DAY 4

- *Database*

- **Web development = client + server + database**
- Ultimately, Let’s suppose we are going to open Restuarant and there is lots of data around it,

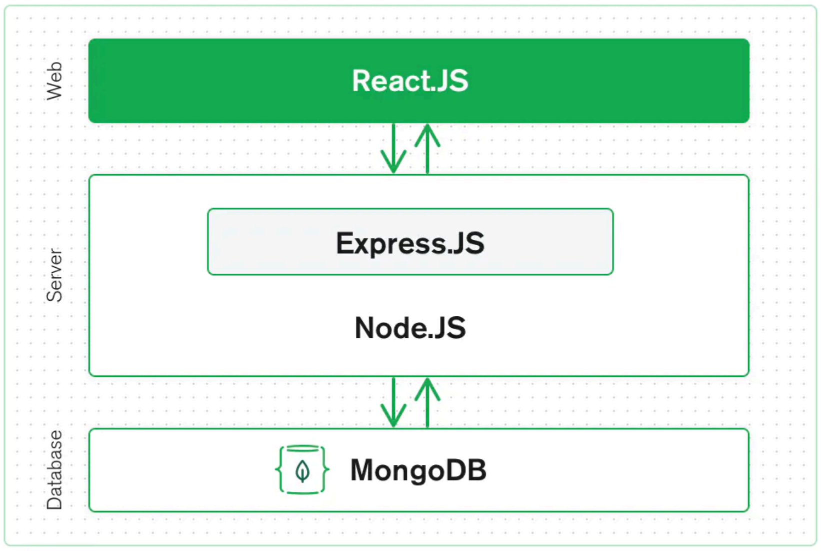
  Number of chefs

  Each Person's Detail (like chef, owner, manager, waiter)
    - Name
    - Age
    - Work
    - Mobile number
    - Email
    - Address
    - salary

  Menu Details (like, drinks, Snacks, main course)
    - Name of dish
    - Price
    - Taste (like, sweet, sour, spicy)
    - Is_drink (boolean true, false)
    - Ingredients (array of data – [“wheat”, “rice”, “sugar”]
    - Number of sales (like 76)

- This is all Data we must have to store to run a fully functional restaurant
- So we have to deal with data

- Now There are lots of Database out there in the market, we can use according to our need
  - SQL
  - PostgreSQL
  - MongoDB
  - MariaDB
  - Oracle

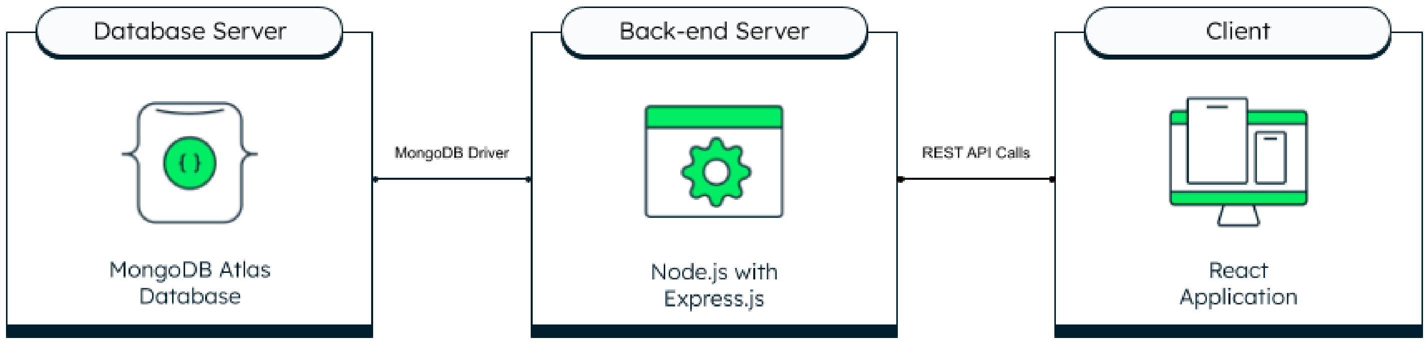


- Databases typically have their own server systems to manage and provide access to the data they store.
- These database server systems are separate from Node.js servers but work together to create dynamic and data-driven web applications

- **Node.js Server and Database Server:**

- A database server is a specialized **computer program** or system that manages databases. It stores, retrieves, and manages data efficiently
- The database server stores your application's data. When your Node.js server needs data, it sends requests to the database server, which then retrieves and sends the requested data back to the Node.js server.

- Node.js server is responsible for handling HTTP requests from clients (like web browsers) and returning responses.
- It processes these requests, communicates with the database server, and sends data to clients.



- *Setup MongoDB*

Q) So as We are creating a backend server, as same do we need to create a Database server as well?

Ans) NO

—> **Setup MongoDB locally is quite tough for most of you. But relax we are here to help you**

- Go to MongoDB's official website
- Or Google MongoDB download

For MacOS
https://stackoverflow.com/questions/65357744/how-to-install-mongodb-on-apple-m1-chip

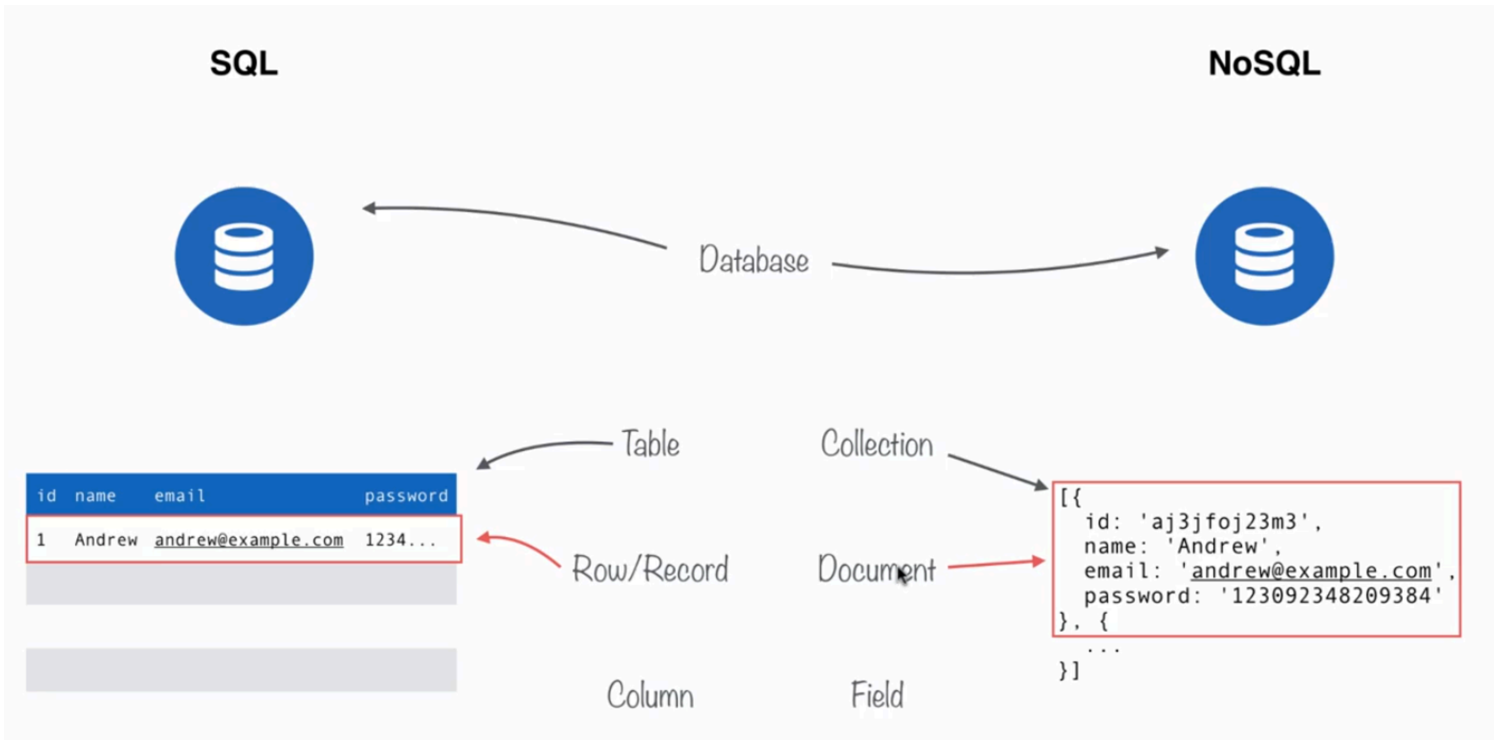
For Windows
https://www.geeksforgeeks.org/how-to-install-mongodb-on-windows/

—- Trust me Give some time to it until you will download the MongoDB in your system. There is no point to move ahead without installing MongoDB

- *MongoDB Syntax and Queries*

Now As we know there is 2 thing, MongoDB

- → Start **MongoDB Server**
- → now we can use **MongoDB shell** from where we can interact with databases



```
[{
  id: 'aj3jfoj23m3',
  name: 'Andrew',
  email: 'andrew@example.com',
  password: '123092348209384'
}, {
  ...
}]
```

SQL	MongoDB
database	database
table	collection
column	field
row (record)	document

```
{
  _id: "5cf0029caff5056591b0ce7d",
  firstname: 'Jane',
  lastname: 'Wu',
  address: {
    street: '1 Circle Rd',
    city: 'Los Angeles',
    state: 'CA',
    zip: '90404'
  }
}
```

1. Create a Database:

In SQL:

```
CREATE DATABASE mydb;
```

In MongoDB:

```
use mydb;
```

2. Create a Table (Collection in MongoDB):

In SQL:

```
CREATE TABLE users (
  id INT PRIMARY KEY,
  username VARCHAR(50),
  age INT
);
```

In MongoDB:

```
db.createCollection("users");
```

3. Insert Data:

```
INSERT INTO users (id, username, age)
VALUES (1, 'Alice', 25);
```

In MongoDB:

```
db.users.insertOne({ id: 1, username: 'Alice', age: 25 });
```

4. Query Data:

In SQL:

```sql
SELECT * FROM users WHERE age > 21;
```

In MongoDB:

```javascript
db.users.find({ age: { $gt: 21 } });
```

5. Update Data:

In SQL:

```sql
UPDATE users SET age = 22 WHERE username = 'Alice';
```

In MongoDB:

```javascript
db.users.updateOne({ username: 'Alice' }, { $set: { age: 22 }
});
```

6. Delete Data:

In SQL:

```sql
DELETE FROM users WHERE id = 1;
```

In MongoDB:

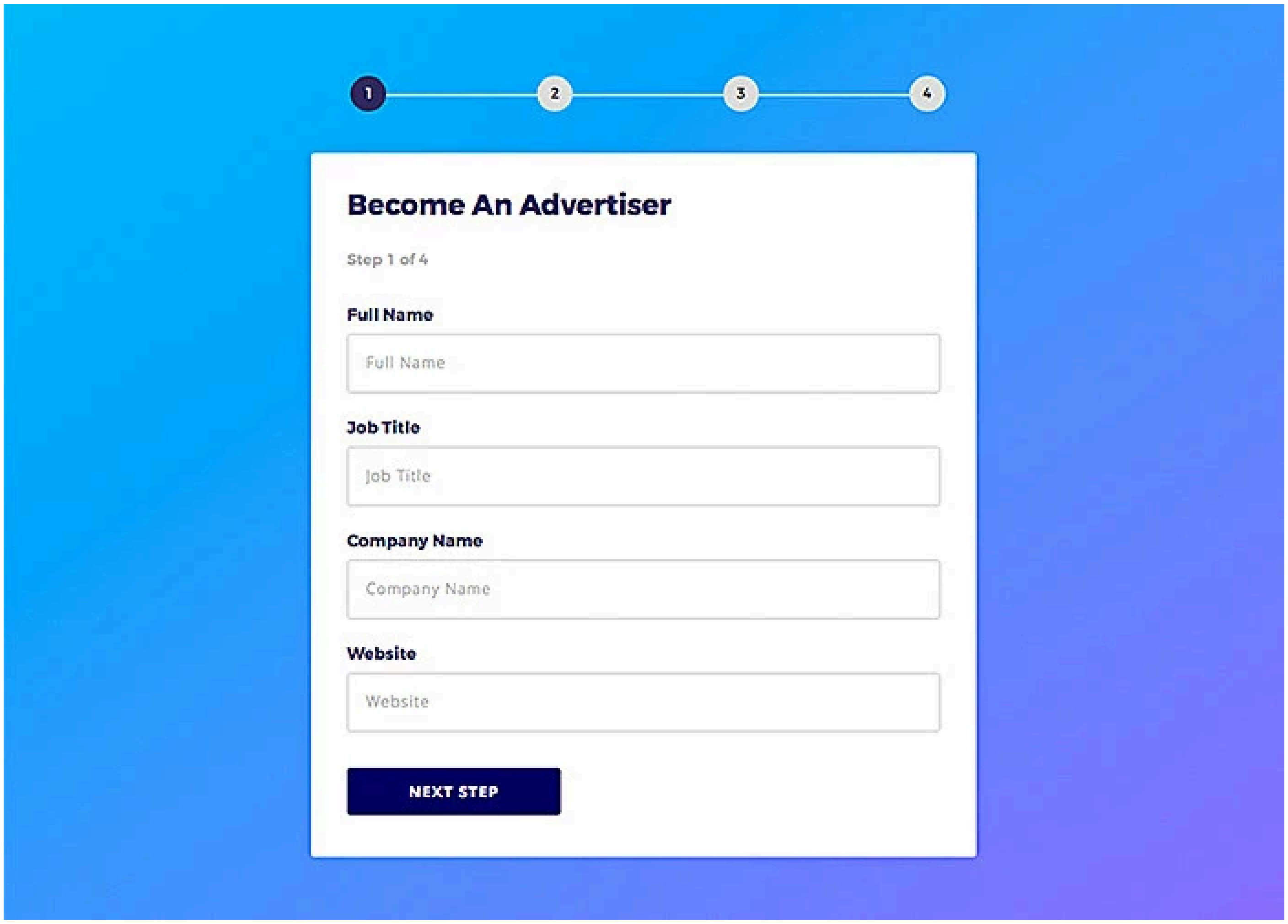
```javascript
db.users.deleteOne({ id: 1 });
```

- *MongoDB Compass GUI*

- There are lots of Tools in the market that help to visualize data like mongoDB compass, MongoDB Robo 3T
- mongodb://127.0.0.1:27017

- *Data Desing and Postman*

- Now in order to use the database we have to integrate MongoDB with nodejs
- Now, there should be a form built on ReactJS or HTML or CSS to add chef or person details
- Now currently we don't have a such frontend thing, so we are using **Postman** for this



- Every Frontend Application, collect Data format it according to backend Requirements and then will send it to backend APIs

- Let's suppose creating Dummy Data for Person
- Each Person's Detail (like chef, owner, manager, waiter)

```
{
      "name": "Alice",
      "age": 28,
      "work": "Chef",
      "mobile": "123-456-7890",
      "email": "alice@example.com",
      "address": "123 Main St, City",
      "salary": 60000
}
```

- Each Menu's Detail

```
{
      "name": "Mango Smoothie",
      "price": 4.99,
      "taste": "Sweet",
      "is_drink": true,
      "ingredients": ["mango", "yogurt", "honey"],
      "num_sales": 45
 }
```

```
{
      "name": "Spicy Chicken Wings",
      "price": 9.99,
      "taste": "Spicy",
      "is_drink": false,
      "ingredients": ["chicken wings", "spices", "sauce"],
      "num_sales": 62
}
```

- *Connect MongoDB with NodeJS*

- Now, To connect MongoDB with NodeJS we need a MongoDB driver (a set of programs)

- A MongoDB driver is essential when connecting Node.js with MongoDB because **it acts as a bridge** between your Node.js application and the MongoDB database.

- MongoDB speaks its own language (protocol) to interact with the database server.
- Node.js communicates in JavaScript.
- The driver translates the **JavaScript code from Node.js** into a **format that MongoDB can understand** and vice versa.
- The driver provides a set of functions and methods that make it easier to perform common database operations from your Node.js code.
- The driver helps you handle errors that might occur during database interactions. It provides error codes, descriptions, and other details to help you troubleshoot issues.

- The most popular driver is the official MongoDB Node.js driver, also known as the `mongodb` package.

```
npm install mongodb
```

- *Mongoose*

- Now but we are going to use Mongoose, rather than `mongodb`

- Mongoose is an ***Object Data Modeling (ODM)*** library for MongoDB and Node.js
- There are lots of reasons we prefer Mongoose rather than a native official driver
- Things are a lot easier here

(Relate Real life Examples with mobiles with earphones)

- Mongoose is like a translator between your Node.js code and MongoDB. It makes working with the database smoother and easier.

- With Mongoose, you can define how your data should look, like making a blueprint for your documents. It's like saying, "In our database, each person's information will have a name, age, and email." This makes sure your data stays organized.
- Mongoose helps you make sure the data you put into the database is correct. It's like having someone check if you've written your email address correctly before sending a message.
- Very easy to query from the database

—> But if you are using `mongodb` Native Driver

- You need to write a lot of detailed instructions to make sure everything works correctly.
- Without Mongoose, your code might get messy and harder to understand.
- Since you need to handle many details yourself, it can take longer to finish your project.

In a nutshell, using Mongoose makes working with MongoDB in Node.js much simpler and smoother. It gives you tools that handle complexities for you, so you can focus on building your application without getting bogged down in technical details.