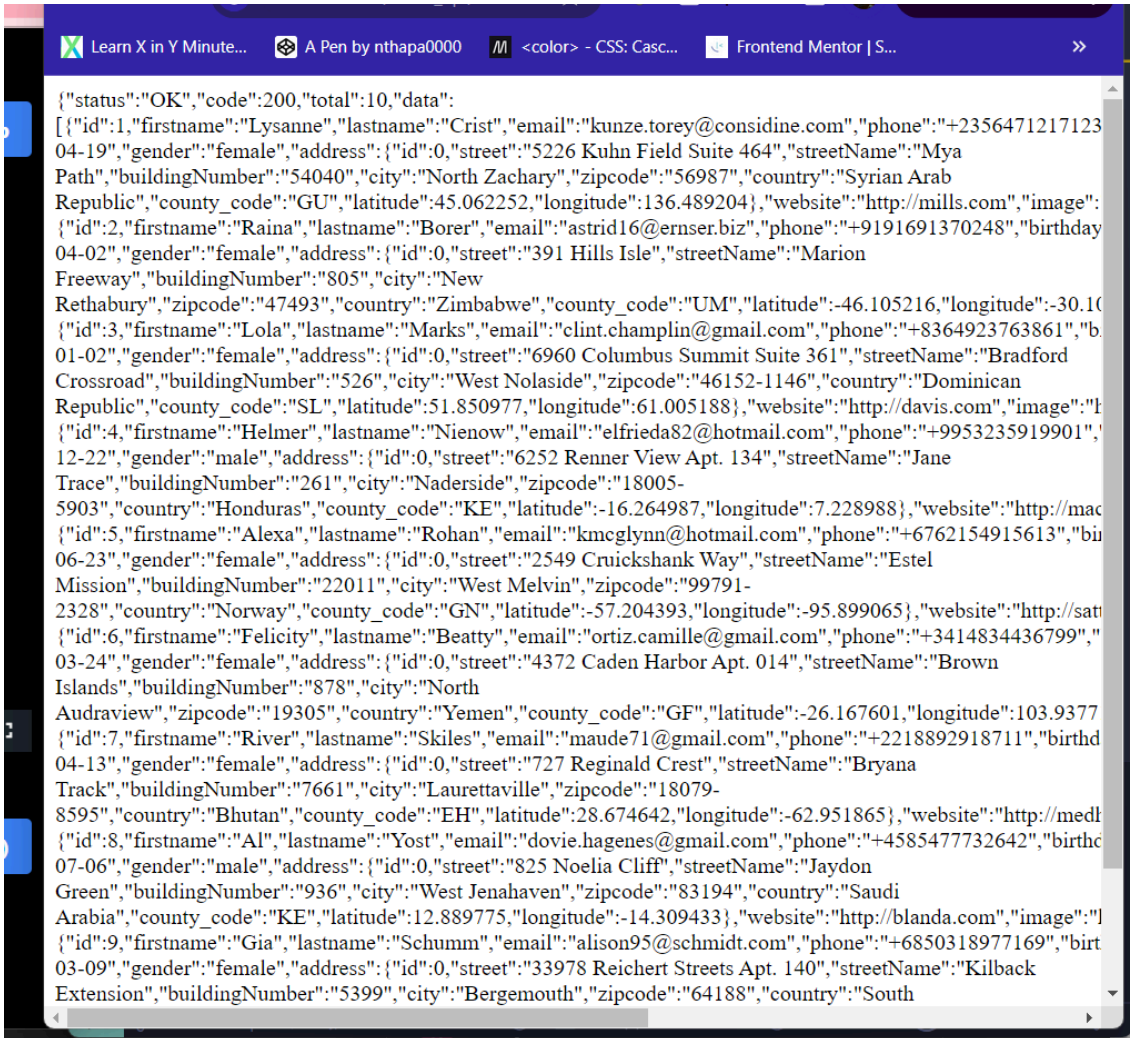
Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        async function getAnimalData(){
            const response = await
fetch('https://fakerapi.it/api/v1/persons')
            const finalData = await response.json()
            document.getElementById('userData').innerHTML =
JSON.stringify(finalData)
        }
    </script>
    <div id="userData">
        <button onclick="getAnimalData()">
            Get Animal Data
        </button>
    </div>
</body>
</html>
```

Webpage

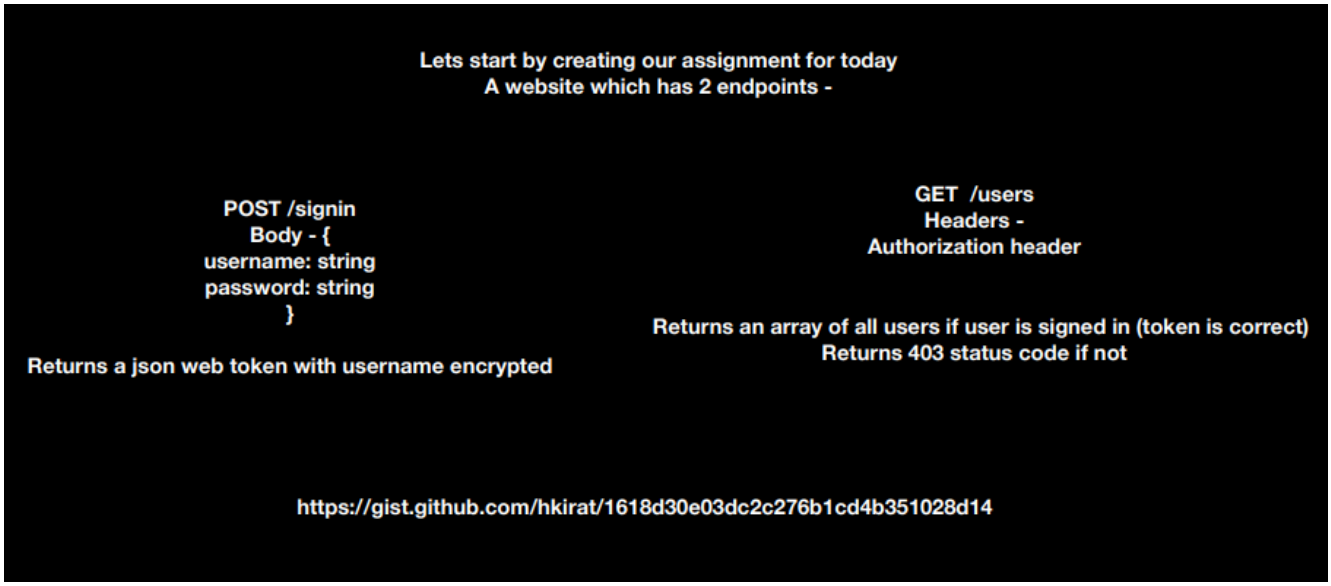


When we click on button Get Animal Data



3. We use Parse when we want to throw an exception and we use safeParse success without exception
4. Where to store the key in encryption-decryption
   Some people store it in GitHub secret etc.
5. We can also use axios library instead of fetch.
   const response = axios.get('https://fakeapi.it//api/v1/persons')

Assignment



Initial codebase

```
const express = require("express");
const jwt = require("jsonwebtoken");
const jwtPassword = "123456";

const app = express();
// in memory users stored in an array
const ALL_USERS = [
  {
    username: "harkirat@gmail.com",
    password: "123",
    name: "harkirat singh",
  },
  {
    username: "raman@gmail.com",
    password: "123321",
    name: "Raman singh",
  },
  {
    username: "priya@gmail.com",
    password: "123321",
    name: "Priya kumari",
  },
];
```

```
function userExists(username, password) {
  // write logic to return true or false if this user exists
  // in ALL_USERS array
}

// exposed to to endpoints
app.post("/signin", function (req, res) {
  const username = req.body.username;
  const password = req.body.password;

  if (!userExists(username, password)) {
    return res.status(403).json({
      msg: "User doesnt exist in our in memory db",
    });
  }

  var token = jwt.sign({ username: username }, "shhhhh");
  return res.json({
    token,
  });
});

app.get("/users", function (req, res) {
  const token = req.headers.authorization;
  try {
    // verifying the token send as headers
    const decoded = jwt.verify(token, jwtPassword);
    const username = decoded.username;
    // return a list of users other than this username
  } catch (err) {
    return res.status(403).json({
      msg: "Invalid token",
    });
  }
});

app.listen(3000)
```

Solution:

https://www.npmjs.com/package/jsonwebtoken to know more about jsonwebtoken

npm install jsonwebtoken express

Code:

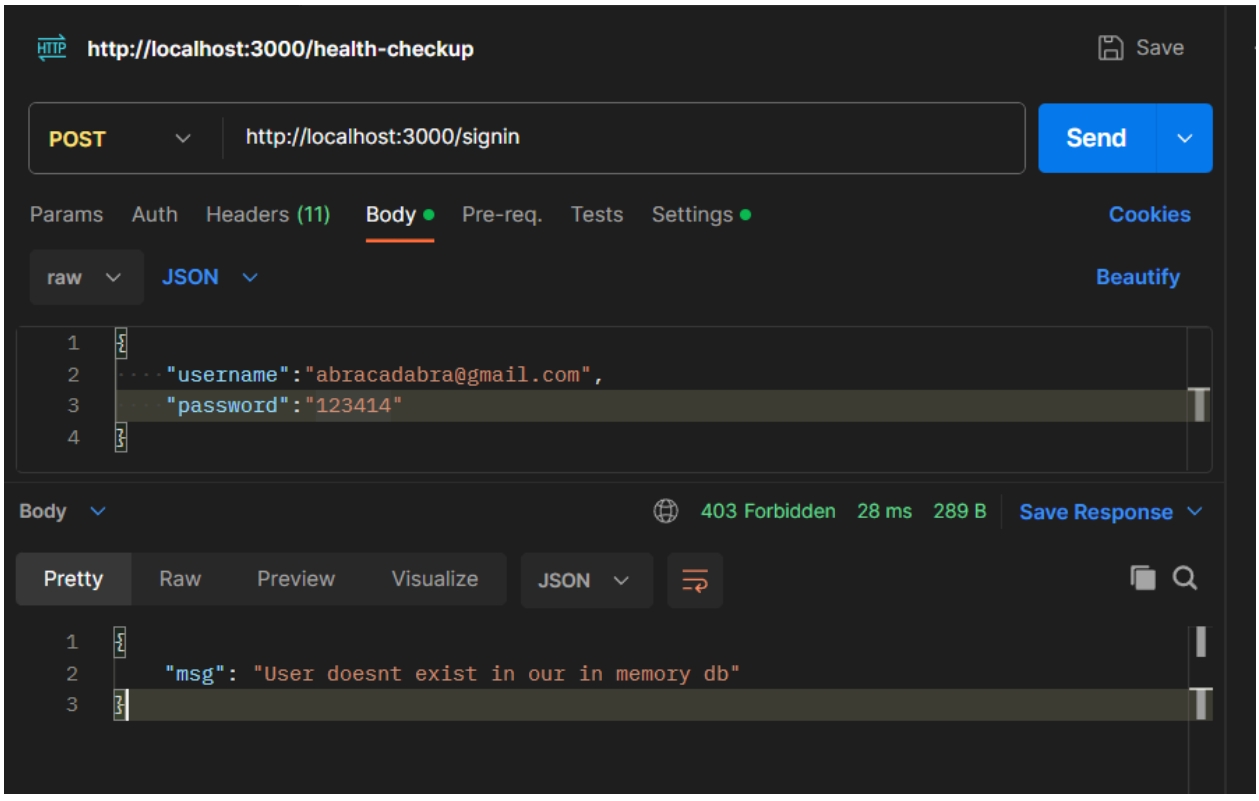
```
const express = require("express");
// import json webtokens, it is by ietf
const jwt = require("jsonwebtoken");
const jwtPassword = "123456";
// usually stored in some secret file

const app = express();
app.use(express.json());
//if no such line then it wont be able to decode the body

// in memory database
const ALL_USERS = [
  {
    username: "harkirat@gmail.com",
    password: "123",
    name: "harkirat singh",
  },
  {
    username: "raman@gmail.com",
    password: "123321",
    name: "Raman singh",
  },
  {
    username: "priya@gmail.com",
    password: "123321",
    name: "Priya kumari",
  },
];

function userExists(username, password) {
```

```
  // write logic to return true or false if this user exists
  // in ALL_USERS array
//  todo: try to use the find function in js
  let userExists = false;
  for(let i=0;i<ALL_USERS.length;i++){
    if(ALL_USERS[i].username == username && ALL_USERS[i].password ==
password){
        // even if one user is found we return true
        userExists = true;
    }
  }
  return userExists;
//   cleaner way to do is find function
}

app.post("/signin", function (req, res) {
    // when request comes to sign in
    // when user send a post request , they need to send username and
password in body
  const username = req.body.username;
  const password = req.body.password;

  if (!userExists(username, password)) {
    return res.status(403).json({
      msg: "User doesnt exist in our in memory db",
    });
  }


  var token = jwt.sign({ username: username }, jwtPassword);
  //get back the token and send to the user
  return res.json({
    // user responsibilty to store the token
    token,
  });
});

app.get("/users", function (req, res) {
    // expect user will send the authorizarion
  const token = req.headers.authorization;
```

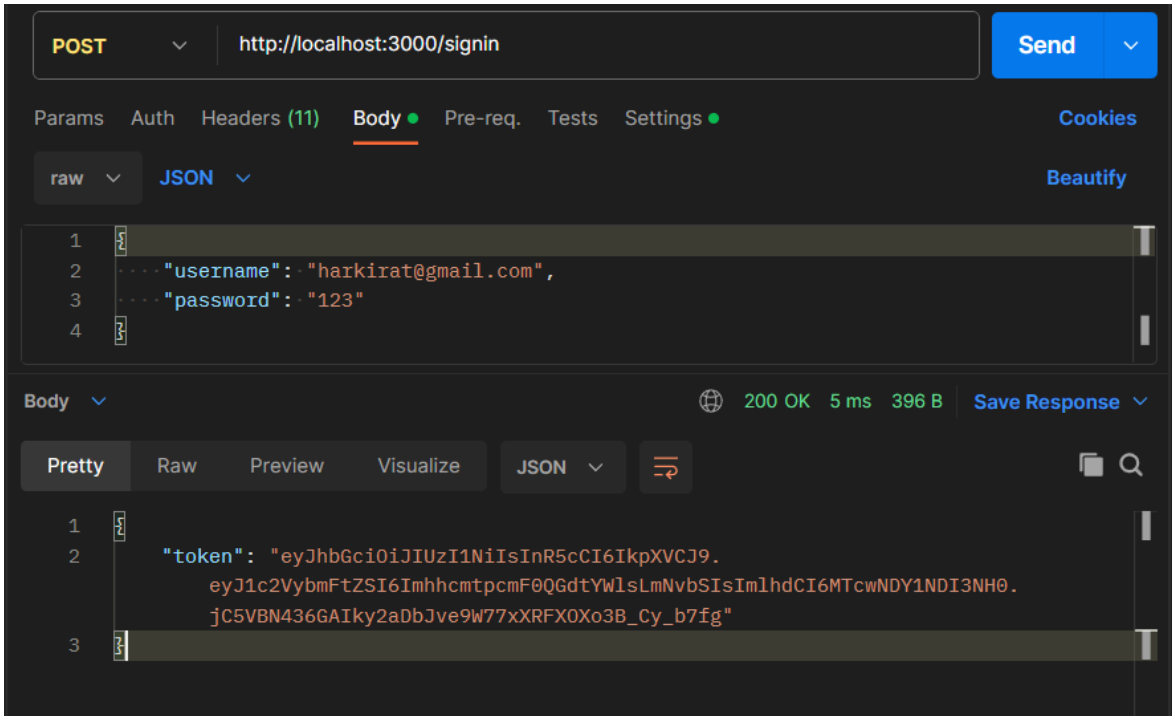
```
  try {
    const decoded = jwt.verify(token, jwtPassword);
    const username = decoded.username;
    // return a list of users other than this username
  } catch (err) {
    // if token is invalid we send 403 status code
    return res.status(403).json({
      msg: "Invalid token",
    });
  }
});

app.listen(3000)
```

Sending request,

1. Username/password is not valid



2. Username/password is valid



every time we send the request even of the same user it will send a unique token as a response

If we want to decode this jwt token we can do it in **jwt.io**

Now returning the list of users in get request when we successfully verify the token send by the user in the authorization headers

Code/Final code

```
const express = require("express");
// import json webtokens, it is by ietf
const jwt = require("jsonwebtoken");
const jwtPassword = "123456";
// usually stored in some secret file
```

```
const app = express();
app.use(express.json());
//if no such line then it wont be able to decode the body

// in memory database
const ALL_USERS = [
  {
    username: "harkirat@gmail.com",
    password: "123",
    name: "harkirat singh",
  },
  {
    username: "raman@gmail.com",
    password: "123321",
    name: "Raman singh",
  },
  {
    username: "priya@gmail.com",
    password: "123321",
    name: "Priya kumari",
  },
];

function userExists(username, password) {
  // write logic to return true or false if this user exists
  // in ALL_USERS array
//  todo: try to use the find function in js
  let userExists = false;
  for(let i=0;i<ALL_USERS.length;i++){
    if(ALL_USERS[i].username == username && ALL_USERS[i].password ==
password){
        // even if one user is found we return true
        userExists = true;
    }
  }
  return userExists;
//   cleaner way to do is find function
}
```

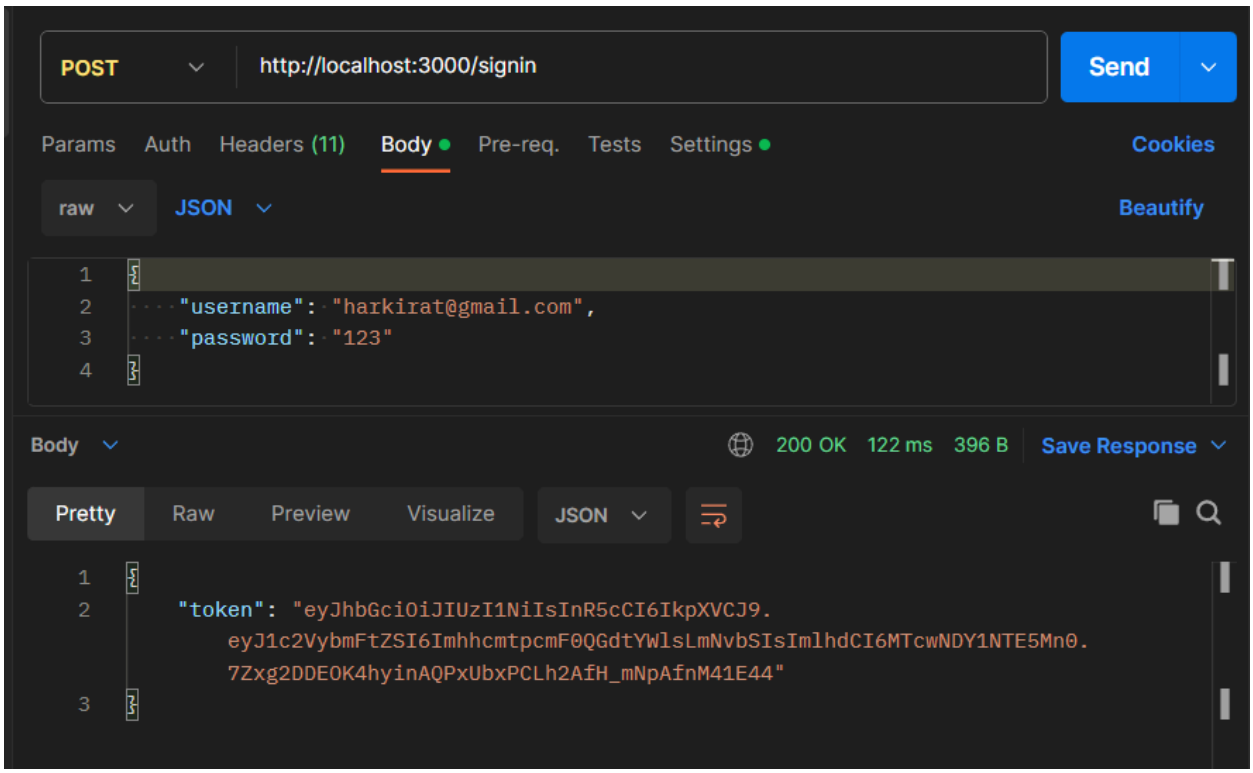
```
app.post("/signin", function (req, res) {
    // when request comes to sign in
    // when user send a post request , they need to send username and
password in body
  const username = req.body.username;
  const password = req.body.password;

  if (!userExists(username, password)) {
    return res.status(403).json({
      msg: "User doesnt exist in our in memory db",
    });
  }


  var token = jwt.sign({ username: username }, jwtPassword);
  //get back the token and send to the user
  return res.json({
    // user responsibilty to store the token
    token,
  });
});

app.get("/users", function (req, res) {
    // expect user will send the authorizarion
    // token in authorization header
  const token = req.headers.authorization;
  try {
    const decoded = jwt.verify(token, jwtPassword);
    const username = decoded.username;
    // return a list of users other than this username
   res.json({
        users: ALL_USERS.filter(function(value){
            if(value.username != username){
                return true;
            }
            else{
                return false;
            }
        })
    })
```

```
  } catch (err) {//catching errors
    // if token is invalid we send 403 status code
    return res.status(403).json({
      msg: "Invalid token",
    });
  }
});

app.listen(3000)
```

Sending the request via postman

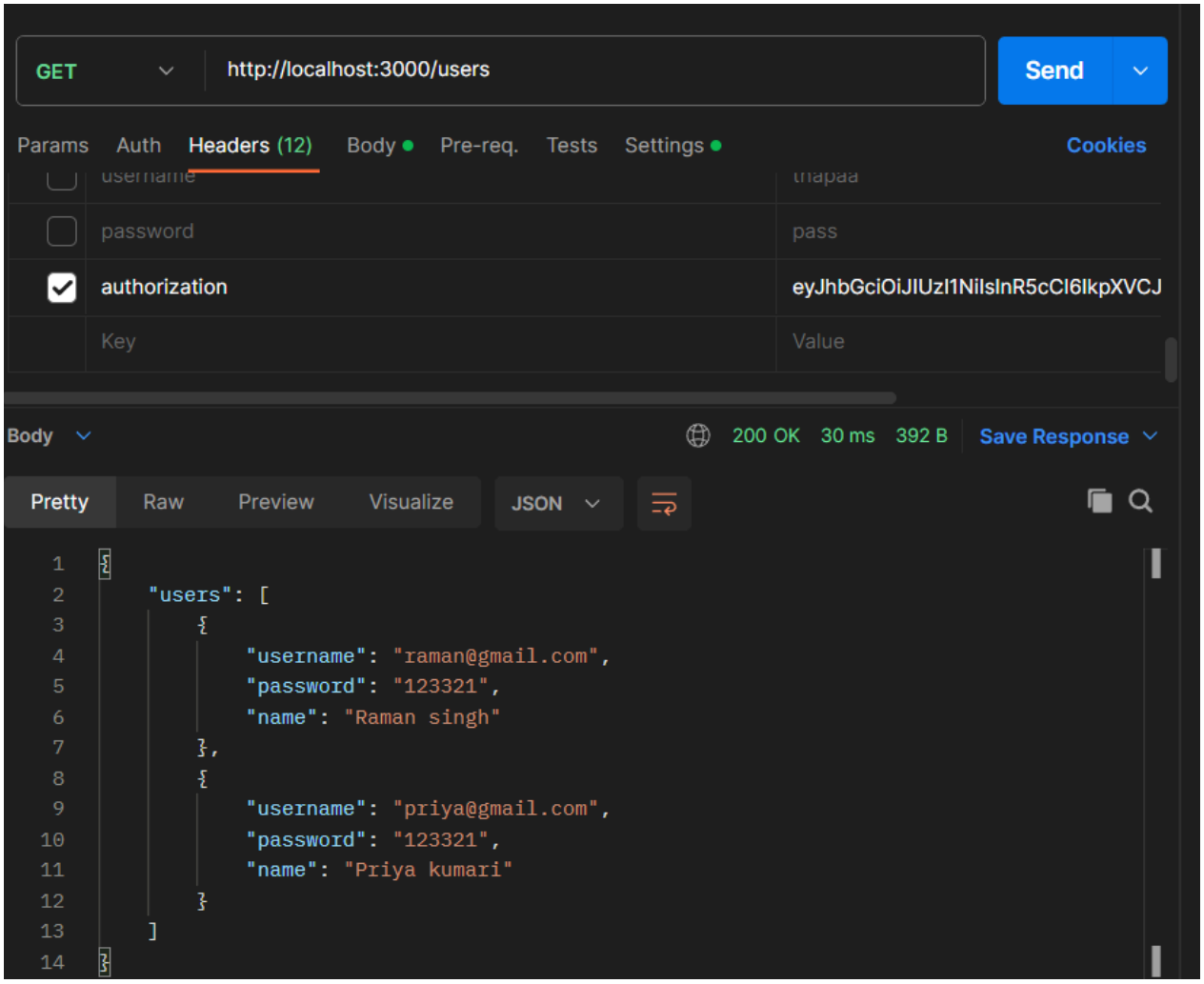
In the authorization header(in GET request) we send token which we get from the POST request.

This is token for the following user.

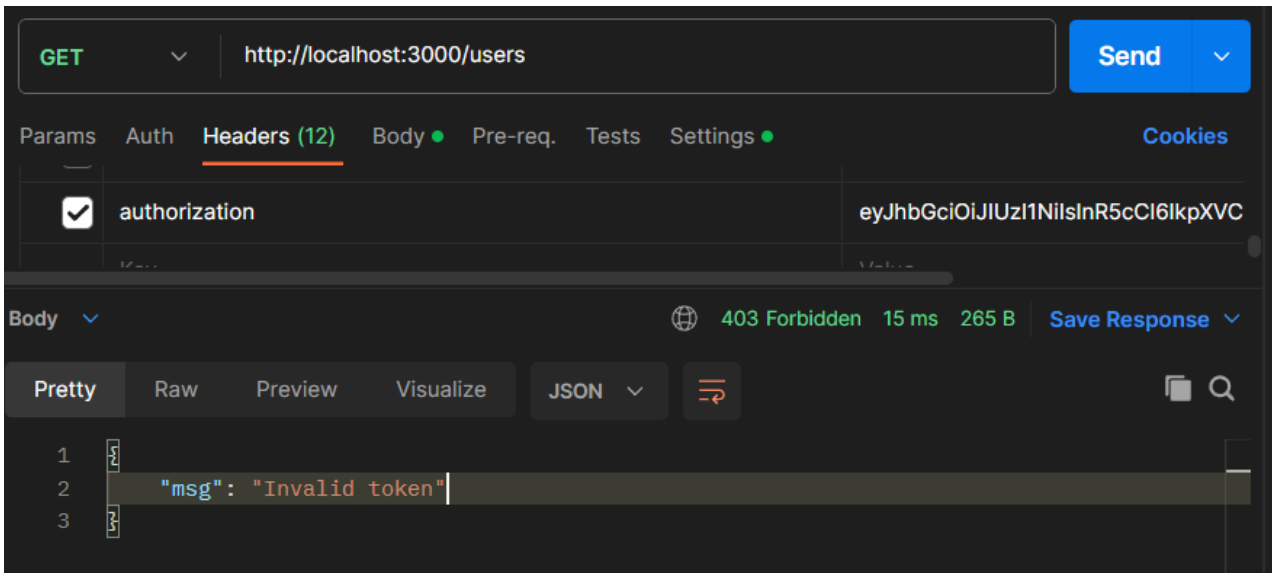


Now we will send this token as Authorization header in GET request

We get response of data of users other than harkirat@gmail.com

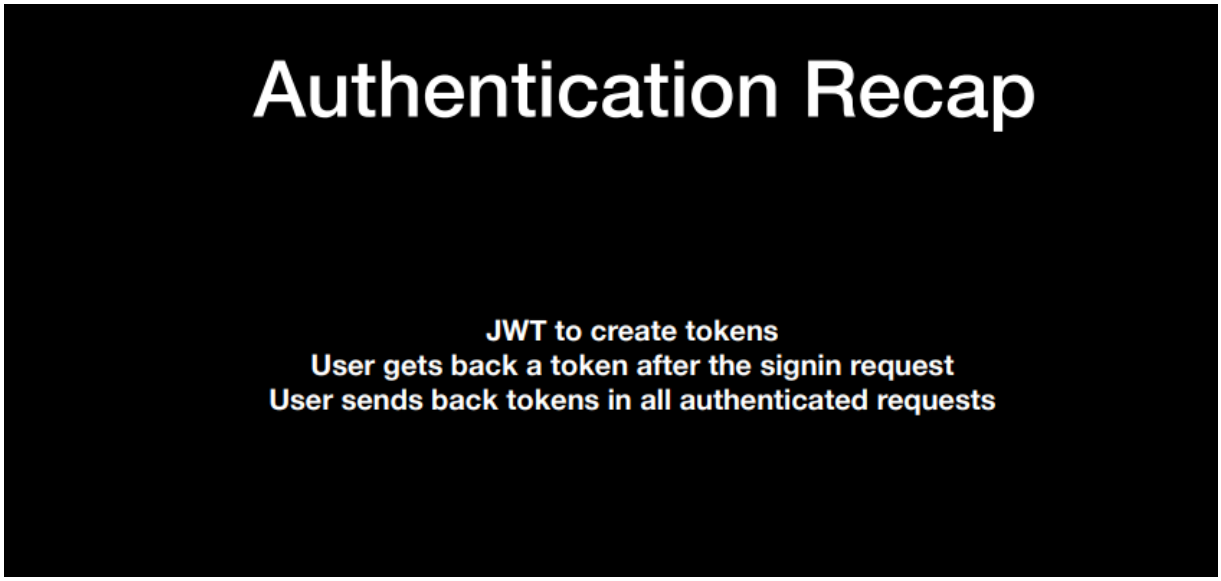


Now if we send an invalid token or no token at all



We will send This error message even if the token is correct but **jwtPassword** is changed from when the token was generated.

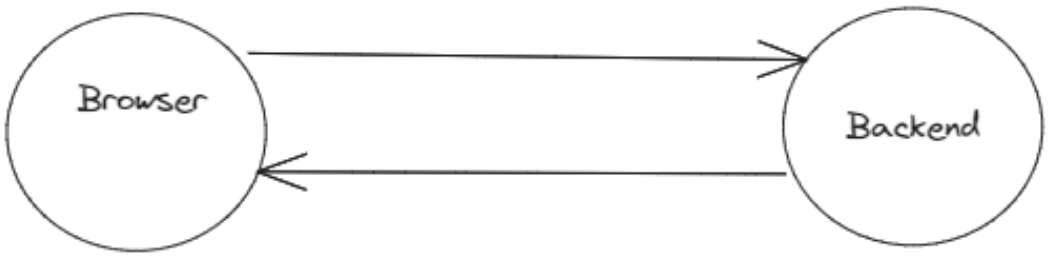
We made a real-world backend with a signin endpoint and another endpoint that does something. It is authenticated and nobody can access it unless they send correct jwt



Databases

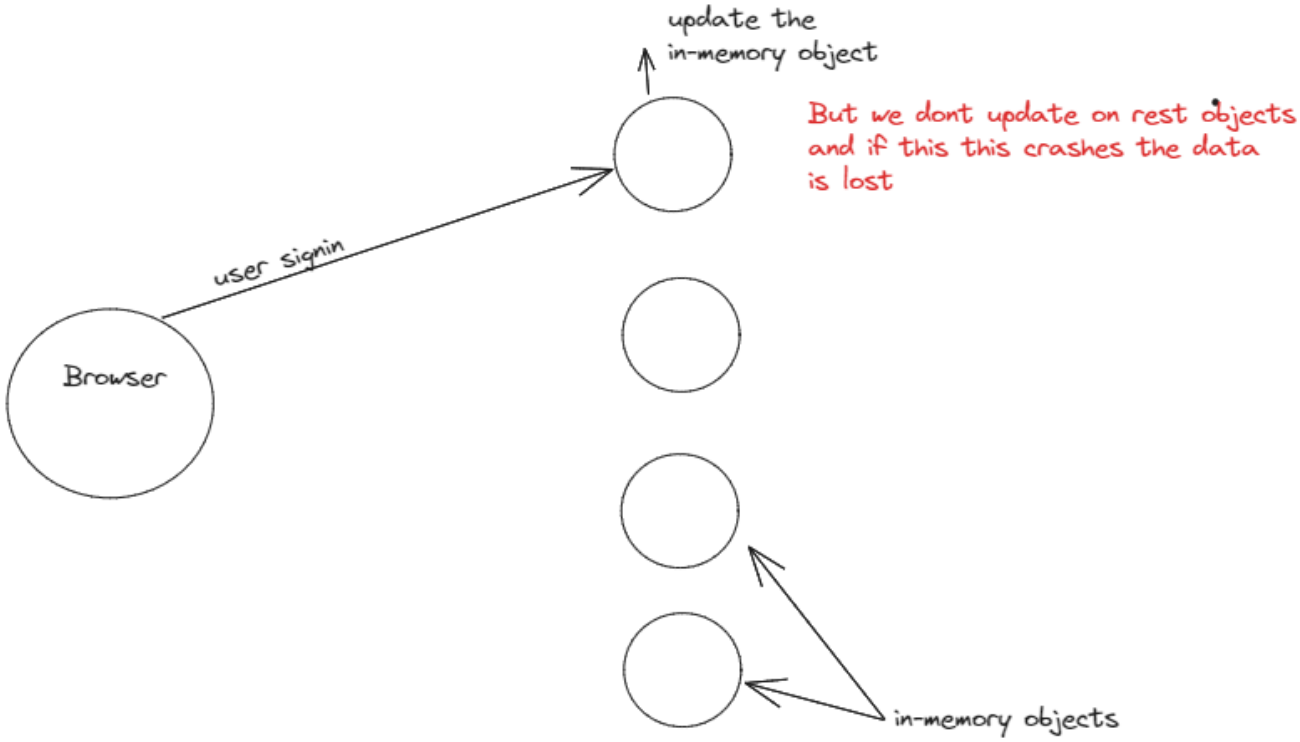
We had in-memory object

- We didnt allow users to sign in themselves
- We don't allow users to delete or leave the platform
- If the user is in the list then it fine otherwise we can't do anything
- All the updates are lost if the program crashes
- We dont have single backend server in real-world



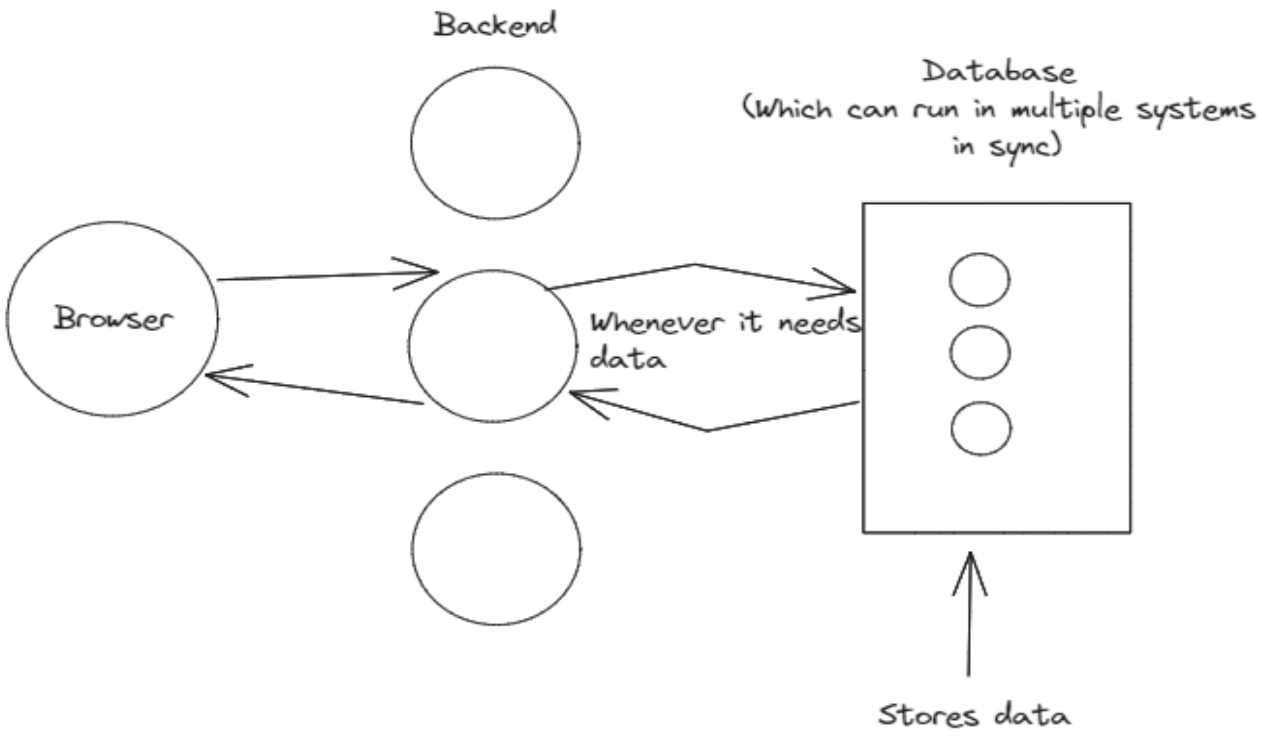
- Instead we have multiple backend servers

Why we must NOT have in-memory database in real world applications



- What will the system kinda look like

User doesn't have access to the database



How is database exposed?

How does it allow backend server to put data on it?

How does it let backend server get the data?

How does backend know the credentials of database?

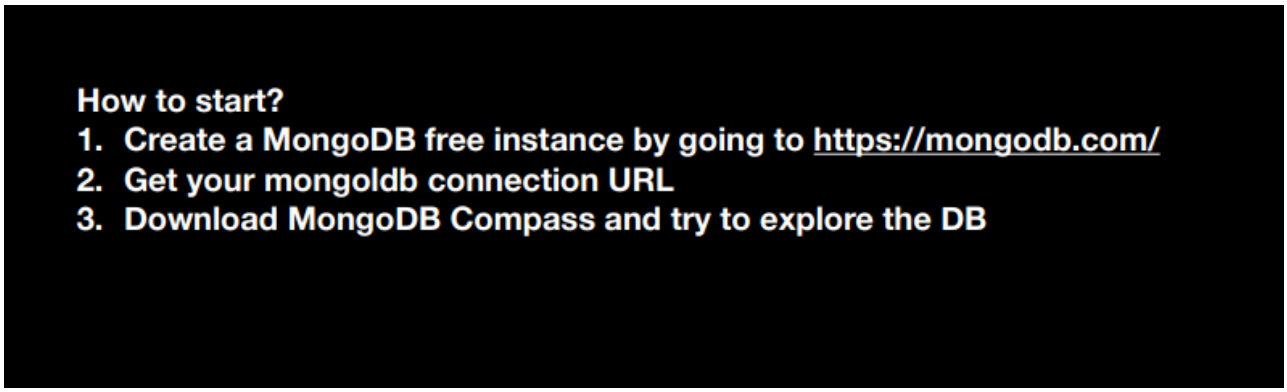
There are various type of databases

1. Graph DBs
2. Vector DBs -> ML
3. SQL DBs -> Most full stack application
4. NoSql DBsm -> MongoDB

MongoDB

- MongoDB lets you create databases (Multiple database) (Facebook has messenger ,FB , insta)

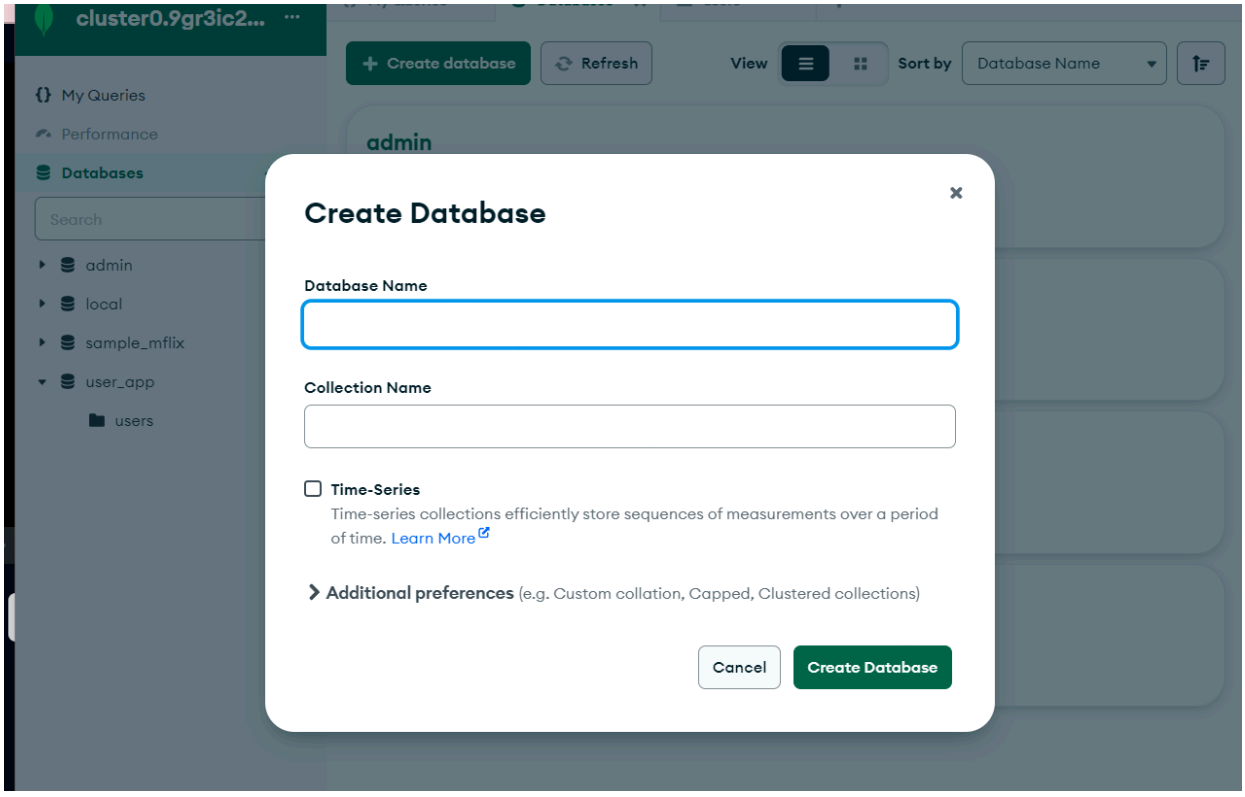
- In each DB it lets us to create tables(collections)
- In each table , it lets you dump JSON data
- It is schemaless
- It scales well and is a decent choice for most use case



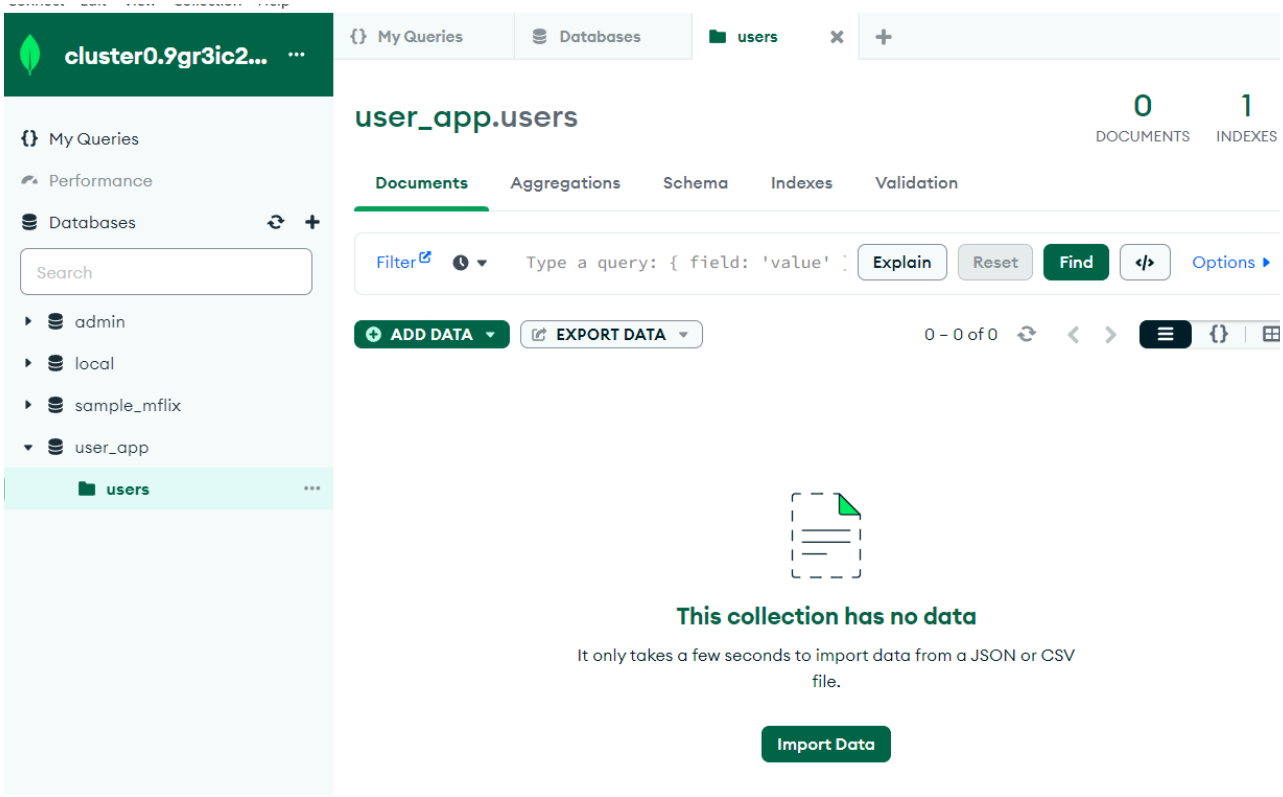
Check Video 3.0.7

MongoDB compass is GUI very helpful while debugging.

Creating a database



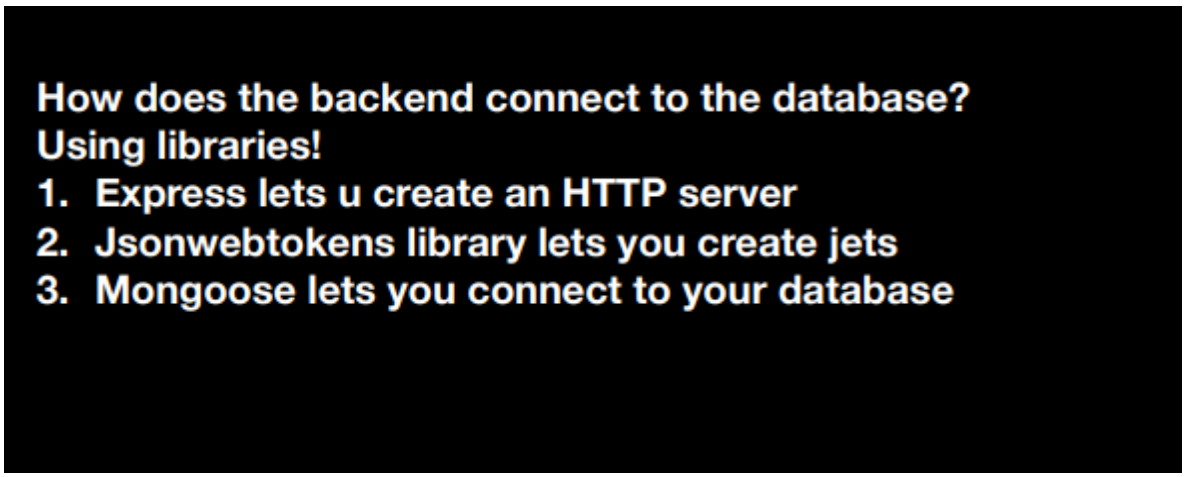
The place where we should store the in-memory data is here.



Now we will create a sign up end point, and when a user sign up its data will be stored in the database.

How can the nodejs process connect to this and put data on it?

By doing this we will know how to write to a database and how to read from the database. This is what is full stack is.



Let's explore **mongoose**.

https://mongoosejs.com/

Let's do an Assignment:

We have to create a backend logic for the server that is connected to the database

End user can send one of three requests:

1. /signup
2. /signin (does this user exist in database and is the password correct) (we return them the jwt so they dont have to sign in again and again)
3. /users (we expect headers (jwt) and return the user the list of users)

Starting Code:

```
const express = require("express");
const jwt = require("jsonwebtoken");
const mongoose = require("mongoose");
const jwtPassword = "123456";

mongoose.connect(
  "your_mongo_url",
);

const User = mongoose.model("User", {
  name: String,
  username: String,
  pasword: String,
});

const app = express();
app.use(express.json());

function userExists(username, password) {
  // should check in the database
}

app.post("/signin", async function (req, res) {
```

```
  const username = req.body.username;
  const password = req.body.password;

  if (!userExists(username, password)) {
    return res.status(403).json({
      msg: "User doesnt exist in our in memory db",
    });
  }

  var token = jwt.sign({ username: username }, "shhhhh");
  return res.json({
    token,
  });
});

app.get("/users", function (req, res) {
  const token = req.headers.authorization;
  try {
    const decoded = jwt.verify(token, jwtPassword);
    const username = decoded.username;
    // return a list of users other than this username from the database
  } catch (err) {
    return res.status(403).json({
      msg: "Invalid token",
    });
  }
});

app.listen(3000);
```

Since MongoDB is schemaless it can have anything. So we use mongoose.

JavaScript

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://127.0.0.1:27017/test');
```

mongoose will first ask what are you planning to put in the database

From client side but still we can put random from mongodb

```javascript
const Cat = mongoose.model('Cat', { name: String });
```

And then only allow to save (put data in database)

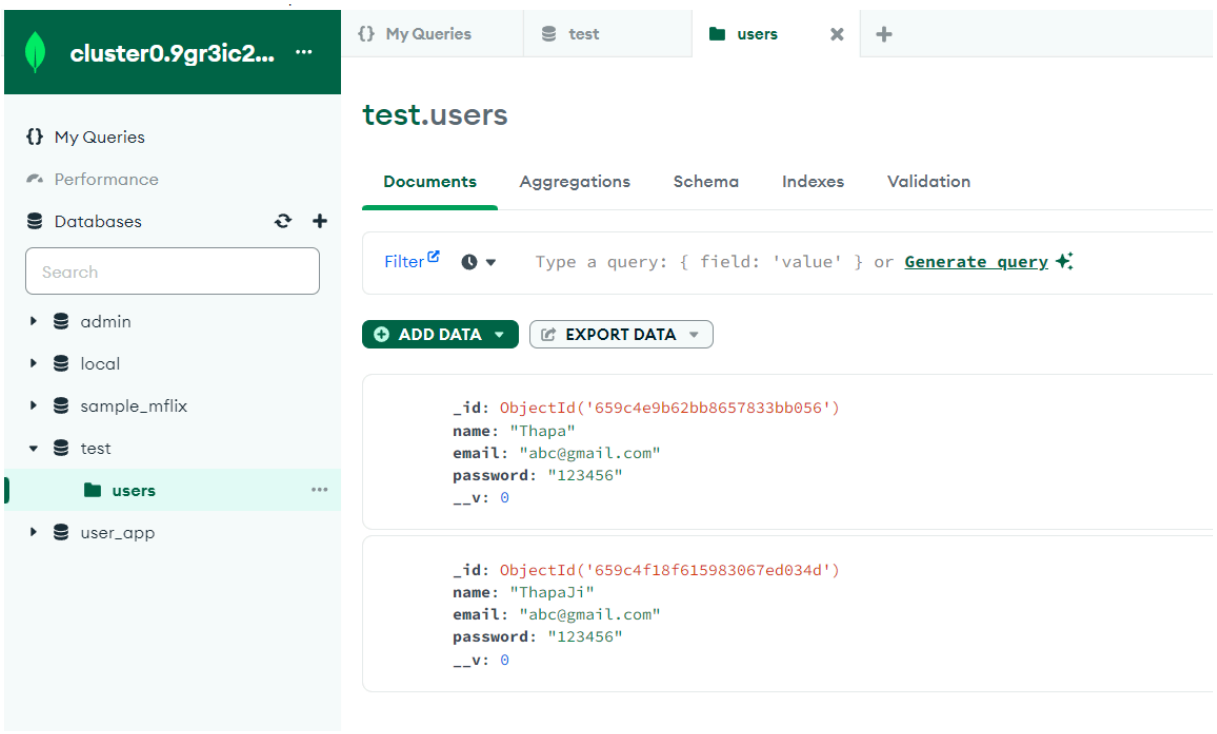
```javascript
const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Example:

```
const mongoose = require("mongoose");

mongoose.connect("mongodb+srv://admin:KmCihXn0l1podXRj@cluster0.9gr3ic2.mongodb.net/")
// put your connect here

// Describing the model / schema/ table
const User = mongoose.model('Users', { name: String, email: String, password: String });

// user table
const user = new User({
    name: "ThapaJi",
    email: "abc@gmail.com",
    password: "123456"
});

// putting data into database
```

```
user.save();
```

npm install mongoose



Now sending data through an http server

Code:

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();

app.use(express.json());
mongoose.connect("mongodb+srv://admin:KmCihXn0l1podXRj@cluster0.9gr3ic2.mo
ngodb.net/")

// Describing the model / schema/ table
const User = mongoose.model('Users', { name: String, email: String,
password: String });
```

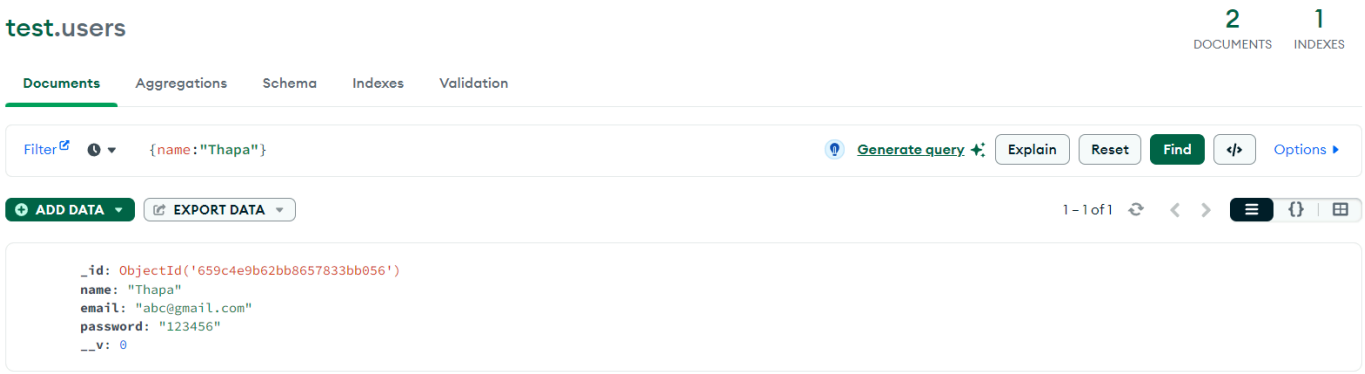
```
// creating a signup endpoint which doesnt put data in  in-memory database
but in the actual database.
app.post('/signup',function(req,res){
    const username = req.body.username;
    const password = req.body.password;
    const name = req.body.name;

    // a bug and there are many more
    // we should also check that does there exist an user with this
username

    const existingUser = await User.findOne({email: username});
    // mongoose documentation
//CRUD: create read update delete

    if(existingUser){
        return res.status(400).json({
            msg: "User already exists"
        })
    }

    const user = new User({
        name: name,
        email: username,
        password: password
    });
    await User.create({
    //     name: name,
    //     email: username,
    //     password: password
    // })

    user.save();
    return res.json({
        msg: "User created successfully"
    })
})

app.listen(3000);
// putting data into the database
```

This code not correct yet.

Q/Na

1. What is findOne function?

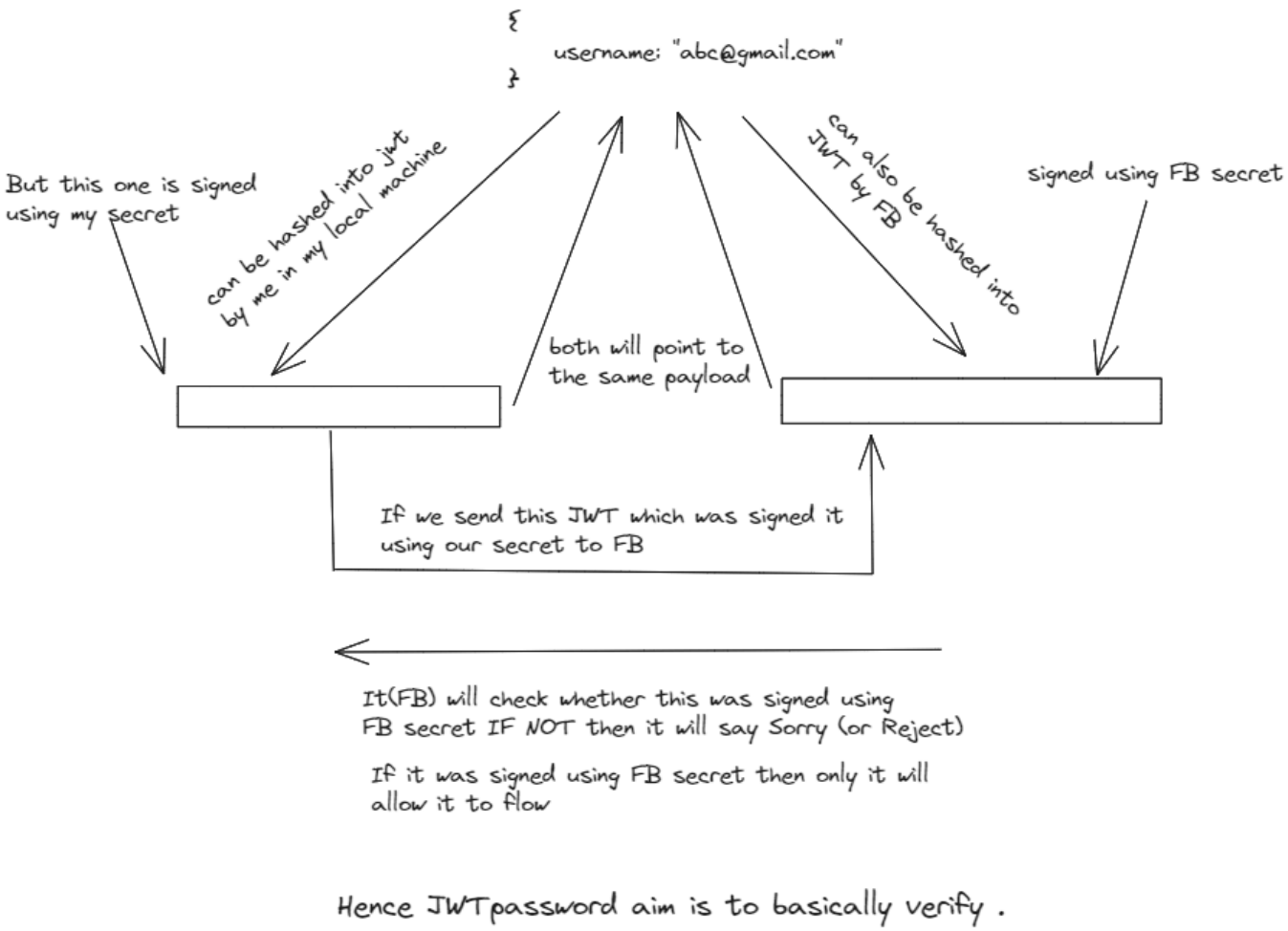
It is similar to what we do in MongoDB Compass



2. User.delete and user.update
3. MongoDB and mongoose what is the difference, in the MongoDB library we don't define schema
4. We will eventually move to axios
5. Jwt is never stored in the server, once we have Jwt, it just checks it
6. Picture like fb are stored in **object stores**

Eg Amazon S3 is commonly used for storing various media files, including pictures and videos. Its object storage system is well-suited for handling multimedia content due to several reasons.

7. Goal of jwtPassword ?? IF it can be decoded by anyone



8. jwt.sign : will give different output mostly due to iat which it includes in payload
9. token can be of various format

   By writing Bearer it tell the format of token which is coming.

   Then we will have actual token
10. Why to store in database when we can store data in a file??\
    - You dont have standard way to store data
    - Hard to distribute
    - Even if we store in mongodb (it is stored in multiple location)
    - DBs are optimized for read and write
    - DBs have logic for indexing to make some queries faster
11. How are Hashing and Encryption are implemented on the code level??

There are some famous algorithms for hashing and encryption (AES-128, SHA-256)

cryptojs library in node will allow us to use this algorithm

crypto.input(“My string to encrypt”).secret(“AscdAfve”)

12. If we every want to decode our data then we will use encryption and if we dont want to decode our data then we will use hashing example: password we dont want anyone to decrypt hence we use hashing and for videos we use encryption before sending it to object store.
13. Local storage function in the browser.

14. How is JWT verifying the user like what it compare with and how it verifies the signature?

```
const jwt = require("jsonwebtoken");
const password = "123456";

const token = jwt.sign({
    name: "harkirat",
},password)
//  this token was created/ hashed using this password
//  whoever has token can see that it is signed jwt token
//  we can verify it whether this token was verified using this password
or not

//  it is similar to we have cheque book in our house suppose it is issued
by the name of my father , i can see the cheque book and if i and my
father both signed the two cheque separately and submitted to the bank.
// Only one signed by my father will be accepted in the bank. The bank
verifies whether the cheque is signed by the person who was issued the
chequebook or not.
```

```
console.log(token);
```

15. Can’t we run MongoDB on a local server??

Yes, we can check out the docker video of Harikat on youtube.

16. How can we store jwt token .

We store them in cookie and local storage and explicitly send them in headers

Cookie are more secure.

It is quite hard to do cookie in mobile app and hence they send fetch request along witht he authorization header

```
fetch ('acs.com',{
    headers: {
        authorization: "Bearer " + localStorage.getItem("token")
    }
})
```

17. We dont need jwtPassword to decode but we need it to sign and verify.
18. Adding expiry to the jwt

```
jwt.sign({
    "name": "harkirat",
    expiresAt: new Date().getTime() + 60*60
})
```

19. jwt.decode() will take input and decode without jwt password
20. We can make middleware to check whether user exists or not.
21. Jwt requires a password during creating a token and during verification
22. findOne() function of mongoose is async since we are asking database to find something