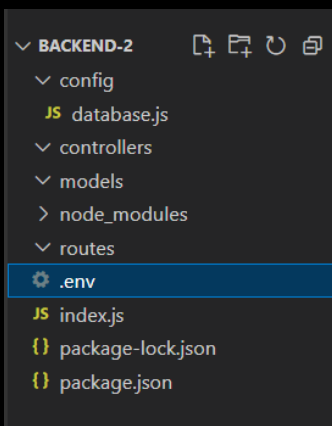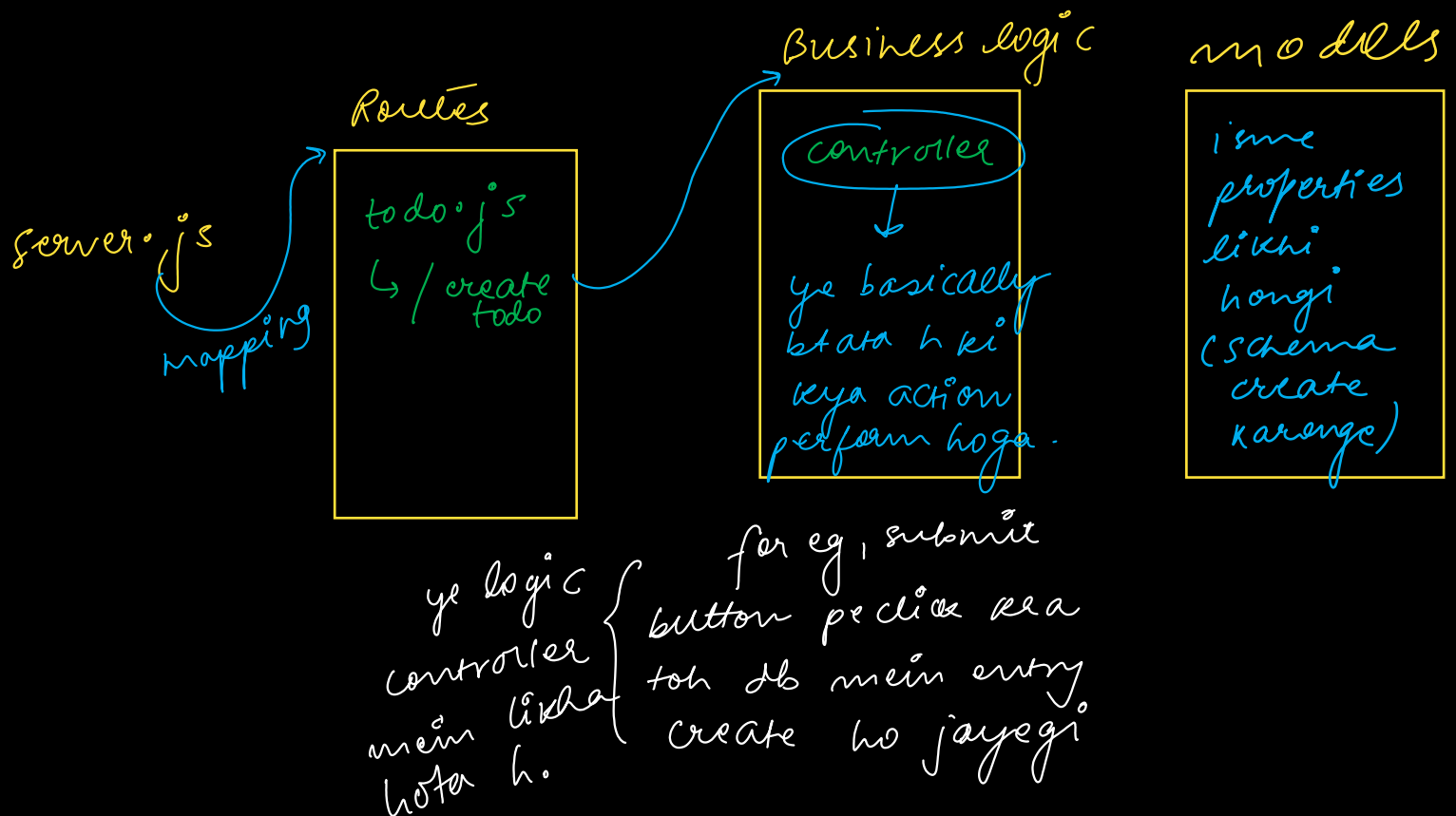# (#) Backend L-2

If there are 100's of routes, we can not create all of them inside the the same server.js file,

so to improve the readability, we seperate things inside different folders.

Routes

Business logic

models

server.js

mapping

todo.js
↳ /create todo

controller

ye basically btata h ki kya action perform hoga.

isme properties likhi hongi (schema create karenge)

ye logic controller mein likha hota h. { for eg, submit button pe click kea toh db mein entry create ho jayegi

```
∨ BACKEND-2
  ∨ config
    JS database.js
  ∨ controllers
  ∨ models
  > node_modules
  ∨ routes
    ⚙ .env
  JS index.js
  {} package-lock.json
  {} package.json
```

created all the required folders
There is a seperate file called as .env
↓
This stores the environment variable

```
app. listen (3000, () => { })
```

Bad practice h ye,

So instead we specify those values inside our
.env files, and then acces it inside other files

for this we, need to install dotenv package

```
npm i dotenv
```

```
JS index.js    JS database.js    ⚙ .env    ✕    {} package.json
⚙ .env
    1   PORT = 3000
    2   DATABASE_URL = mongodb://127.0.0.1:27017/naugsDataBase
```

name of the datab-
ase

This is same as
localhost

inside our
.env file we've
specified the
variables,

```
> JS database.js > [∅] dbConnect
1   const mongoose = require("mongoose");
2
3   const dbConnect = () =>{
4       mongoose.connect(process.env.DATABASE_URL)
5   }
```

we've accessed
mongoose, and
now through this
function we're
establishing the
connection b/w index.js
and the database

already installed
the .env package,
now accessing the
values.

```js
const mongoose = require("mongoose");
require("dotenv").config();

const dbConnect = () =>{
    mongoose.connect(process.env.DATABASE_URL, {
        useNewUrlparser:true,
        useUnifiedTopology:true,
    })
    .then(() => console.log("DB ka connection is successfull"))
    .catch( (error) => {
        console.log("Issue in DB connection")
        console.log(error.message);
        process.exit(1);
    });
}

module.exports = dbConnect;
```

jo bhi kuch environment ke andar define kiya gya hoga, vo sb kuch load ho jayega process ke andar.

This is how connection is made b/w the index.js and database.

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

const todoSchema = new mongoose.Schema(
    {
        title:{
            type:String,
            required:true,
            maxLength:50
        },
        description:{
            type:String,
            require:true,
            maxLength:50
        },
        createdAt:{
            type:Date,
            require:true,
            default:Date.now(),

        },
        updatedAt:{
            type:Date,
            require:true,
            default:Date.now(),
        }
    }
)
```

This is a schema

for a to-do app, we've specified the properties

properties

```js
module.exports = mongoose.model("Todo", todoSchema);
```

exported the schema
and now can be used
with the name Todo.

→ ab iske baad jo important cheez hoti h,
  vo h controller,
→ Joh jo v path create krte h humlog, unn sab
  ka controller hota h.
→ jaise humlog project mein todo app bna rhe h,
  toh create todo ka ek controller banayenge
  i.e todo type ke ek object ki entry database
  mein store krna chahta he

final architecture :↑