1] Backend overview :-

Page No.
Date

⇒ Backend :-

Server : a software which some thing serve. which can ~~rhythm~~ computers.

→ Not large computer which myths ☺

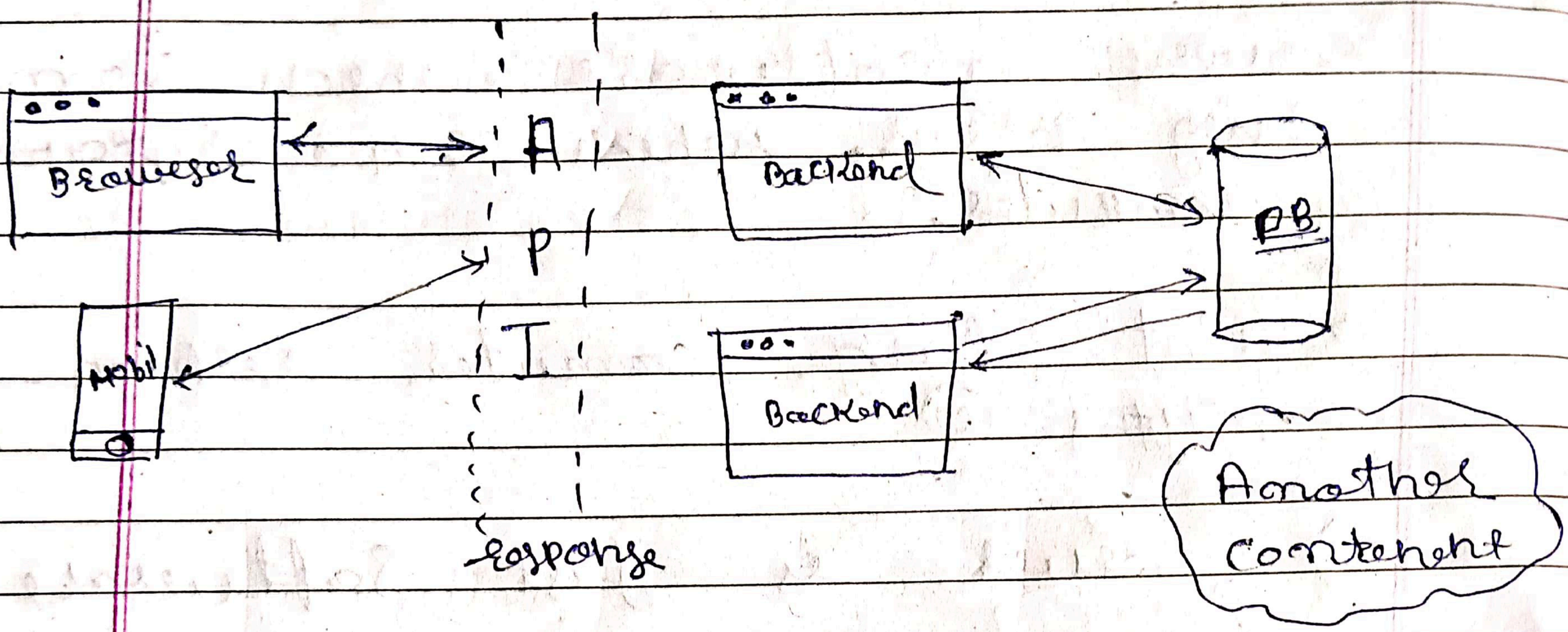
⇒ Server is just software.

⇒ 2 Major components :-

Programming lang.	A database
→ Java, JS, PHP, golang c++, etc & it's framework.	→ Mongo, MySQL, Postgres, sqlite,
⇒ we use lang and its framework for case.	→ ORM, ODM etc: mongoose,

Page No.
Date

⇒ How the backend works.



⇒ Backend is the Processing of Storing, Retrieving and manage data through the API and framework for storing data.

⇒ A Java Script based backend.

⇒ whenever we make the backend we deal & only with only 3 things.

Data	File	Third Party (API)
String, Num, Obj. any	Img, Pdf, etc.	Google login, AWS, map etc.

⇒ A JS Runtime : Node | Deno | Bun

we can use any of the above to run JS in backend.

⇒ File Structure :-

→ package.json : Dependecies file manage
→ .env : for storing env variable
→ etc (Readme, git, gitent,)

SRC ⇒

- DB
- Models
- controllers
- routes
- Middleware
- utils
- More (depends)

File :-

index
↳ DB conn

App
↳ config
↳ cookie
↳ url encode

constant
↳ enums
↳ DB-name

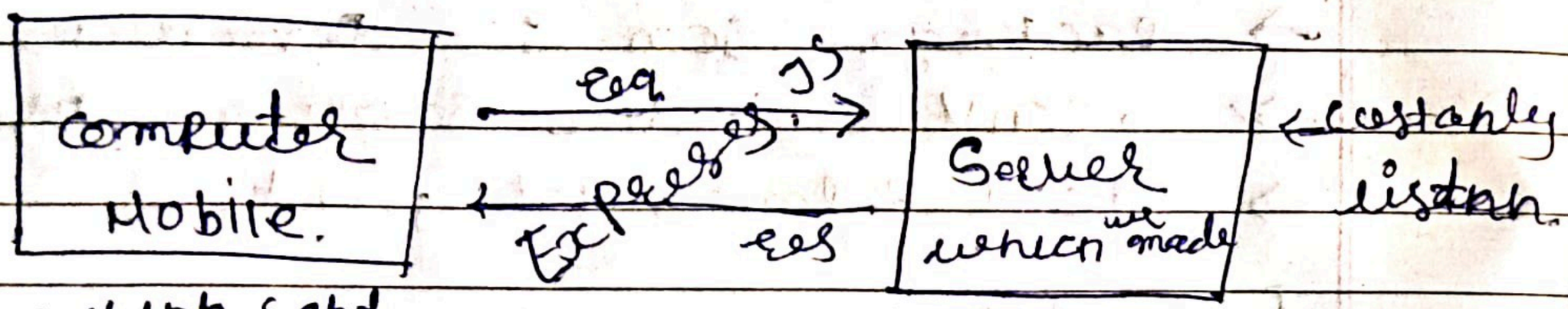
Page No.
Date

2] How to Deploy.

⇒

Just Knowledge:- we use Node.
So check it.
node -V
npm -V

⇒



⇒ Now, we make Server.

1] ~~npm~~ npm init to make Folder Struct. and description.

npm init (-y) ← for automatically & default.

2] Script:
"start" : "npm nodemon index.JS"
dev
build
etc.

Page No.
Date

⇒ It's for starting or running the file or server.

⇒ Express :- install it.

→ we used it for routing.

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

const app = express();
const Port = 4000;
```

⇒ Now we have Passed to use express through app Variable

⇒ PORT : where our Server is listen.
- it just Virtual Port like physic all port is our comp..
- there is may 65535 Port.

```
app.get("/" (req, res) => {
    res.send("Hello World");
}

app.listen( port, () => {
    c.log("..running on{Port} }
```

Page No.

Date

=> I PEST Your .env Variable and deploy on your fav. host Plat form.

Page No.
Date

3] Connect Backend Frontend.

⇒ there is three ways to use files or Package.

① comon JS ② module JS.

ex: require("express") ex: import "" from

⇒ If you use ② module JS then you have tell to Package.Json file "type": "module" in file.

⇒ So, it treat the files as module.

⇒ If you use it and dont specify in Package.Json file it gives error.

Tips:- whenever you get JSON data for bette visualization & understand go the JSON formater and lintly other and post your JSON data that beautify in tree, graph structure. to see proffesional way

⇒ Toal chain /bundler :-

→ to make react app from Vite, CRA, Parcel etc which convert the make bundler of HTML, CSS, JS which web under stand end of the day.

⇒ CI/CD like introduce in this era mean where this tool chain are come in market.

⇒ Make a front-end.

Create Vite @ latest .

Tips:- dot is for create app in only this folder instead of create a other folder.

⇒ For fetching & other request action in front end use lye axios

→ web not have to parse data, it do automatically.

cross origin resource sharing

Page No.
Date

⇒ CORS :- It provide safety to your application. (Safety.)

⇒ CORS Policy origin error :- you might be face this error.

Ex: we not allow in our home to other unknown person same it not allow other API in our origin.

→ whenever our frontend backend are in deffrent domain, port, Hosted this are cross origin.

Solution-1 to say your backend team to make your IP, or API whitelist.

Solution:

⇒ . to install CORS package and use it.

→ and using this package define your origin and also you can define your whitelist api.

4] Data modeling for Backend with mongoose.

Tips: fresher are just starting work and experienced are asking question on it, what is, why, how it step. for proper requirements.

⇒ In this lecture owners of the How the application to build which process, best practices and like all things are deciding before starting build application.

⇒ Data modeling :

→ It is the process and building the application of which type of Data we required.

⇒ How they are relation with the others data

→ In fact, there all team for it to deciding the what we required data

@Jumed Khan Pathan

Page No.
Date

⇒ After that they build model with the help of tools like Moon Modeler and etc.

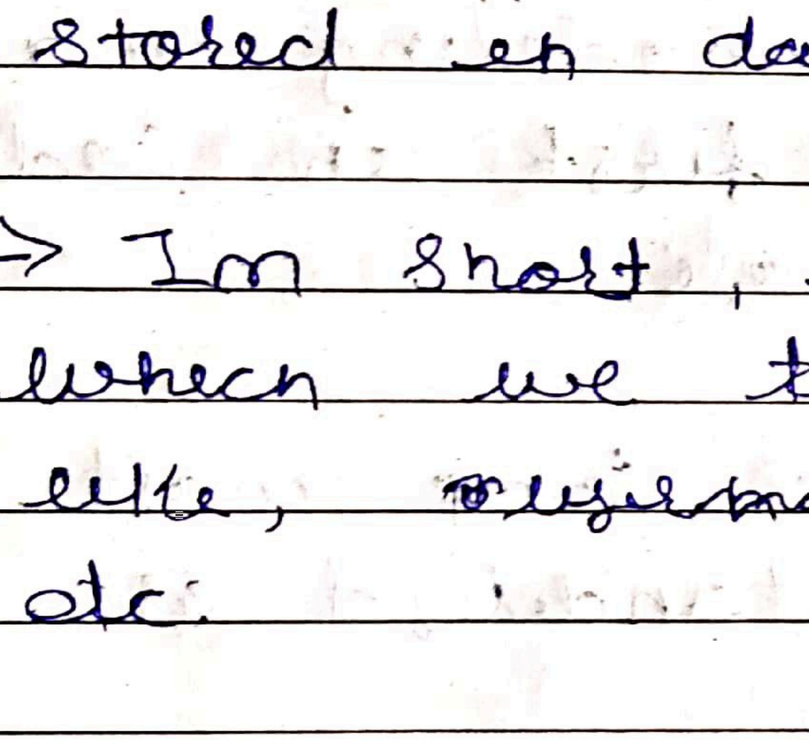
⇒ For Free. erazer.io we can use for data modeling.

TIPS: whenever we some build first we have to decide to How we save data:

Ex: what we first make login or Register
→ login is just validation to match stored in data.

→ In short, first we have decide which we take field required. So title, username, Pass, DOB, photo.. etc.

⇒ FOR example; Google idea of TODOS



For outer box: we need.

title
color {
subTodos: [{}, {}, {}]

For inner Todos:

content
markedAsDone
Created date.

Page No.
Date

⇒ So, first we make modeling of the our & API then we use the mongoose, prisma other for storing data.

Tips: It's called helper which is help to store the data. (mongoose, prisma)

⇒ Code phase :-

⇒ After deciding modeling make on making models first in code with help of mongoose.

→ It's file but most popular conventions and standerized are as follow.

user.js ✗ user.model.js ✓

→ name convention bcoz it is model file so we write it as. (Just best practices)

same

user.controller.js ✓

Page No.
Date

=> for creating model schema in mongoose. (3) line are main

~~import~~

import mongoose from "mongoose"

const userSchema = new mongoose.Schema({})

export const User = mongoose.model("User", userSchema);

=> Right way to define field in Schema.

{	{
username: String,	username: {
email: String	type: String,
password: String	required: true,
isActive: Boolean	unique: true
}	lowercase: true
	}
	}

=> Also, they provide Validation in Schema go to Docs → Validation.

=> TimeStamp is most common thing in Schema that's why mongoose made it easy to just pass object {timestamps: true} in Schema after

defining all fields. So it auto matically add createdAt & updatedAt field added.

⇒ Imp in Schema.

createdBy: { ← its type
    type: mongoose.Schema.Types.ObjectId ← of ref
    ref: "User"
}
↑
name of ref database schema model name.

.

.

SubTodos: [ ] (acceptable)

but

SubTodos: [
    {
        type: mongoose.Schema.Types.ObjectId
        ref: "SubTodo"
    }
]

⇒ There is we can also make schema in the same file.
ex: orderItemSchema

⇒ also we can define enum for choices, or the only given value.

⇒ default also property there.

5] Set up a Professional backend Project

→ 100% Production grade
→ most Company's Standerize aRRoach.

→ git track only files, So when we create folder or inner folder without any files so, we can't Push that folder.

→ Foder Structure is properly follow by the company according its requirment but mostly are as discussed & other are follows.

=> Prettier: code beauteby & formatter.
- it required because we not only work but in company there is people which work on same Project so, it's other defined How much spaces, semicolon are we keep in our all code.

→ it's setting other defined by Project manager or head for code consistency.