

## TypeScript-Simple-Mern-Project\thapaMernCodeno\_33.jsx

```
1 Welcome, to MERN series as we code our service page backend. Learn about Schema,
  Model, Route, and Controller, and get hands-on with Atlas.
2
3 Here is the our Service API
4
5 ```json
6 [
7   {
8     "service": "Web Development",
9     "description": "Crafting tailor-made websites and web applications.",
10    "price": "$1,500 - $7,000",
11    "provider": "Thapa Technical Youtube Inc."
12  },
13  {
14    "service": "E-commerce Website Development",
15    "description": "Building powerful e-commerce websites for your business.",
16    "price": "$2,000 - $8,000",
17    "provider": "Thapa Technical Youtubec."
18  },
19  {
20    "service": "Responsive Web Design",
21    "description": "Creating visually stunning and responsive websites.",
22    "price": "$1,200 - $6,000",
23    "provider": "Thapa Technical Youtube."
24  },
25  {
26    "service": "Mobile App Development",
27    "description": "Developing innovative and user-friendly mobile applications.",
28    "price": "$2,500 - $10,000",
29    "provider": "Thapa Technical Youtube."
30  },
31  {
32    "service": "WordPress Website Development",
33    "description": "Building dynamic websites using the WordPress platform.",
34    "price": "$1,300 - $5,500",
35    "provider": "Thapa Technical Youtube"
36  },
37  {
38    "service": "UI/UX Design Services",
39    "description": "Crafting intuitive and user-centric UI/UX designs for your
40    projects.",
41    "price": "$1,800 - $7,500",
42    "provider": "Thapa Technical Youtube."
43  }
44 ]
45 ```
46 1. Let's build the services model.
47
48   ...
49   const { Schema, model } = require("mongoose");
50
51   const serviceSchema = new Schema({
52     service: { type: String, required: true },
```

```

53     description: { type: String, required: true },
54     price: { type: String, required: true },
55     provider: { type: String, required: true },
56   });
57
58   const Service = new model("Service", serviceSchema);
59
60   module.exports = Service;
61   ```
62

```

2. Create a route **and** add the controllers file **for** handling services.

```

64   ```
65
66   const express = require("express");
67   const router = express.Router();
68
69   router.route("/service").get(services);
70   module.exports = router;
71
72   //after
73
74   const express = require("express");
75   const router = express.Router();
76   const services = require("../controllers/service-controller");
77
78   router.route("/service").get(services);
79   module.exports = router;
80   ```
81

```

3. Define the logic to retrieve services data on our services page when users visit the service route.

```

83   ```
84
85   const Service = require("../models/service-model");
86
87   const services = async (req, res) => {
88     try {
89       const response = await Service.find();
90       if (!response) {
91         // Handle the case where no document was found
92         res.status(404).json({ msg: "No service found" });
93         return;
94       }
95       return res.status(200).json({ msg: "Service found", data: response });
96     } catch (error) {
97       console.log(`error from the server ${error}`);
98     }
99   };
100
101   module.exports = services;
102   ```

```