**Food App (Backend)**

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

***Create Backend folder inside Project Folde***r

Open terminal

*npm init*

*Entry point server.js*

***Create server.js in backend folder***

***Now install dependencies***

***------------------------------------------------------------------------------------------------------------------------------------------***

npm install express mongoose jsonwebtoken bcrypt cors dotenv body-parser multer stripe validator nodemon

***------------------------------------------------------------------------------------------------------------------------------------------***

***Mongoose -*** use for database

***Jsonwebtoken -*** use for create authentication system

***Bcrypt -*** encript the user data and store in the database

***Cors -*** frontend and backend url access

***Dotenv -*** create environment variable

***body-parser -*** perse the data coming through the user

***Multer -*** image store system

***Stripe -***  payment detway

***Validator -*** username and password validate

**Nodemon** - auto restart if any changes done and save

***In package.json:***

***------------------------------------------------------------------------------------------------------------------------------------------***

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1" // remove this

"server" : "nodemon server.js" // add this

  },

***------------------------------------------------------------------------------------------------------------------------------------------***

***Folder structure:***

Create:

**config:** store configaration files link Database config file

**models:** Store database models

**middleware:**

**controllers**: logic of backend

**routes**:

**uploads**:

***Create .env*** -> where we can store all environment variables

To create module type server: Open ***package.json:***

***------------------------------------------------------------------------------------------------------------------------------------------***

 "main": "server.js",

  "type": "module", // add this

***------------------------------------------------------------------------------------------------------------------------------------------***

***Open server.js file:***

***------------------------------------------------------------------------------------------------------------------------------------------***

// create a basic express server

import express from "express";

import cors from "cors";

// app config

const app = express();

const port = 4000;

// middleware [communicate between backend and Database]

app.use(express.json()); //

app.use(cors()); // access backend from any frontend

// create route

// get() is a http method which can use for request data from server

app.get("/", (req, res) =>{

    res.send("API working"); // get response from api end point

});

// run the express server

// `` -> this is called "Template Literals"

app.listen(port, ()=>{

    console.log(`Server is runing on port - http://localhost:${port}`)

});

***------------------------------------------------------------------------------------------------------------------------------------------***

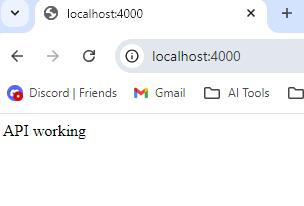
**Run code command**:

***------------------------------------------------------------------------------------------------------------------------------------------***

npm run server

***------------------------------------------------------------------------------------------------------------------------------------------***

**In browser:**



We will use **Thunder Client** visual code extension for check API

**Connect MongoDB to backend**

Create **db.js** file inside **config** folder

***------------------------------------------------------------------------------------------------------------------------------------------***

import mongoose from "mongoose";

export const connectDB = async () => {

    await mongoose.connect('mongodb+srv://joydipsarkar01:qlkdgd8hqtlrs7ah@foodapp-test.bkuh9.mongodb.net/food-app').then(() => console.log("DB Connected"));

}

***------------------------------------------------------------------------------------------------------------------------------------------***

To access db connection in **server.js**, open **server.js** file

***------------------------------------------------------------------------------------------------------------------------------------------***

After middleware section

// db connection

connectDB();

\*\* it will imported the path at top

import { connectDB } from "./config/db.js";

***------------------------------------------------------------------------------------------------------------------------------------------***

**So server.js full code:**

***------------------------------------------------------------------------------------------------------------------------------------------***

import express from "express";

import cors from "cors";

***import { connectDB } from "./config/db.js";***

// app config

const app = express();

const port = 4000;

// middleware [communicate between backend and Database]

app.use(express.json()); //

app.use(cors()); // access backend from any frontend

***// db connection***

***connectDB();***

// create route

// get() is a http method which can use for request data from server

app.get("/", (req, res) =>{

    res.send("API working"); // get response from api end point

});

// run the express server

// `` -> this is called "Template Literals"

app.listen(port, ()=>{

    console.log(`Server is runing on port - http://localhost:${port}`)

});

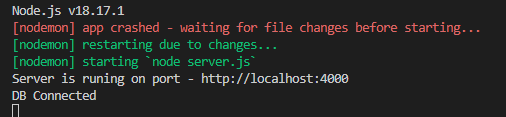
***------------------------------------------------------------------------------------------------------------------------------------------***

**Run code command**:

***------------------------------------------------------------------------------------------------------------------------------------------***

npm run server

***------------------------------------------------------------------------------------------------------------------------------------------***



**Now create models for store data in database**

***------------------------------------------------------------------------------------------------------------------------------------------***

import mongoose from "mongoose";

const foodSchema = mongoose.Schema({

    name: { type:String, required:true },

    description: { type:String, required:true },

    category: { type:String, required:true },

    image: { type:String, required:true },

    price: { type:Number, required:true },

});

const foodModel = mongoose.models.food || mongoose.model("food", foodSchema)

// note: mongoose.models.food -> [this will not create another model when this page rerender] || mongoose.model("food", foodSchema) -> this will create model when page is refresh

export default foodModel;

***------------------------------------------------------------------------------------------------------------------------------------------***

**Create API for add new food item in database**

***------------------------------------------------------------------------------------------------------------------------------------------***

First we will create Model for store in database:

Create **foodModel.js** in **models** folder

**foodModel.js**

***------------------------------------------------------------------------------------------------------------------------------------------***

import mongoose from "mongoose";

const foodSchema = mongoose.Schema({

    name: { type:String, required:true },

    description: { type:String, required:true },

    category: { type:String, required:true },

    price: { type:Number, required:true },

    image: { type:String, required:true },

});

const foodModel = mongoose.models.food || mongoose.model("food", foodSchema)

// note: mongoose.models.food -> [this will not create another model when this page rerender] || mongoose.model("food", foodSchema) -> this will create model when page is refresh

export default foodModel;

***------------------------------------------------------------------------------------------------------------------------------------------***

Create **foodController.js** in **controller** folder for Add food request

***------------------------------------------------------------------------------------------------------------------------------------------***

import foodModel from "../models/foodModel.js";

import fs from "fs";

// add food

const addFood = async (req, res) => {

    // we will create route for endpoint

    let image\_filename = `${ req.file.filename }`;

    const food = new foodModel({

        name: req.body.name,

        description: req.body.description,

        category: req.body.category,

        price: req.body.price,

        image: image\_filename,

    })

    try {

        await food.save();

        res.json({ success:true, message:"Food added" })

    } catch (error) {

        console.log(error);

        res.json({ success:false, message:"error on save" })

    }

}

export { addFood }

***------------------------------------------------------------------------------------------------------------------------------------------***

Create **foodRoute.js** for create food router in **router** folder

***------------------------------------------------------------------------------------------------------------------------------------------***

import express from "express";

import { addFood } from "../controllers/foodController.js";

import multer from "multer"; // for image store system

// create express router. using this router we can create get, post method

const foodRouter = express.Router();

// img store engine

const foodImgStorage = multer.diskStorage({

    destination: "uploads",

    filename: (req, file, callback) => {

        return callback(null, `${Date.now()}${file.originalname}`);

    }

})

const uploadFoodImg = multer({ storage:foodImgStorage })

// craete a post method

//foodRouter.post("/add", addFood); // without image

foodRouter.post("/add", uploadFoodImg.single("image"), addFood);

// export router

export default foodRouter;

// note: for create api endponter we will go to server.js

***------------------------------------------------------------------------------------------------------------------------------------------***

Now we need to create main api url endpoint for all type of methods related to food (post, get).

For that need to open **server.js**

***------------------------------------------------------------------------------------------------------------------------------------------***

import foodRouter from "./routes/foodroute.js";

// api main endpoint for foodRouter

app.use("/api/food", foodRouter);

***------------------------------------------------------------------------------------------------------------------------------------------***

***So the updated server.js code below***

***------------------------------------------------------------------------------------------------------------------------------------------***

import express from "express";

import cors from "cors";

import { connectDB } from "./config/db.js";

**import foodRouter from "./routes/foodroute.js";**

// app config

const app = express();

const port = 4000;

// middleware [communicate between backend and Database]

app.use(express.json()); //

app.use(cors()); // access backend from any frontend

// db connection

connectDB();

**// api endpoint for foodRouter**

**app.use("/api/food", foodRouter);**

// create route

// get() is a http method which can use for request data from server

app.get("/", (req, res) =>{

    res.send("API working"); // get response from api end point

});

// run the express server

// `` -> this is called "Template Literals"

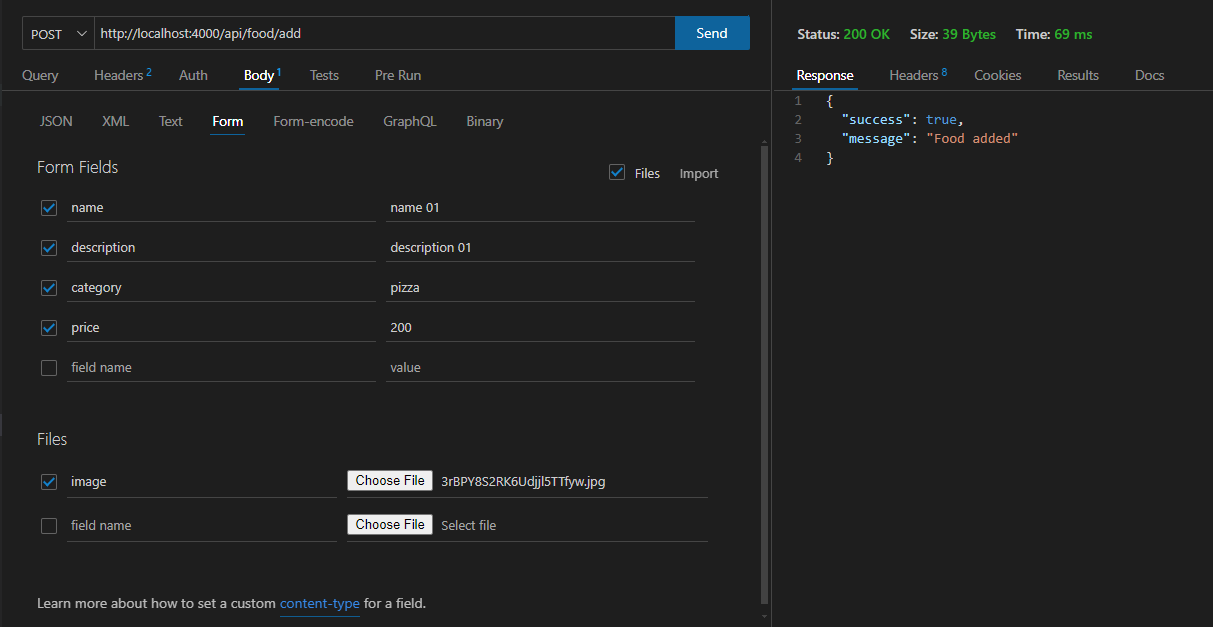
app.listen(port, ()=>{

    console.log(`Server is runing on port - http://localhost:${port}`)

});

***------------------------------------------------------------------------------------------------------------------------------------------***

***For check the api open thunder client***



**In MongoDB**

