**FRONTEND:**

Index.html:

<!-- Root index.html file -->

<!DOCTYPE html>

<html lang="en">

  <head>

    <script

   src="<https://maps.googleapis.com/maps/api/js?key=AIzaSyAX3ogg6uXEyBxm_OyGGhvv9Z6hUr6yKts&libraries=places&callback=initMap>">

    </script>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta

      name="description"

      content="Find your favorite foods on Purdue's campus"

    />

    <link

      href="<https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700;900&display=swap>"

      rel="stylesheet"

    />

    <link

      rel="stylesheet"

      href="<https://fonts.googleapis.com/icon?family=Material+Icons>"

    />

    <style>

      \* {

        margin: 0;

        font-family: "Roboto", sans-serif;

      }

    </style>

    <title>Titan Health App</title>

    <link rel="icon" type="image/x-icon" href="/favicon.png" />

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

CORECINFO.CSS

/\* Styling for corec info block \*/

.corecinfo {

    width: 100vw;

    height: 100vh;

    background: linear-gradient(to bottom,

            rgba(0, 0, 0, 0, ) 0%,

            rgba(0, 0, 0, 1) 100%),

            url("../../components/titan\_background.jpeg");

    background-size: cover;

    position: relative;

    color: white;

    display: flex;

    align-items: center;

    justify-content: center;

    .header {

        font-weight: bold;

    }

     .text {

        color: white;

    }

     .space {

        margin-top: 5px;

    }

    .filter {

        margin-left: 50px;

        margin-right: -150px;

    }

          .smallListItem {

        font-size: 14px;

    }

          .listItem {

        // font-size: 18px;

        font-weight: normal;

    }

     //for use in stack component (padding is weird)

    .stack {

        margin-left: 25px;

        margin-right: 15px;

        .stackedFilter {

            margin-top: 0px;

            padding-top: 10px;

        }

 }

    .searchBar {

        transform: translate(7.5px, 0%);

        margin-bottom: 10px;

        z-index: 999;

    }

          .customPrefs {

        transform: translate(-40px, 0%);

    }

    .menuItems {

        margin-left: 32.5px;

        transform: translate(10px, 0%);

    }

     .menuTimes {

        // margin-left: -125px;

        transform: translate(25px, 0px);

    }

          .filters {

        transform: translate(-5px, 0px);

    }/\* Makes filter text white \*/

    .MuiSelect-select {

        color: white;

    }

     .sectionTitle {

     margin-top: 25px;

    }

    .evenMoreSpace {

        margin-top: 35px;

    }

           .checkboxes {

        // align-items: center;

        transform: translate(15px, 0);

    }

            .howBusy {

        margin-top: -25px; /\* moves left column up a bit \*/

    }

     .contactInfo {

        font-weight: bold;

        transform: translate(0px, 340px);

        position: absolute;

     .icon {

            transform: translate(0px, 5px);

        }

    .location {

            &:hover {

                cursor: pointer;

           }

        }

    }

           .sectionHeader {

        display: flex;

        justify-content: space-between;

        align-items: center; /\* Ensures vertical alignment \*/

    }

     .menuTitle {

        margin-left: 16px; /\* Adjust as needed \*/

        font-size: larger; /\* Makes the text a bit bigger \*/

        font-size: 24px;

    }

     .ratingHeader {

        text-align: right;

        margin-right: 16px; /\* Adjust as needed \*/

        display: flex;

        flex-direction: column;

        justify-content: center; /\* Centers the content vertically \*/

        font-size: 14px;

    }

    .ratingTitle {

        font-weight: bold;

        font-size: larger; /\* Makes the text a bit bigger \*/

    }

    .ratingSubtitle {

        display: block; /\* Makes the text appear below \*/

        color: grey; /\* Sets the color to grey \*/

        font-size: 12px; /\* Makes the text smaller \*/

        margin-bottom: 2.5px;

    }

}

**BACKEND:**

/\* This file represents the entry point for out node.js application\*/

const mongoose = require("mongoose");

const dotenv = require("dotenv");

const axios = require("axios");

const express = require("express");

const schedule = require("node-schedule");

/\* Create instance of app \*/

const app = express();

/\* Define REST API endpoint routes \*/

const authenticationRoute = require("./routes/auth");

const menuInfoRoute = require("./routes/menuInfo");

const recommendationsRoute = require("./routes/recommendations");

const problemsRoute = require("./routes/problem");

const ratingsRoute = require("./routes/ratings");

const savedRoute = require("./routes/saved");

const usersRoute = require("./routes/users");

/\* Configure .env (hidden env vars) \*/

dotenv.config();

/\* Establish connection to MongoDB \*/

mongoose

    .connect(

        process.env.MONGO\_URL, {

        useNewUrlParser: true,

        useUnifiedTopology: true

    })

    .then(() => console.log("Successfully connected to MongoDB."))

    .catch(err => console.log(err));

/\* Use express middleware to parse requests from frontend \*/

app.use(express.json());

/\* Allow our app instance to use our API endpoints \*/

app.use("/api/auth", authenticationRoute);

app.use("/api/menuInfo", menuInfoRoute);

app.use("/api/problems", problemsRoute);

app.use("/api/ratings", ratingsRoute);

app.use("/api/recommendations", recommendationsRoute);

app.use("/api/saved", savedRoute);

app.use("/api/users", usersRoute);

/\* Have backend server listen on port 8000 on the local host \*/

const PORT = 8000;

app.listen(PORT, async () => {

    console.log(`Backend is running. Listening on port ${PORT}`);

    console.log("Attempting to connect to MongoDB.");

    /\* Uncomment this to immediately parse new dining data on server startup \*/

    // try {

    //     await [axios.post](http://axios.post/)('<http://localhost:8000/api/menuInfo/load'>);

    // } catch (error) {

    //     console.log("ERROR PARSING DINING DATA ON STARTUP: " + error);

    // }

    /\* Uncomment this to immediately reset all users' food trackers on server startup \*/

    // try {

    //     await axios.delete('<http://localhost:8000/api/users/resetTrackers'>);

    // } catch (error) {

    //     console.log("ERROR RESETTING TRACKER AT MIDNIGHT: " + error);

    // }

});

/\* Schedule jobs to run every midnight scheduler uses CRON formatting: <https://crontab.guru/every-night-at-midnight> \*/

schedule.scheduleJob('0 0 \* \* \*', async () => {

    /\* Parse dining data everyday at 12:00 am \*/

    try {

        await [axios.post](http://axios.post/)('<http://localhost:8000/api/menuInfo/load'>);

    } catch (error) {

        console.log("ERROR PARSING DINING DATA AT MIDNIGHT: " + error);

    }

    /\* Reset user's trackers everyday at 12 am \*/

    try {

        await axios.delete('<http://localhost:8000/api/users/resetTrackers'>);

    } catch (error) {

        console.log("ERROR RESETTING TRACKER AT MIDNIGHT: " + error);

    }

});

DINING.JS

/\* Defines Schema in DB for a menu item \*/

const mongoose = require("mongoose");

const schema = new mongoose.Schema(

    {

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

        formalName: { type: String },

        googleID: { type: String }, //google places id for future use

        mealInfo: {type: [Object] }, //{meal name, start time, end time}

        // mealInfo: {type: Object}, //{meal name, start time, end time}

        // caption: { type: String, required: true },

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

        // stations: { type: [String], required: true },

        // locationImg: { type: String, required: true },

        // menuItems: { type: [ItemSchema], required: true }

    }

);

module.exports = mongoose.model("DiningCourt", schema);