**תיק למתכנת**

מסד נתונים ו ממשק חיצוני:

For the server we imported express

import express from 'express';

import cors from 'cors';

import bodyParser from 'body-parser';

with express we used the given api to initialize a server that will bind to our function which are connected to the DB with an api that will be explained soon.  
The initialization of the server using the express api:

which in the (\*) we bind each post put and get method of the http protocol   
to the appropriate function   
  
after that with (\*\*) we use app.listen() to start the server which is given the port and IP which is localhost 127.0.0.1  
// Creating an instance of the Express application

const app = express();

app.use(bodyParser.json({ limit: '50mb' }));

app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));

app.use(cors());

// Routes

(\*)

app.use('/api', recipeRoutes);

app.use('/api/users', usersRoutes);

// Starting the server.  
(\*\*)

const server = app.listen(config.port, () => {

  console.log(`Server is live @ ${config.hostUrl} on port ${config.port}`);

});

server.on('error', (error) => {

  if (error.code === 'EADDRINUSE') {

    console.log(`Port ${config.port} is already in use. Trying another port...`);

    // Attempt to handle the error, for example, by trying a different port.

  } else {

    console.error('An error occurred while starting the server:', error);

  }

});

we used firebase/FireStore as our DB which granted us with an built in api for getting the information that was stored.  
  
using a .env file we stored our DB info which allowed us to connect to the DB.  
  
import { initializeApp } from 'firebase/app';

import config from './config.js';

import { getFirestore } from 'firebase/firestore';

import { getStorage } from 'firebase/storage';

const firebase = initializeApp(config.firebaseConfig);

const db = getFirestore(firebase);

const storage = getStorage(firebase);

export {firebase, db, storage};

using the api below we managed to set,get,update,approach and delete data   
import {

  collection,

  deleteDoc,

  doc,

  getDoc,

  getDocs,

  setDoc,

  updateDoc,

} from "firebase/firestore";

setDoc(doc(collection(db, "recipe"), name), newData);  
getDocs(collection(db, "recipe"));  
deleteDoc(recipeRef);  
deleteDoc(recipeRef);  
and the below api which allowed us to get a reference of a specific item in the DB  
doc(db, "recipe", id);

we used the above api and managed to create our own api between the FRONEND and the BACKEND.

פונקציות עיקריות:

FRONTEND:

async createRecipe(formData)

function get the form that was taken from the html tags, and then, using api upload the data to the erver.  
  
name, description, ingredients, image, calories, fat, proteins, preparation   
  
async getRecipes({page, pageSize, searchQuery})

function that sends the page, the page size and a search query, than using api to the server returns the appropriate recipes.

async getRecipeById(id)

function for getting a specific recipe by id using an api to the server

async updateRecipe(id, formData)

function that update existing recipe, using the id of the recipe sends new data, than again, using a proper api updating the recipe that in the DB   
  
async deleteRecipe(id)

function which deleting existing recipe that in the DB using an api to the server

BACKEND:

Each function of the above using an appropriate function in the BACKEND side,  
this allowed us to separate dependencies during the development.  
  
async createRecipe(req, res)

Function which receives all the needed information about a recipe   
and used the api talked about in the first section of this report   
 “ setDoc(doc(collection(db, "recipe"), name), newData); ”

Later on we used   
async getRecipes(req, res)  
async getRecipe(req, res)  
async updateRecipe(req, res)   
async deleteRecipe(req, res)

Where each of the above receiving the appropriate data and using the api mentioned before.

שימוש בכלי בינה מלאכותית :

In addition to all mentioned earlier:   
the following code which creating a popup on screen   
was generated by GPT-4

import { useState, useEffect } from 'react';

// Custom hook for managing popup messages

const usePopupMessage = (initialVisibility = false, duration = 3000) => {

  // State variables to manage visibility and message content

  const [isVisible, setIsVisible] = useState(initialVisibility);

  const [message, setMessage] = useState('');

  // Effect to handle popup visibility and message duration

  useEffect(() => {

    let timer;

    if (isVisible) {

      // Set a timer to hide the popup after duration

      timer = setTimeout(() => {

        setIsVisible(false);

        // Clear the message after duration

        setMessage('');

      }, duration);

    }

    // Clean up function to clear the timer

    return () => clearTimeout(timer);

  }, [duration, isVisible]);

  // Function to show a popup with a message

  const showPopup = (msg) => {

    setMessage(msg);

    setIsVisible(true);

  };

  // Return state variables and function to be used by components

  return { isVisible, message, showPopup };

};

export default usePopupMessage;

import { useState } from 'react';

// Custom hook for managing a confirmation dialog

const useConfirmationDialog = () => {

    // State variables to manage visibility and callbacks

  const [isVisible, setIsVisible] = useState(false);

  const [onConfirmCallback, setOnConfirmCallback] = useState(null);

  const [onCancelCallback, setOnCancelCallback] = useState(null);

  // Function to show the confirmation dialog

  const showConfirmationDialog = (onConfirm, onCancel) => {

    setIsVisible(true);

    // Set onConfirm callback to execute onConfirm function and hide the dialog

    setOnConfirmCallback(() => {

      return async () => {

        await onConfirm();

        // Hide the dialog

        setIsVisible(false);

      };

    });

    // Set onCancel callback to execute onCancel function and hide the dialog

    setOnCancelCallback(() => {

      return () => {

        onCancel();

        // Hide the dialog

        setIsVisible(false);

      };

    });

  };

  // Function to handle confirmation

  const handleConfirm = () => {

    onConfirmCallback && onConfirmCallback();

  };

  // Function to handle cancellation

  const handleCancel = () => {

    onCancelCallback && onCancelCallback(); };

  // Return state variables and functions to be used by components

  return {

    isVisible,

    setIsVisible,

    showConfirmationDialog,

    handleConfirm,

    handleCancel,

  };

};

export default useConfirmationDialog;

ספריות וסביבות מיוחדות:

  
