James Dumitru

ITMD 462

Final Project

For this project, I attempted to make a web app that would be the foundation for a future store I would implement later on in the future to start a local business. I attempted to have CRUD implementations for Users when logging into the website. I did run into a bit of trouble with the CRUD but I will get to that later. I attempted the 3d aspects of obtaining CAD files but I needed more time to understand how it works to view because I could make a simple one view but wasn’t what I wanted exactly. I changed my direction and started with the mongo db.

If you’re starting this project, make sure you do ```npm install``` to get all the node-modules because it was a huge zip when I did it.

This shows the Cluster0 being deployed and operational.

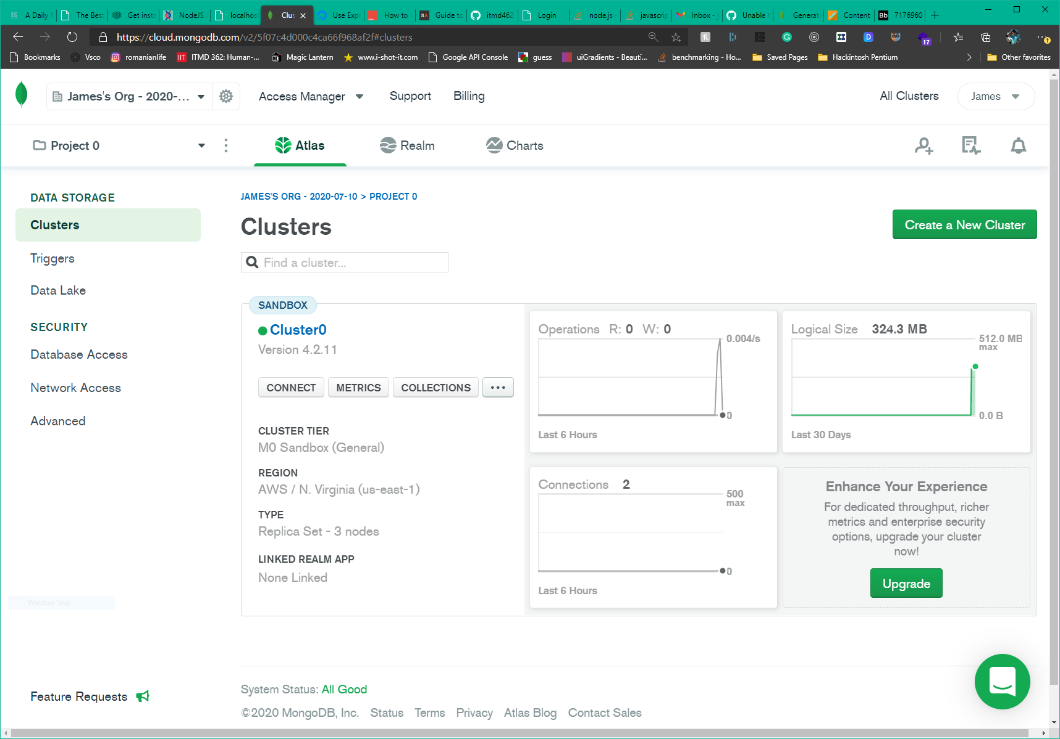


Figure Cluster

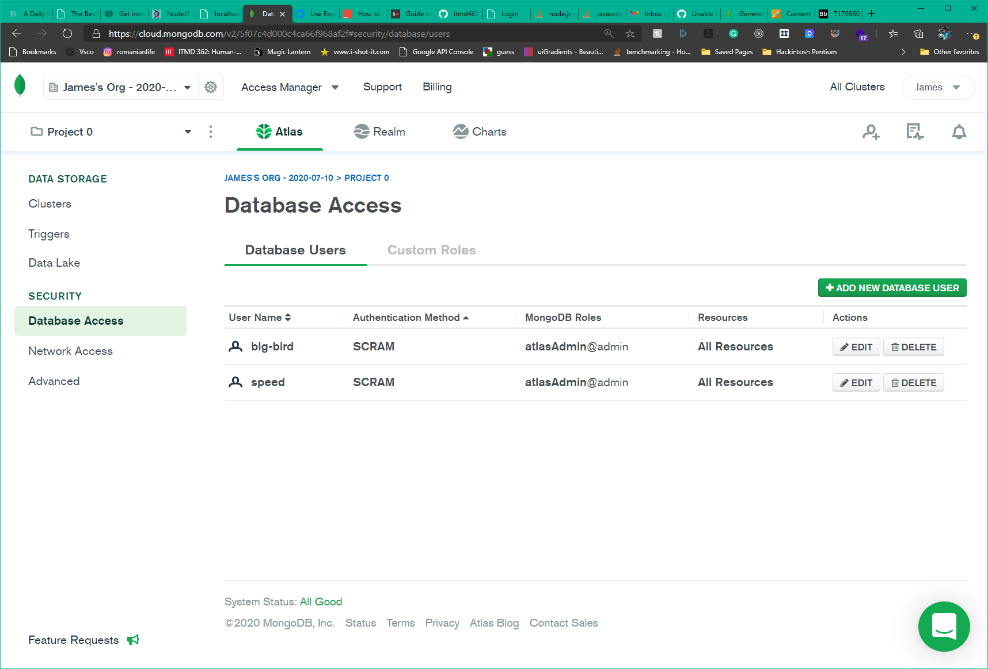
At first, I maintained the connection from my IP address as a test, then created it accessible from anywhere because I figured you wouldn’t be able to run the code if it wasn’t specified to do so. “speed” is the test, while big-bird is the one that’s accessible from anywhere, the password being bigfeet so I could remember.

Figure Database Access

Now the problem I had was understanding with the back end being able to write and to the server. I’m still unsure of how mongo works because I’m used to relational database like in MySQL.

I started with a simple login page and added a bit of color. I started with html but then got into handlebars, so I have a copy of sorts when you view the code and I had separate CSS files for each html page. I had a rough time with accessing the files through node because I found that the naming has to be specific. It wouldn’t access it normally but once you specified an endpoint or a sub-endpoint, and if it was in a folder in the views folder was ‘/layout/login’ . You can view all the endpoints in the server.js file where all the html file names were recreated as an endpoint to reach the handlebar files.

Below is a view of the pages when moved through. Once the connection is done, you should be able to sign up or login and then you can move through myAccount in which you are able to view the stored information of the users account and then have options to edit that information in the following links on the left. You’re also able to view the about to answer future aspects of the code for quotes later on. The other part is when you access the endpoint [localhost:5000/list](http://localhost:5000/user) to view users in store and then [localhost:5000/addOrEdit](http://localhost:5000/addOrEdit) to add a user manually. Edit will come in later development as I needed more time.

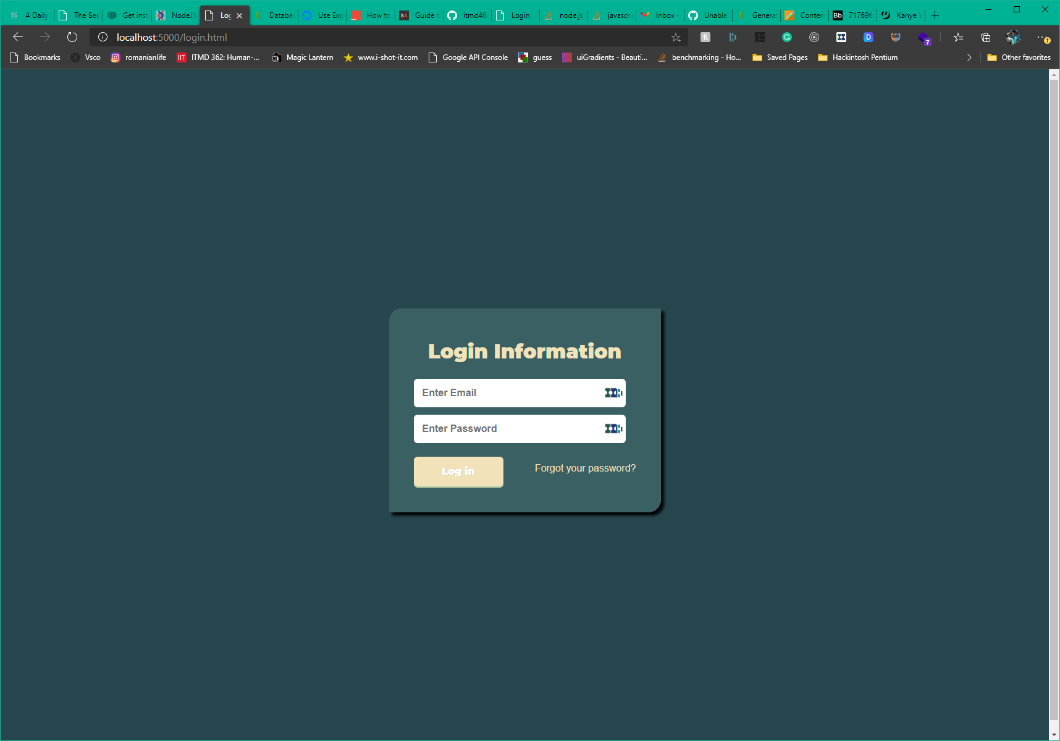


Figure Login

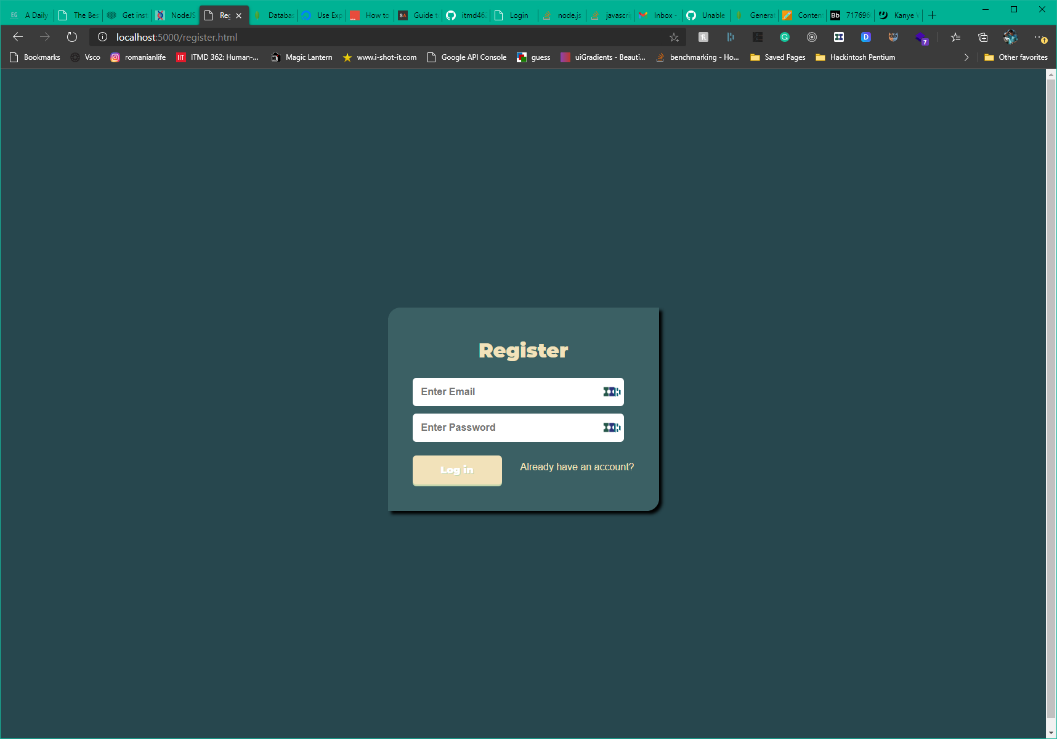


Figure Register

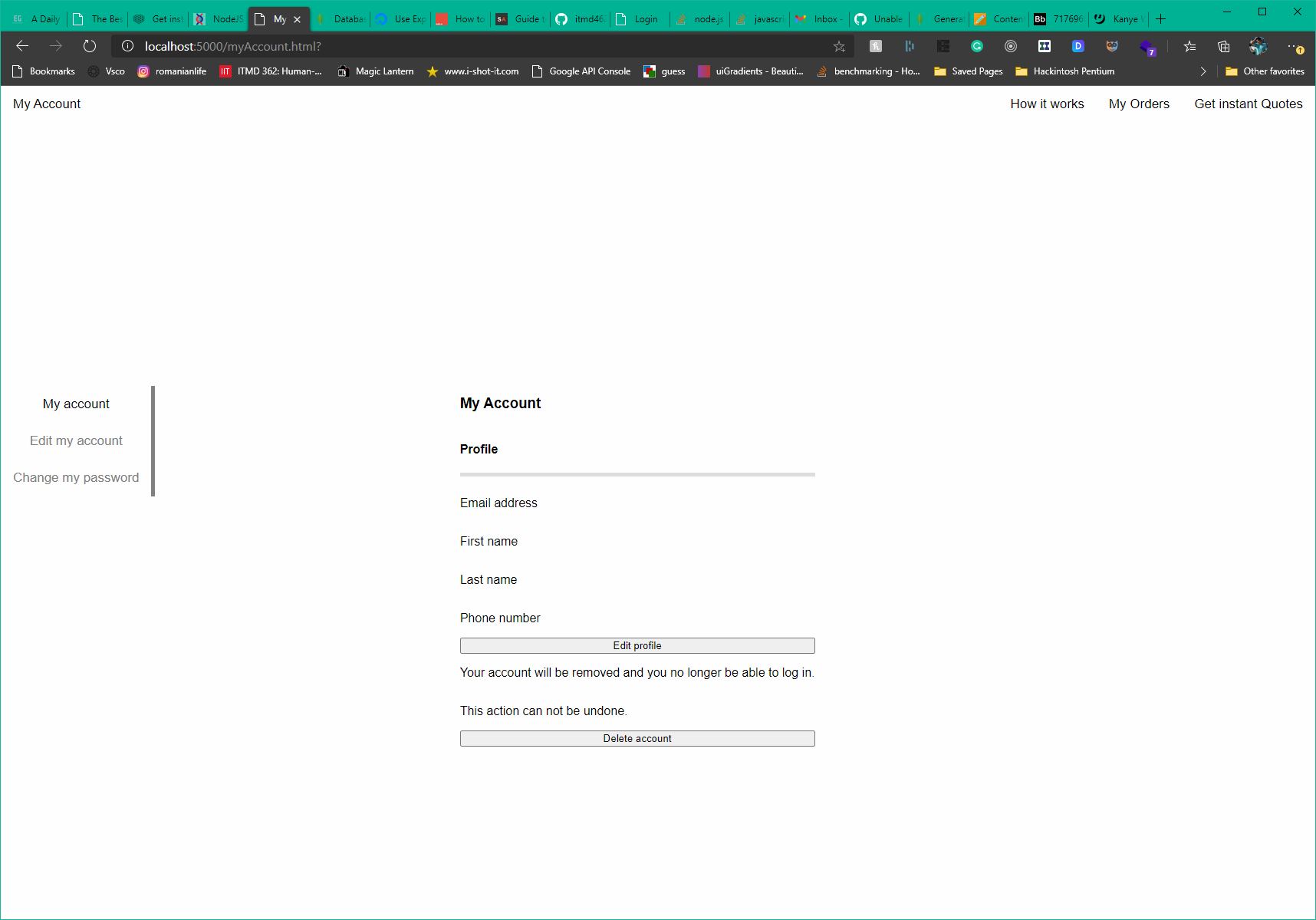


Figure myAccount

For my account I had the plan of displaying the information saved from when you edit the information on the account to display into this page. I had difficulties connecting my routing, however. editMyAccount is where you would save this information and where you would edit it.

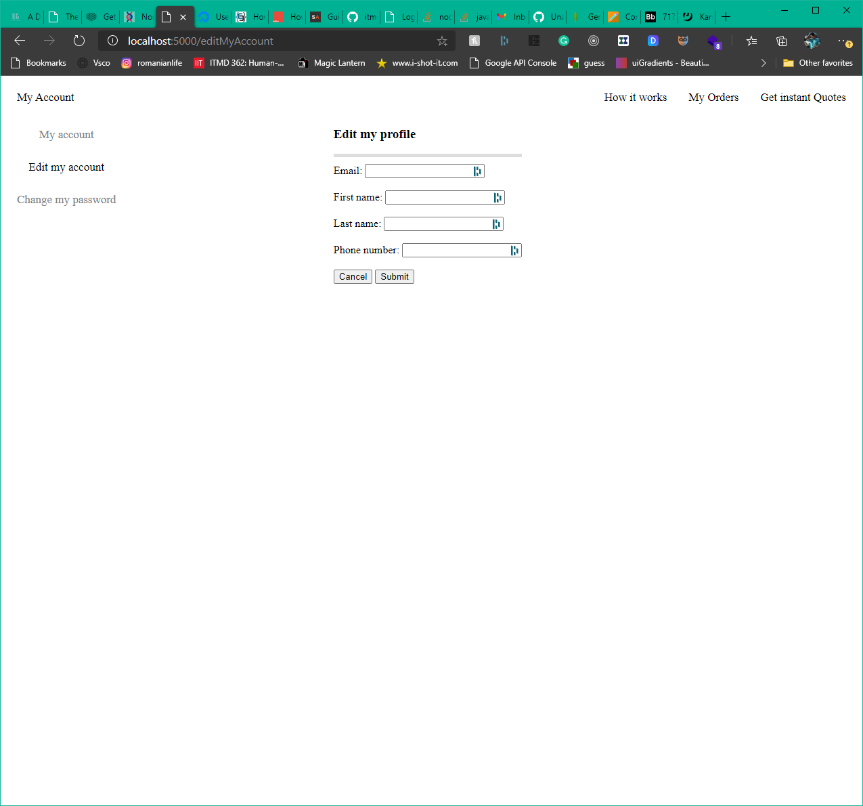


Figure editMyAccount

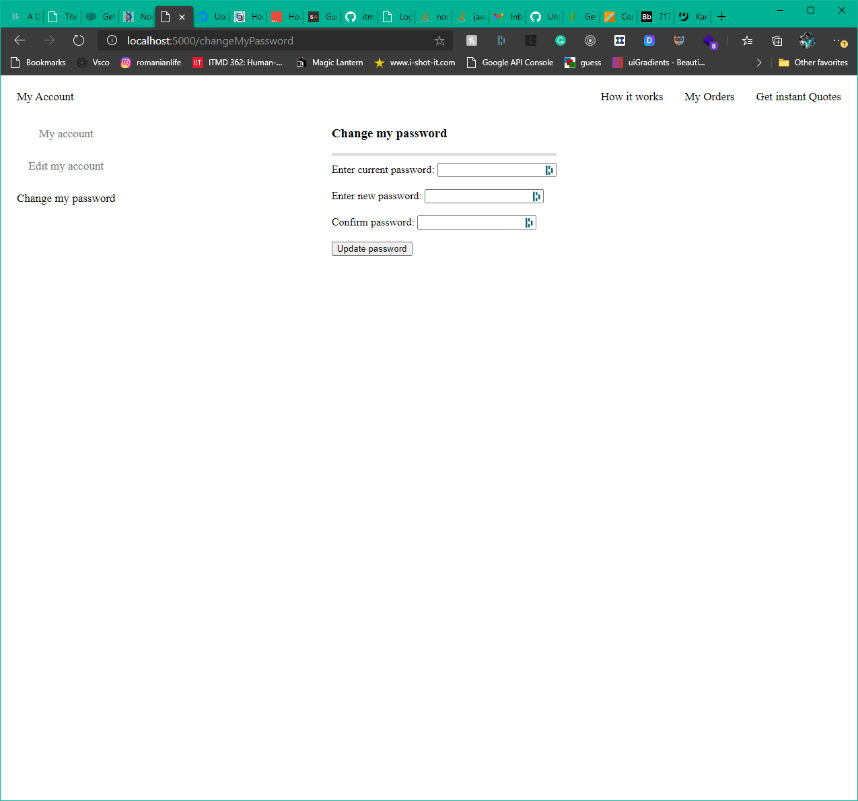


Figure changeMyPassword

The about page would be for when you place an order in the future about how to use the tool and what you need to get an instant quote. In the future I plan on incorporating links to explain what kind of files that you need and connect them to the Get instant Quote page.

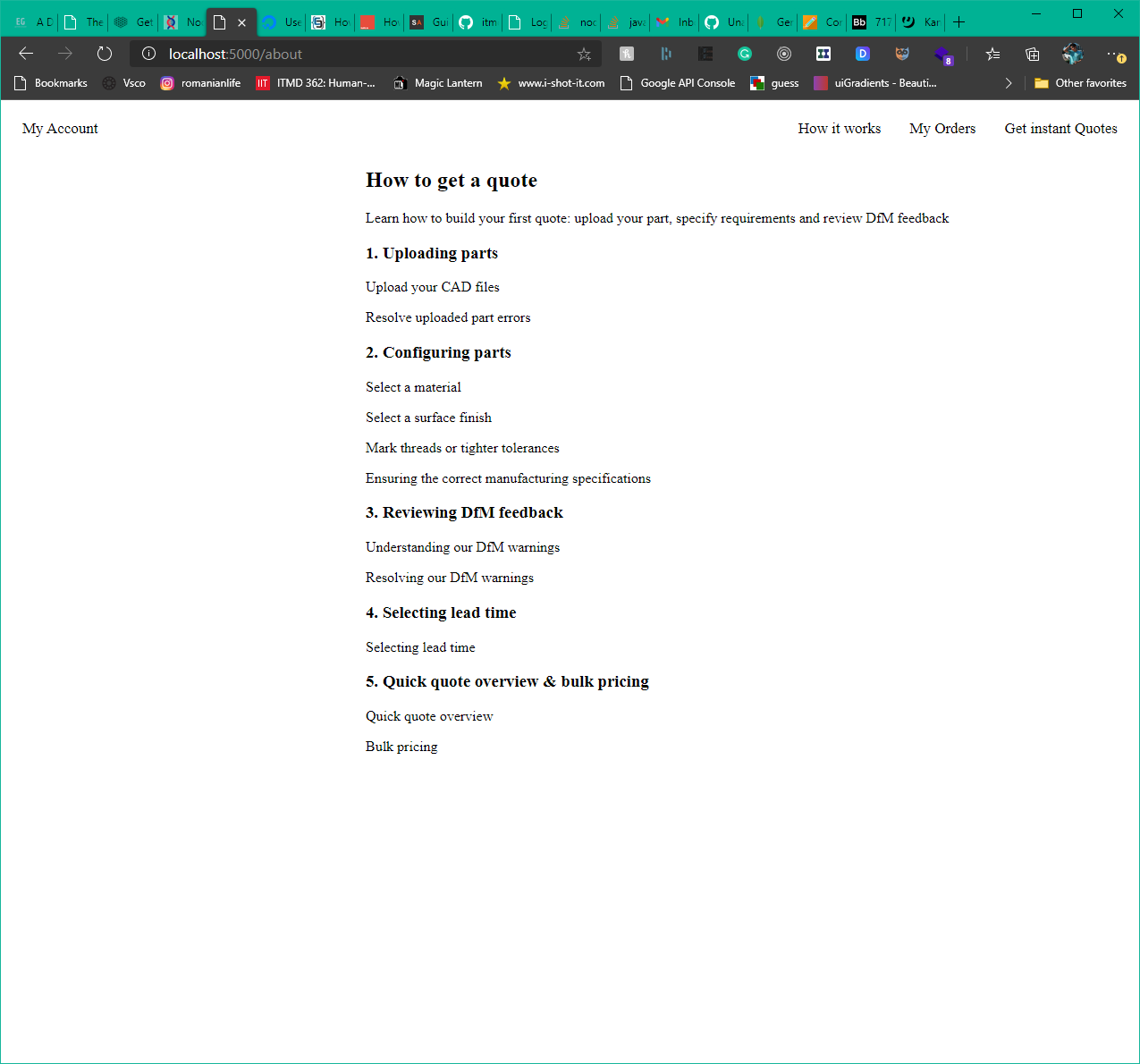


Figure about

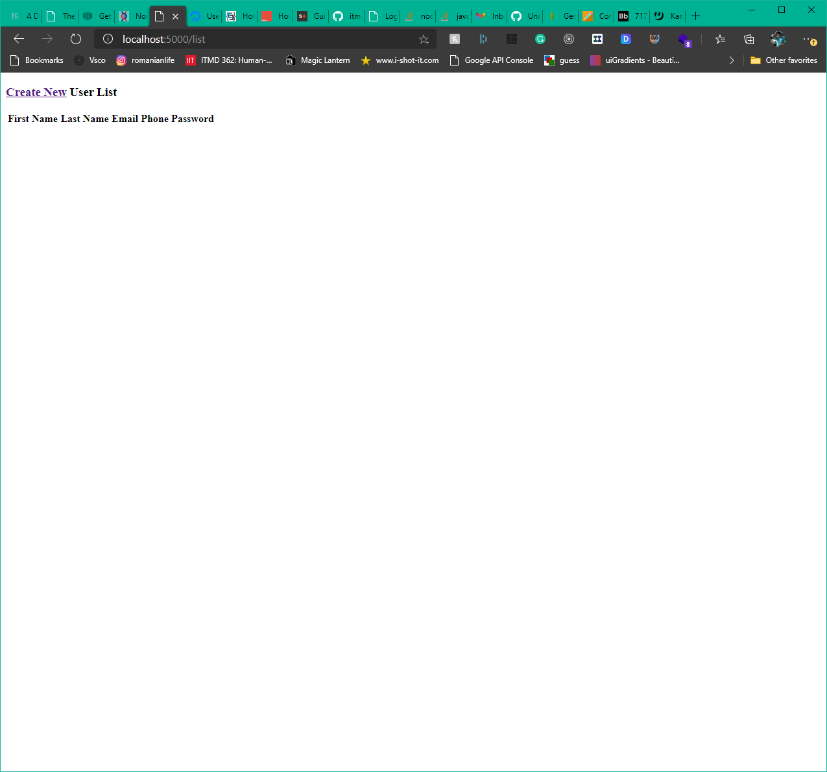


Figure list

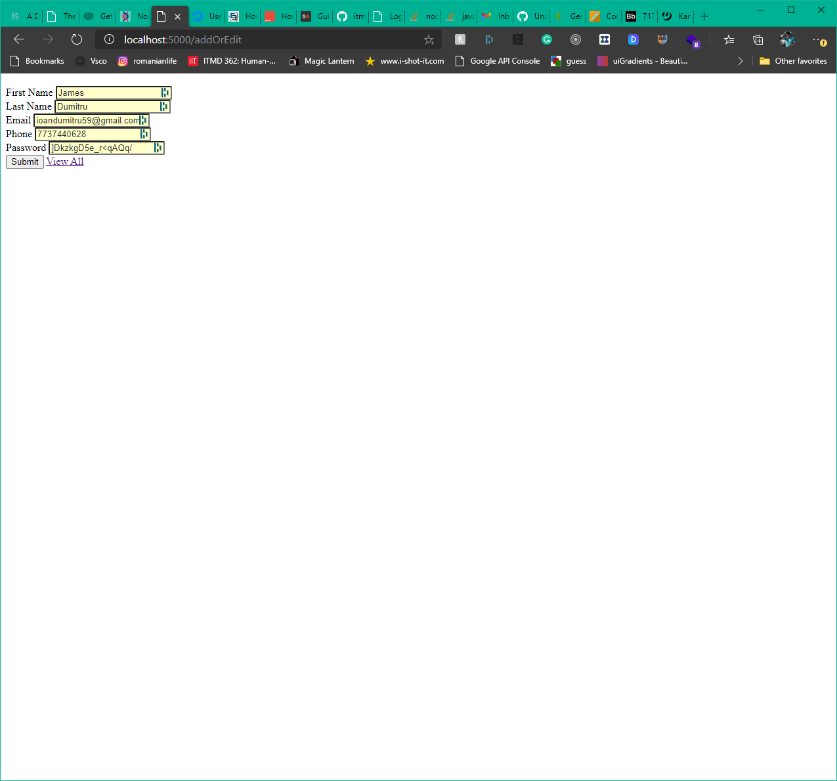
The plan for list was for dev to be able to see the user and manually create the user and view all users attached to the website and written to the mongo db.

Figure addOrEdit

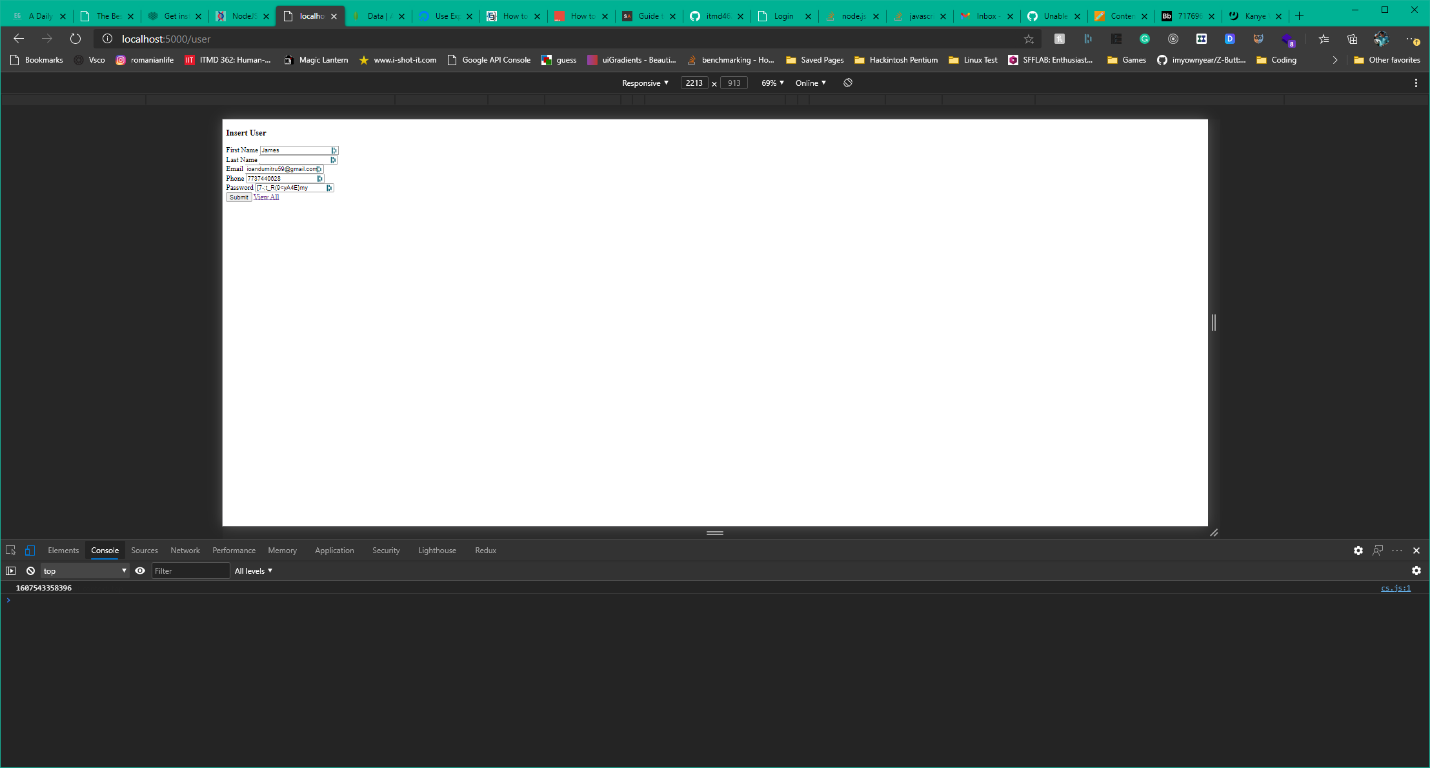
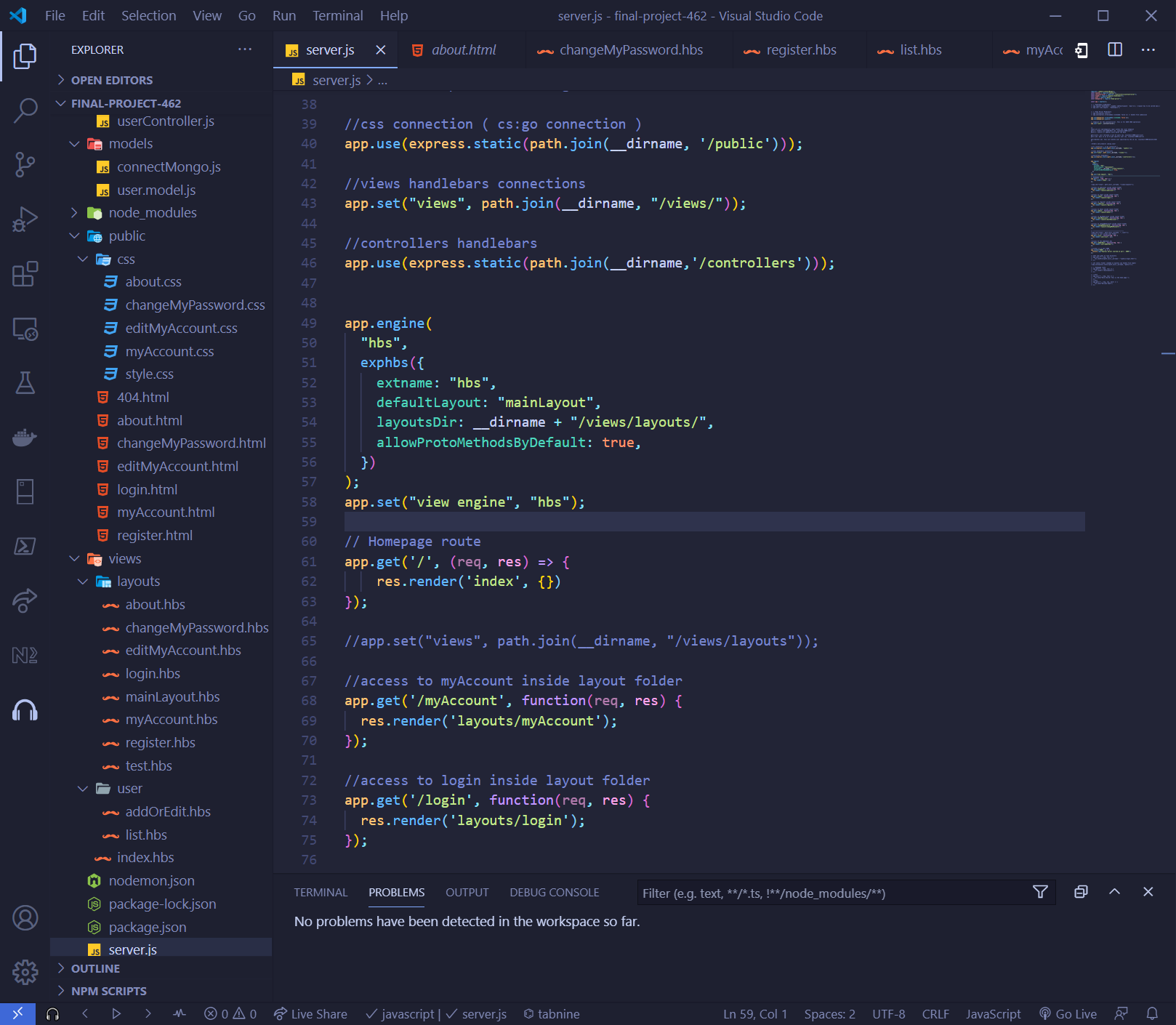
For the addOrEdit handlebar page, I am confused on why it doesn’t incorporate the routing, my guess is I’m not calling it correctly.

Figure id in console log

I get the id to display in console log when I submit and it changes but I’m stuck on how it inserts to mongo and stays saved because I don’t think it’s inserting correctly into mongo when I go check for a database or some type of change.



//server.js main api handler for file connection and handlebars.

require("./models/connectMongo");

const express = require("express");

const userController = require("./controllers/userController");

const exphbs = require("express-handlebars");

const path = require("path");

const bodyparser = require("body-parser");

const app = express();

// // Handlebars middleware

// app.engine('handlebars', exphbs( {defaultLayout: 'main'})); //layout has to be called main.layout

// app.set('view engine', 'handlebars');

// // Body Parser Middleware

// app.use(express.json());

// app.use(express.urlencoded({ extended: false })) // handle form submission

app.use(bodyparser.urlencoded({ extended: false }));

app.use(bodyparser.json());

// endpoint for the userController. This is for USER CRUD operations

app.use("/user", userController);

/\*

these are all sub-endpoints (api) within the /user endpoint

get(/): renders user/addOrEdit (ex. localhost:5000/user/)

post(/): inserts or updates users given the data

get(/list): will retrieve a list of users (ex. localhost:5000/user/list)

get(/:id): pass in an id to get specific user (ex. localhost:5000/user/2)

get(/delete/:id): this will delete user specified by the id (ex. localhost:5000/delete/420)

\*/

//Orders and products coming soon!

//css connection ( cs:go connection )

app.use(express.static(path.join(\_\_dirname, '/public')));

//views handlebars connections

app.set("views", path.join(\_\_dirname, "/views/"));

//controllers handlebars

app.use(express.static(path.join(\_\_dirname,'/controllers')));

app.engine(

  "hbs",

  exphbs({

    extname: "hbs",

    defaultLayout: "mainLayout",

    layoutsDir: \_\_dirname + "/views/layouts/",

    allowProtoMethodsByDefault: true,

  })

);

app.set("view engine", "hbs");

// Homepage route

app.get('/', (req, res) => {

    res.render('index', {})

});

//app.set("views", path.join(\_\_dirname, "/views/layouts"));

//access to myAccount inside layout folder

app.get('/myAccount', function(req, res) {

  res.render('layouts/myAccount');

});

//access to login inside layout folder

app.get('/login', function(req, res) {

  res.render('layouts/login');

});

//access to register inside layout folder

app.get('/register', function(req, res) {

  res.render('layouts/register');

});

//access to about inside layout folder

app.get('/about', function(req, res) {

  res.render('layouts/about');

});

//access to editMyAccount inside layout folder

app.get('/editMyAccount', function(req, res) {

  res.render('layouts/editMyAccount');

});

//access to changeMyPassword inside layout folder

app.get('/changeMyPassword', function(req, res) {

  res.render('layouts/changeMyPassword');

});

// ------------------------------------------

//app.set("views", path.join(\_\_dirname, "../user"));

//access to list inside user folder

app.get('/list', function(req, res) {

  res.render('user/list', {});

});

//access to the addOrEdit.hbs

app.get('/addOrEdit', function(req, res) {

  res.render('user/addOrEdit');

});

//listens in on port 5000

app.listen("5000", () => {

  console.log("Express Server started at port : 5000");

});

// when user goes to root directory

// app.use("/", (req, res) => {

//   res.sendFile(path.join(\_\_dirname + "/public/login.html"));

// });

// set static folder (needed so express can handle file types)

//app.use(express.static(path.join(\_\_dirname, "public")));

// // Homepage route

// app.get("/", (req, res) => {

//   res.render("/mainLayouts");

// });

// //test

// app.use("/", (req, res) => {

//   res.send("Hello World! This is the home page.");

// });

// //test

// app.get("/", (req, res, next) => {

//   res.send("Welcome Home");

// });