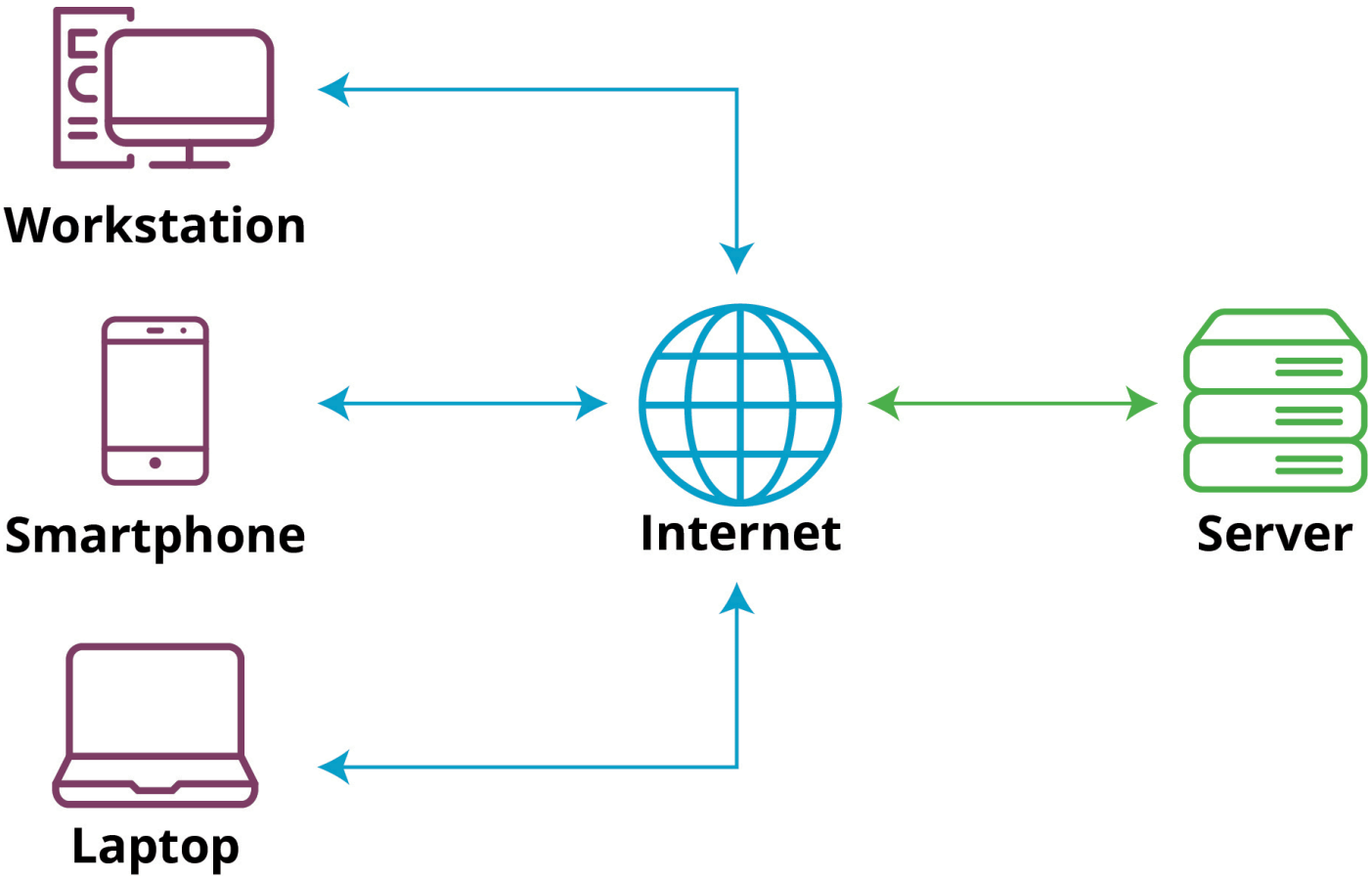
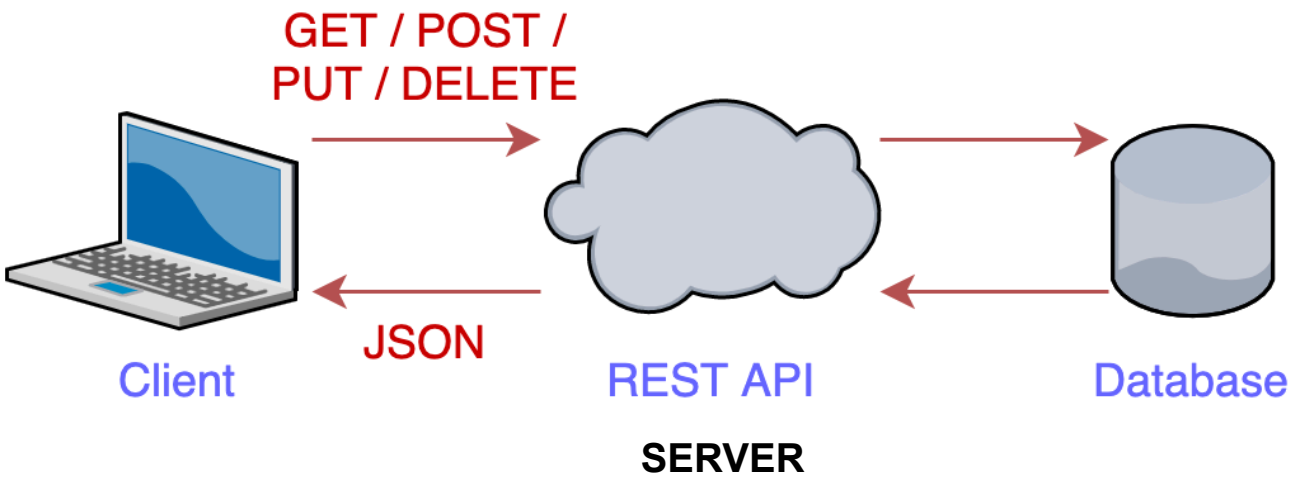
DAY 3

- *Server in Nodejs*

- **Server** – A server is a Person who communicates with clients
- Analogy → server = waiter
- Analogy → chef = database

- A server is a **computer program** that's responsible for preparing and delivering data to other computers
- web pages, images, videos, or any additional information





JSON: JavaScript Object Notation

- Imagine you're sending a message to your friend, and you want to include information like your **name, age, and a list of your favorite hobbies**.
- You can't just send the message as is,
- you need to organize the information in a way that both you and your friend understand.
- JSON is a bit like this organized format for exchanging data between computers.

- JSON is a lightweight
- Structured and organized Data because
- in most contexts, JSON is represented as a string

```
{
  "name": "Alice",
  "age": 25,
  "hobbies": ["reading", "painting", "hiking"]
}
```

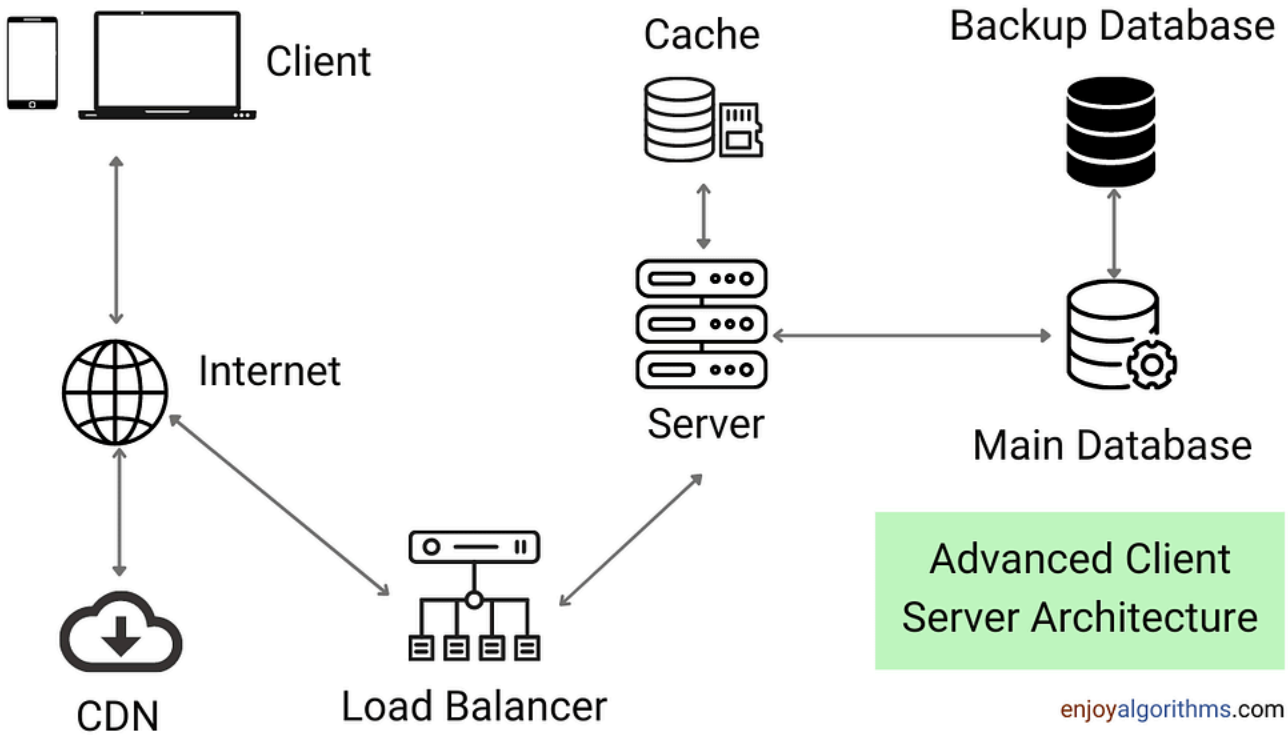
- **Inter Conversion JSON to an Object in Node.js:**

```
const jsonString = '{"name": "John", "age": 30, "city": "New York"}';
const jsonObject = JSON.parse(jsonString); // Convert JSON string to object
console.log(jsonObject.name); // Output: John

-----------------------------------------------------------

const objectToConvert = { name: "Alice", age: 25 };
const jsonStringified = JSON.stringify(objectToConvert); // Convert object
JSON string
console.log(jsonStringified); // Output: {"name": "Alice", "age":25}
```

- **ADVANCED ARCHITECTURE OF WEB FLOW**



- *What are API and Endpoints?*

- Imagine a menu card in a restaurant
- Lots of options are there, each option will give you a different order
- Now, collection of that list = Menu card = API's
- And an option in that list = Endpoint
- And the waiter only understood whatever things are written on the menu card

- *Create a server*

- Creating a server in NodeJs via **express** package

- Express.js is a popular framework for building **web applications** and **APIs** using Node.js.
- When you create an Express.js application, you're setting up the **foundation for handling incoming requests** and defining how your application **responds** to them.

- Now we are going to create a server == waiter
- Now the waiter has his own home?

In simple terms, "**localhost**" refers to your **own computer**. After creating a server in NodeJS, you can access your environment in 'localhost'

- Port Number?
- Let's suppose in a building – 100 rooms are there, for someone to reach he must know the room number right?

- *Methods to share data*

- Now, in the world of web development, we need to deal with data
- How data is sent and received between a client (like a web browser) and a server (built with Node.js)

- So there are lots of methods out there to **send or receive data** according to their needs.

- GET
- POST
- PATCH
- DELETE

- **GET**

- Imagine you want to read a book on a library shelf.
- You don't change anything
- you just want to get the information.

Similarly, the GET method is used to request data from the server.

For example, when you enter a website URL in your browser,
your browser sends a GET request to the server to fetch the web page.

- Code that we have written on the videos

```
const express = require('express')
const app = express();


app.get('/', function (req, res) {
    res.send('Welcome to my hotel... How i can help you ?, we
have list of menus')
})

app.get('/chicken', (req, res)=>{
    res.send('sure sir, i would love to serve chicken')
})

app.get('/idli', (req, res)=>{
    var customized_idli = {
        name: 'rava idli',
        size: '10 cm diameter,
        is_sambhar: true,
        is_chutney: false
    }
    res.send(customized_idli)
})

app.listen(3000, ()=>{
    console.log('listening on port 3000');
})
```