API design

get(“/restaurants”, getAllRestaurant) //get all restaurant

get(“/restaurants/:id”, getRestaurant) // get a specific restaurant

post(“/restaurant”, createRestaurant) // create a new restaurant

delete(“/restaurant/:id”, deleteRestaurant) //create a specific restaurant

patch(“/restaurant/:id”, updateRestaurant) // update a specific restaurant

get(“/user”, getAllUsers) //get all user

post(“/user”, signup) //sign up for new user

restaurant schema

{

name:{

type: String,

require: [true, ”a restaurant must have a name”],

trim: true,

unique: true,

maxLength:[40, “a restaurant’s name has to be less than or equal to 40 characters”],

minLength: [5, ”a restaurant’s name has to be greater than or equal to 5 characters”]

},

address:{

type: String,

require: [true, “a restaurant must have an address”],

trim: true

},

phoneNumber:{

type: Number,

require: [true, “a restaurant must have a phone number”],

},

imageCover:{

type: String,

require: [true, “A restaurant must have a main image”],

trim: true

},

website:{

type: String,

trim: true

},

openHours:{

type: Map,

of: String,

default: {

Sunday: ”8AM – 8PM”,

Monday: ”8AM – 8PM”,

Tuesday: ”8AM – 8PM”,

Wednesday: ”8AM – 8PM”,

Thursday: ”8AM – 8PM”,

Friday: ”8AM – 8PM”,

Saturday: ”8AM – 8PM”,

}

},

rating:{

type: Number,

default: 0

},

ratingQuanity:{

type: Number,

default: 0

} ,

images:[String],

comments: [{

username: {type: string, require: true, trim: true},

comment: {type: string, require: true},

rate: {type: Number, require: true}

}]

}

User schema

{

name:{

type: String,

require: [true, “User must have a name”],

trim: true

},

emai: {

type: String,

require: [true, “user must have an email”],

trim: true

},

photo: String,

role:{

type: String,

enum: [“user”, “admin”],

default: “user”

},

password:{

type: String,

require: [true, “user must have password”],

trim: true,

select: false,

minLength: 8

},

passwordConfirm:{

type: String,

require: [true, “please confirm your password”],

validate:{

validator: function(el){

return el == this.password

},

message: “password are not the same”

}

}

}

Functionality:

* can see all of the restaurant:
  + server: API call get(“/restaurants”) to return an array of restaurant.
  + client: see all of the restaurant in the home page.
* Can see detail of a specific restaurant
  + Server: API call get(“/restaurants/:id”) to get a specific restaurant
  + Client: when user click on an item in the home page move them to the detail page and fetch the API
* Can add new restaurant
  + Server: API call post(“/restaurant”) to create a new restaurant, make sure user login before create new restaurant, and the restaurant name is unique
  + Client: when user click the plus symbol in the homepage, check if user already login or not, if not ask user to, otherwise, show the form for user to create new restaurant, when user click submit button in the form, call find function to make sure the new restaurant hasn’t existed in the database, finally refresh the page with the new restaurant
* Login
  + Server API call post(“/login”) to post the username and password to confirm
  + Client: when user click the login button in the home page, show a form for user to login, when user click the login button in the form, if username and password are correct close the login form and show the icon and username in the homepage
* Signup
  + Server API call post(“/signup”) to post the username and password to create new user
  + Client: when user click the signup button in the homepage, show the form for user to create their account, when user click the signup button in the form if nothing wrong close the form and show icon and username in the homepage
* Can add rate and comment to a restaurant, which use patch to update
  + Server API call patch(“/restaurant/:id”) to update restaurant new rate and new comment and new rate quantity, calculate the new rate by using old rate and old rate quantities, make sure user is login.
  + Client:
* Can search restaurant name
* Can update or remove a restaurant (only when login as admin role)