**מפרט ארכיטקטוני של המערכת**

**ניהול קורונה לקופת חולים**

שכבת הנתונים

מאגר המידע בmongo DB.

שם ה: schema Clients.

Schema Clients - מתארת עבור כל חבר בקופת חולים את פרטיו האישים ובנושא הקורונה.

פרטי לקוח:

 fullName : String, required

  idCard : String, required

  address : String, required

  dateOfBirth :Date

  mobilePhone : String, required

  telephone : String

  positive\_result\_date :Date

  recovery\_date :Date

  vaccination\_dates :[ Date ]

  vaccine\_manufacturers :[ String ]

שכבת היישום

צד השרת NODE.JS -

שמכיל:

1. תיקיית models- מכילה את Schema Clients
2. תיקיית controllers – מכילה פונקציות:addClient getAllClients
3. תיקיית routers- מכילה ניתובים לפונקציות ב controllers

שכבת המצגת

צד לקוח, API –REACT

שכולל את :

1. App.js – דף ראשי. מכיל את הקומפוננטה Routings
2. Index.js – הדף שרץ בדפדפן. רינדור הקומפוננטה App(פעם אחת).
3. Api.js – getAllClients : קבלת כל מערך הלקוחות מהמסד נתונים. addClient: הוספת לקוח למסד נתונים.
4. Routings.js – מכיל את הניתובים לדפים השונים.
5. Layout.js – מערך, סרגל הניתובים ל:
6. - Manager: כפתור לגישת מנהל

- Sign In : כניסת לקוח

- :Sign Up הרשמת לקוח

1. SignUp.js – הרשמה. הוספת לקוח למסד נתונים.
2. SignIn.js – כניסה לפרטי הלקוח.
3. Manager.js – כניסה למנהל קופת חולים ע"י הקשת קוד (2552), כולל גישה לדפים הבאים:
4. Clientsdetails.js – פרטי כל הלקוחות.
5. Activeclients.js – חישוב מספר לקוחות חולים.
6. Unvacciatedclients.js – חישוב מספר לקוחות לא מחוסנים.
7. Style.css – עיצוב הפרוייקט.

**החיבור בין השכבות**

המסד נתונים לשרת:

קובץ .env

DB\_CONNECT="mongodb+srv://hilag:hilamongodb@manage.3gvqyed.mongodb.net/?retryWrites=true&w=majority&appName=Manage"

PORT=8000

בקובץ db.js

const connect = () => {

  console.log(process.env.DB\_CONNECT)

  if(process.env.DB\_CONNECT)

  mongoose

    .connect(process.env.DB\_CONNECT)//('mongodb://127.0.0.1:27017/cookbookDb')

    .then(() => {

      console.log("Connected to MongoDB");

    })

    .catch((error) => {

      console.error("Unable to connect to MongoDB.", error);

    });

    else

      console.log("You're not connected to MongoDB");

};

השרת לצד לקוח:

הגדרה בAPI נקודות קצה עבור פעולות שונות, כגון קבלת נתונים, יצירת נתונים, עדכון נתונים ומחיקת נתונים.

getAllClients – הצגת כל הלקוחות מהמסד נתונים

async getAllClients(req, res) {

    try {

      const client = await clientModel.find();

      if (client instanceof Error) {

        res.status(404).json({ message: client.message, error: true });

      } else {

        res.status(200).json(client);

      }

    } catch (error) {

      res.status(400).json({ message: error });

    }

  }

addClient- הוספת לקוח למסד נתונים

async addClient(req,res){

    console.log(req.body)

    try {

        const newClient=await clientModel.create(req.body);

        res.status(201).json(newClient);

      } catch (error) {

        console.error(`error: ${error}`);

        return error;

      }

  }

updateClient- עדכון מצב של אחד הלקוחות

async updateClient(req, res) {

    try {

      const Client = await clientModel.findOneAndUpdate(req.params.id);

      if (Client instanceof Error) {

        if (Client.message === 'Client\_NOT\_FOUND') {

          return res.status(404).json({ message: 'Client\_NOT\_FOUND' });

        } else {

          return res.status(400).json({ message: Client, error: true });

        }

      }

      return res.status(200).json(Client);

    } catch (error) {

      return res.status(400).json({ message: error.message, error: true });

    }

  }

deleteClient- מחיקת לקוח

async deleteClient(req, res) {

    try {

      const Client = await clientModel.findOneAndDelete({id:req.params.id});

      if (Client instanceof Error) {

        if (Client.message === 'Client\_NOT\_FOUND') {

          return res.status(404).json({ message: 'Client\_NOT\_FOUND' });

        } else {

          return res.status(400).json({ message: Client, error: true });

        }

      }

      return res.status(200).json(Client);

    } catch (error) {

      return res.status(400).json({ message: error.message, error: true });

    }

  }

import axios from "axios";

const baseUrl = 'http://localhost:5000';

const backendService = {

    clients:{

        getAllClients: () => axios.get(`${baseUrl}/clients`).then(response => response.data),

        addClient: (client) => {console.log(client);

            return axios.post(`${baseUrl}/clients/addClient`, client).then(response =>{console.log(response.data); return response.data})}

    }